Subject: Oddball Event Handling (Longer than it Ought to Be)
Posted by davidf on Sun, 31 Dec 2000 04:46:43 GMT
View Forum Message <> Reply to Message

Folks,

Speaking of oddball event handling (We were speaking about oddball event handling, weren't we?), I've been fooling around with an interesting project where I am building a bunch of compound widgets (out of objects, naturally). These widgets can show up in various incarnations, sometimes as part of another widget program, sometimes in their own top-level base, etc.

The event handler for the compound widget is simple, it just reads a "message" stored in the user value of each widget that can cause an event, and the message tells the event handler what method to call to handle the event (via a Call_Method command). That part is simple, simple, simple, and you can read all about it in my book. :-)

But, when the event is through being processed there is a possibility that some other widget, which is NOT part of the compound widget, might want to know about it. For example, the compound widget might change the character size of a plot, and the widget that is re-drawing the plot might want to know about the character size thingy being changed, etc.

This is absolutely no problem if the compound widget is part of the widget hierarchy of the plot re-draw widget, but it *was* a problem if the compound widget happened to be in its own top-level base. What would happen is that the event I wanted to send would have the wrong top-level base ID, thereby causing the info structure with all the program information for the other widget program to become lost.

Oh, man. I hate this kind of problem!

I sometimes solve it by putting the info structure in a pointer and passing the pointer here, there, and everywhere. But, uughhh. That involves a *bunch* of modifications, and then I have to explain this program to the client, and I've been talking about simpler is better, and .... Well, you get the idea.

By the way, you aren't doing anything else today, are you?
I mean, it being the first of the year and all. Because
I haven't even gotten to the *point* of this article yet. :-)

So, I got to thinking that I wanted something that was
simpler than modifying my whole info structure scheme.
What I wanted to know was the identifier of that *other*
top-level base over there, so I could substitute it's
identifier in event.top for the one I had, before I passed
the event along. And what I knew was the identifier of
the parent of the compound widget, which just happened
to *always* be in the hierarchy of that widget over there.

(OK, here comes the point of this too long story.)

All I had to do was traverse UP the widget hierarchy
until I got to the top!

Now, you have to understand, I din't have no computer
learnin in schol. Recursive functions and I have no
common ground. But a recursive function is what I needed.
(Can something that goes UP be recursive!?)

So, anyway, after fooling around for 10 minutes I
wrote something that--somehow, for some reason known
only to JD and a couple of others, probably--works!!!!

I don't have a clue. Really. But I thought it might
be useful for y'all, in case you ever wanted to do
something like this. You just pass it a widget ID, and
it spits out the widget identifier at the top of that
widget hierarchy.

Pretty neat, huh? I *love* IDL sometimes. :-)

Happy New Year,

David

```
 ********************************************************** ***
FUNCTION FindTLB, startID

; This function traces up the widget hierarchy to find the top-level base.

FORWARD_FUNCTION FindTLB
parent = Widget_Info(startID, /Parent)
IF parent EQ 0 THEN RETURN, startID ELSE parent = FindTLB(parent)
RETURN, parent
```

END
********** ******

P.S. Let's just say I don't really want to hear about it
if this thing doesn't work. It works for me. :-)

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155
