TransSkel Programmer's Notes

1: Changes in Apple menu support

Who to blame: Paul DuBois, dubois@primate.wisc.edu
Note creation date: 01/29/91
Note revision: 1.04
Last revision date: 01/06/94
TransSkel release: 3.00

This Note describes the changes in Apple menu support that have been implemented in TransSkel release 3.0.

Change 1: Multiple Application Items in Apple Menu

Release 1 and 2 Apple menu support

The `SkelApple()` call in releases 1 and 2 has the following form:

```
SkelApple (StringPtr aboutTitle, void (*aboutProc) (void));
```

The first argument, if non-`nil`, is typically "About *application*…" and is inserted as the first item in the Apple menu, followed by a gray line and any available desk accessories. If the argument is `nil`, the menu is loaded only with desk accessories.

The second argument is a callback function to be invoked whenever the first Apple menu item is selected. If `nil`, or if the first argument is `nil`, no callback is performed.

This support was sufficient for most applications, which install either zero or one items into the Apple menu. But some applications wish to have a second item, e.g., a "Help" item; possibly some applications have a use for more than two.

There are a number of ways to accomplish this goal, each with various strengths and weaknesses. The following (non-exhaustive) list illustrates a few methods, of which the last was selected for implementation.

Characteristics deemed desirable in a solution were:

- Method should be extensible, i.e., handle an arbitrary number of application items.
- Minimal modification of existing programs
- Not depend on system-specific features

Approach 1: add arguments to `SkelApple()`

One could add another item title and callback function, e.g.,

```
SkelApple ("\pAbout X…", DoAbout, "\pHelp", DoHelp);
```

Problems:

The calling structure of `SkelApple()` changes. By itself this is not an insuperable difficulty, given the number of other changes made for release 3. But what if the application really wants *three* items in the Apple menu? Or more than three? What if it wants seventeen!? "Yeah, right," you say. Well, OK — but still, this approach is not easily extensible without further changes. Another change is needed every time another extra item is wanted. It would be better to institute a mechanism allowing an arbitrary number of items and which would require no further changes.

Approach 2: define a new function

One could define another function in addition to `SkelApple()`, e.g.,

```
SkelHelp ("\pHelp", DoHelp);
```

Problems:

- This approach requires no changes to `SkelApple()`, but again, what if the application really wants more than two items in the Apple menu? A different function must be invented for each instance.
- A method is needed for inserting the new item into the menu, such as `InsMenuItem()`. That sacrifices 64K ROM compatibility, since those machines don't have the `InsMenuItem()` call. One might get around this by recreating the menu anew and destroying the old one, but the complexity involved in doing so makes such an approach unattractive.
- There is additional bookkeeping involved in keeping track of the callback functions. One could minimize this by requiring the callback function only for `SkelApple()` call, and changing its semantics so that the item number is passed to it.

Approach 3: allow `SkelApple()` to be called multiple times

The semantics of `SkelApple()` could be changed. Rather than allowing it to be called a single time, it could be called once for each application item to be installed. The first call would cause the menu to be created and its handler installed; subsequent calls would modify the existing menu, adding the new item with `InsMenuItem()`. The result might look like this:

```
SkelApple ("\pAbout X…", DoAbout);
SkelApple ("\pHelp", DoHelp);
```

Problems:
Same as for approach 2.

Approach 4: change meaning of SkelApple() arguments

Instead of supposing that the title string specifies a single menu item, it could be used to specify multiple items. To support this, the callback function must be changed to accept a single argument, the item number, instead of taking no argument. In this way only a single call to `SkelApple()` is necessary (as in releases 1 and 2), and the calling structure remains unchanged. For example:

```
SkelApple ("\pAbout X…;Help", DoAppleMenu);
```

It is not necessary for TransSkel to parse the item string to determine the number of items the application wants to insert because selections can be distinguished as application or DA items based on whether they are above or below the gray line item separating them.

Problems:
(i) The callback function calling structure must be changed (although balanced against this is the fact that the callback structure becomes isomorphic to that of selection callbacks for other menu handlers). (ii) The application must be careful not to pass "–" items in the item string, or *TransSkel* won't properly determine whether to invoke the callback or start a desk accessory. That seems reasonably unlikely.

The last approach seems to provide the desired extensibility with the least mandatory modification of existing applications. Those applications which install no items in the Apple menu remain unchanged. Those installing a single item must change the callback to accept a single `Integer` argument. Those wishing to put more than one item in the Apple menu will be able to do so.

Change 2: `SkelApple()` No Longer in TransSkel Core

`SkelApple()` has been removed from *TransSkel.c* and move into its own source file. When the TransSkel routines are built as a project (not a library) and included in another application's project, this has the effect that `SkelApple()` and supporting routines are not linked into the application if it never calls `SkelApple()`.

Change 3: Apple Menu ID Number No Longer 1

The menu ID number formerly used by `SkelApple()` to create the Apple menu was 1, which is in the range of numbers reserved for system resources. It is now 128. Applications no longer need to know a magic ID number, however. The ID is now available as `skelAppleMenuID` in *TransSkel.h*.

Caveat

Macintosh Technical Note TB 35 seems to indicate that applications should only count on being able to include the standard "About…" item in the Apple menu. It may be that applications taking advantage of the changes described in this Note will do so at the cost of MultiFinder compatibility. Then again, maybe not. The changes described here were implemented using only standard Menu Manager calls and do not, following the warnings in TN TB 35, rely on any knowledge of the internal structure of menu data structures. No problems have thus far been reported.

References

Macintosh Technical Note TB 35: MultiFinder Miscellanea