TransSkel

Programmer's Notes

14: inForeground on Launch

Who to blame: Hans van der Meer, hansm@fwi.uva.nl
Note creation date: 05/15/95
Note revision: 1.00
Last revision date: 07/21/95
TransSkel release: 3.22

This Note describes a corrected initialization of the `inForeground` variable in TransSkel release 3.22. [The code fragments actually added to TransSkel differ slightly from those described below — Paul DuBois]

1. Introduction

The `inForeground` variable can be queried in order to find out whether the program executes in the foreground or in the background. However, when the application is launched directly into the background, the initialization statement in TransSkel.c

```
static BooleaninForeground = true;
```

is evidently incorrect.

2. Corrected Initialization

Using the Process Manager – if available of course – enables the programmer to interrogate the system as to whether the program starts up in foreground or background. After that, the resume and suspend events are keeping `inForeground` correct.

We first ensure the presence of the necessary Gestalt selectors along with the declaration of `inForeground`.

```
/* Process Manager Gestalt Selectors */
#ifndef gestaltOSAttr
#define gestaltOSAttr     'os   '     /* ProcessManager attributes */
#endif
static BooleaninForeground = true;
```

In `SkelInit()` we add the following code and we are in business. Unless, however, the Process Manager is unavailable. In the latter case we must be content with blind initialization.

```
ProcessSerialNumber    PSN1, PSN2;
Boolean            answer;
...
/* initialize inForeground with Process Manager query */
if ( SkelGestaltCheck(gestaltOSAttr, -1)
                && GetCurrentProcess(&PSN1) == noErr
                && GetFrontProcess(&PSN2) == noErr
                && SameProcess(&PSN1, &PSN2, &answer) == noErr )
        inForeground = answer;
else /* fall back to best guess */
        inForeground = true;
```

3. Convenience routine for Gestalt checking

In the above code the routine `SkelGestaltCheck()` is used. For convenience a new query function has been defined for testing a Gestalt feature. I believe that the exuberant use of Gestalt selectors warrants this.

In *TransSkel.h* for outside access:

```
pascal Boolean
SkelGestaltCheck(const OSType selector, const short featurecode);
```

In *TransSkel.c* the function declaration.

```
pascal Boolean
SkelGestaltCheck(const OSType selector, const short featurecode)
{
        long result;

        if ( !hasGestalt || Gestalt(selector, &result) != noErr )
                return false;

        return ( featurecode < 0 ) ? true :
                        ( (result & (1 << featurecode)) ? true : false );
}
```

This function returns `true` if the specific `selector` is present and the bit denoted by the `featurecode` is set. Note that a value of –1 for `featurecode` can be used as a mere check on the presence of the selector, without any interest in its value.