Other Topics

Nested Event Loops

The typical TransSkel application sets up, calls `SkelEventLoop()`, then calls `SkelCleanup()` and exits when `SkelEventLoop()` returns. This is not strictly necessary. The only rule is that every invocation of `SkelEventLoop()` must be terminated by a call to `SkelStopEventLoop()`. `SkelEventLoop()` may, if desired, be executed again.

It is also possible to invoke `SkelCleanup()` and install a new set of handlers between calls to `SkelEventLoop()`.

More generally, one may call `SkelEventLoop()` recursively, or call `SkelCleanup()` while `SkelEventLoop()` is executing. The only thing to watch out for is that this sort of thing should not be done from within window handler idle-time procedures. One way in which this property may be used is to force all update events to be processed (so that all windows are completely redrawn) before any other events are handled. This is done by retrieving the current event mask and background procedure. Then set the event mask so that only update events are requested, and install a background procedure that calls EventAvail (with the same mask). Then call `SkelEventLoop()`. The background procedure does nothing as long as there are update events pending, and calls `SkelStopEventLoop()` as soon as there aren't any more. When `SkelEventLoop()` returns, restore the previous event mask and background procedure. The original call to `SkelEventLoop()` resumes operation as before.