TransSkel Programmer's Notes

7: Multitasking Issues

Who to blame: Paul DuBois, dubois@primate.wisc.edu
Note creation date: 11/22/93
Note revision: 1.01
Last revision date: 12/09/93
TransSkel release: 3.04

This Note describes some problems involved with multitasking and how to deal with them when writingTransSkel applications.

Macintosh system software now supports operating system events for the purpose of allowing multitasking (switching between multiple simultaneously executing applications). The way you want the system to treat your application is specified by the application's 'SIZE' resource. The TransSkel programmer's manual describes the relevant flags in this resource and basic mechanics of getting TransSkel to tell your application about multitasking events. The rest of this Note describes how to handle special problems related to multitasking.

Suspend/Resume Events

Problem 1: Suppose your application is receiving suspend/resume events and handling window activations itself. It might seem that the easiest way to do this is simply to call the window's activate procedure. There are two difficulties with this. First, you may not know what that procedure is for some arbitrary window (for instance, the window might be managed by a set of library routines that export no information about the window's handlers). Second, if the window is a modeless dialog, it doesn't have an activate procedure. In order to deal with this, the new function `SkelActivate()` may be called. You tell it which window to handle, and whether the window should be activated or deactivated:

```
SkelActivate (WindowPtr w, Boolean active);
```

This function synthesizes a fake activate event and runs it through TransSkel's event router. Since TransSkel should know your window handler functions, it can make sure the event is routed properly.

Note: `SkelActivate()` knows how to send activate events modeless dialogs, but the right thing to do with them is to hide them on a suspend and show them again on a resume. You must do that yourself. (The TransSkel demonstration application DialogSkel shows a

simple way to do this.)

Problem 2: When your application is suspended, you always deactivate the active window. Then, when you click in one of its windows, the application is brought forward. However, you may not necessarily want to reactivate the frontmost window. For instance, if you clicked in a window other than the frontmost window, what should you do? One option is to bring the clicked-in window forward. Another is to leave the window stacking order unchanged, no matter which window is clicked in.

If you want to bring the clicked-in window forward, you can simply wait for a mouse click event and select the window in response to it. (When you click in any window other than the frontmost one, the system seems to generate a click in addition to bringing your application forward.) However, it may be preferable to avoid activating the frontmost window since you're just going to deactivate it again anyway when the mouse click for the other window is processed. For example, if the window has scroll bars and push buttons, you'd highlight them only to immediately unhighlight them. Functionally this is harmless, but it's pretty ugly.

I don't know of a "canonical" method for ignoring the activate for the frontmost window when another window is going to be activated right away, but the following fragment seems to work. It checks whether there's a mouse click for a non-frontmost window in the event queue, and, if so, declines to activate the frontmost window:

```
EventRecord  event;
GrafPtr      eventPort;
Boolean      doActivate;

doActivate = true;
if (EventAvail (mDownMask, &event))
{
       (void) FindWindow (event.where, &eventPort);
       if (eventPort != FrontWindow ())
              doActivate = false;
}
if (doActivate)
       SkelActivate (FrontWindow (), true);
```

If you want to leave stacking order unchanged instead of bringing the clicked-in window forward, you can test for a mouse click and remove it from the event queue if one is present so you don't respond to it later when the event loop sees it. The DialogSkel demonstration application can be examined as an instance of why this might be desirable, and how to accomplish it.

Getting Front Clicks

The `getFrontClicks` flag in the 'SIZE' resource is set if your application wants to receive mouse clicks that "bring your application into the foreground when the user clicks in your application's frontmost window." That's what Inside Macintosh VI (5-16) says. What IM doesn't specify is what to do if the click isn't in the frontmost window. Should the click also be passed to the window handler in that case, too?

It seems to me that they should, in order to preserve the "respond to every click" feel of this type of application. TransSkel implements this for applications that have the `getFrontClicks` flag set in the 'SIZE' resource by first selecting the window and then passing the mouse click to the window's click handler rather than just selecting the window.

References:

Inside Macintosh, Volume VI. Event Manager chapter