Subject: Re: Communication between top-level bases.
Posted by davidf on Mon, 01 Jun 1998 07:00:00 GMT
View Forum Message <> Reply to Message

David Foster (foster@bial1.ucsd.edu) writes a lot of excellent
advice about widget programming, including this:

> A much more elegant way to share data *within* a widget is to create
> a STATE structure (I always call it STATE{} to avoid confusion) which
> contains the data you want to share. Then, before you register it
> with the XMANAGER, copy this structure into the UVALUE of the
> top-level widget:
>
> widget_control, base, set_uvalue=state, /no_copy

I couldn't have written this part of the article any better
myself, and (as Dave mentions) I suggest something very similar
in my IDL Programming Techniques book. But I developed those
ideas nearly two years ago now, and I find myself using
something quite a bit different today. Poor Dick, who knows
I am working on ideas for the next programming book, calls
me about every other day to see what our Current Best
Practice is now. When I am in the writing mode, it
changes frequently. :-)

But what I do today is put the state structure (I often call
it the "info" structure) in an IDL pointer variable, like this:

```
statePtr = Ptr_New( {image:image, drawID:drawID, ...}, /No_Copy )
```

By placing the pointer in the UValue of the top-level base (TLB),
or by passing the pointer to other widget programs, I can
assure that any program module that has the pointer has the
latest and greatest information that the pointer points to.
The only tricky part is de-referencing the pointer. My event
handler code might look like this:

```
PRO JUNK_EVENT, event
Widget_Control, event.top, Get_UValue=statePtr
Widget_Control, (*statePtr).drawID, Get_Value=windex
WSet, windex
TV, (*statePtr).image
END
```

The pointer has to be de-referenced to a structure by the
use of parentheses before I can de-reference a field in the
structure.

If my image field were itself a pointer (it usually is) defined
like this:

```
statePtr = Ptr_New( {image:Ptr_New(image), drawID:drawID, ...} )
```

Then the image de-referencing would look like this:

```
TV, *((*statePtr).image)
```

It makes my IDL code look more and more like C code, but
it's not so bad when you get used to it. :-)

Using a pointer for the state or info structure has several
advantages. First, I've eliminated all the No_Copy keywords
in getting them into an out of the event handler. Second,
I don't have to worry about putting them "back" in the UValue
of the TLB, since if I make changes to the thing pointed to,
I change the thing itself. Third, I don't have to worry about
whether the info structure is "checked in" when I call another
event handler directly from within my event handler code. This
gives me more flexibility in how I handle events.

The only thing I have to remember to do is destroy the pointer
when the widget program dies. I always do this with a widget
cleanup routine, assigned with the CLEANUP keyword on the
XMANAGER call:

```
XManager, 'junk', tlb, Cleanup='JUNK_CLEANUP'
```

The cleanup routine is where all the pointers, objects,
pixmaps, etc. are cleaned up in my program. The cleanup
routine is called when the TLB is destroyed. This simple
cleanup routine might look like this:

```
PRO JUNK_CLEANUP, tlb
Widget_Control, tlb, Get_UValue=statePtr
Ptr_Free, (*statePtr).image
Ptr_Free, statePtr
END
```

If I ever find the time to write the 2nd Edition of my
book, this is how I'll explain things then. :-)

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/