Subject: Re: Multiple widgetized windows in one application
Posted by davidf on Mon, 26 Apr 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Kenneth P. Bowman (bowman@null.edu) writes:

> I found a typo in my code and have made the program work, but that does
> not mean that I understand /GROUP_LEADER, /JUST_REG, and EVENT_PRO.
>
> Is /GROUP_LEADER *only* used to handle *killing* a widget hierarchy
> containing multiple top level bases?

The GROUP_LEADER keyword is used to assign a "group leader" to
a widget hierarchy. When the group leader "dies", then
every member of the group dies (is "killed" or "destroyed")
as well. Every widget program you write will probably
want to be defined with a GROUP_LEADER keyword:

```
  PRO JUNK, GROUP_LEADER=group_leader
```

You don't need to check this keyword, just pass the
"group_leader" variable along to the XMANAGER command,
where you will use the XMANAGER GROUP_LEADER keyword to
assign this variable as the group leader of the program's
top-level base. If the "group_leader" variable is undefined,
no matter. This is one of the rare times you don't have to
check a keyword before you use it.

```
   XMANAGER, 'junk', tlb, GROUP_LEADER=group_leader
```

Now your widget program is set up to be called from within
some other widget program. For example, from an event
handler like this:

```
   PRO OTHER_PROGRAM_CALL_JUNK_EVENT, event
     Junk, Group_Leader=event.top
   END
```

Now, when this other program is killed, your JUNK program
will be killed as well.

> Should I use EVENT_PRO when I create the top-level bases, or should I call
> XMANAGER with /JUST_REG? For consistency, why not always use EVENT_PRO
> when creating a TLB and then call XMANAGER without any arguments?

EVENT_PRO (or EVENT_FUNC) can be used to assign an event handler
to any widget EXCEPT a widget that is being *directly* managed by
XMANAGER. An event handler is assigned to the widget that is being

directly managed by XMANAGER (for example, the widget "tlb" in
the example above) by using the EVENT_HANDLER keyword to the
XMANAGER command:

```
XMANAGER, 'junk', tlb, EVENT_HANDLER='JUNK_EVENT', $
  GROUP_LEADER=group_leader
```

If this EVENT_HANDER keyword is not used, then a *default*
event handler is assigned to the "tlb" widget by using the
name the program is registered with ("junk" in this case)
and appending an "_event" to the name. In other words, had
I not used the EVENT_HANDLER keyword in the command above,
an event handler of the same name would have *still* been
assigned to the "tlb" widget.

If you do use EVENT_PRO to assign an event handler for
a top-level base being directly managed by XMANAGER, you
will find exceedingly strange things going on in your
widget program, if it works at all. Believe me, you
DON'T want to do this.

I can't think of any particular reason to use JUST_REG
in a widget program. In the old days, when you couldn't get
to the IDL command line when a widget program was running,
and you wanted to start, say, three widget programs all at
once, you might "just register" two of the programs before
you actually started the third one. This way they would all
be on the display and they would all be running when the
last program started actively managing all of the programs
that were "registered" with XMANAGER.

These days I either spawn other widget programs (e.g. XLOADCT)
from within a widget program, or I make my widget programs
non-blocking and just run as many as I like from the IDL
command line.

(Incidentally, don't use KILL_NOTIFY with the top-level base
being managed directly by XMANAGER either. Use the CLEANUP
keyword with XMANAGER to assign a cleanup routine that is called
when this widget dies. KILL_NOTIFY can be used if you like to
assign a cleanup routine to any widget NOT being directly
managed by XMANAGER.)

> Just out of curiosity, how many widget programmers prefer to write a
> separate event-handler for each widget or group of widgets, and how many
> prefer to have a single event handler routine?

More to the point, how many widget programmers like to write

programs that are easy to extend and maintain? And how many
like to write programs that they have to futz around with
for hours to make a simple change?

Event handlers should be assigned logically according to what
they do, in my opinion. I quite often assign an event handler
to a "menu button" so that events caused by this button's
children will bubble up and all be handled by this one event
handler. This is especially the case if all the different
events are similar in some way. Perhaps they all do some
kind of image processing.

> P.S. At least the problem in my code wasn't in the COMMON block!

Thank goodness. :-)

Cheers,

David

P.S. Here is a modification to the example program I
wrote earlier. This time there is only one XMANAGER
command and the program works as before.

--
```
PRO TEST_EVENT, event
Widget_Control, event.id, Get_UValue=thisValue
IF event.type NE 0 THEN RETURN
Print, ''
Print, thisValue
END

PRO TEST

tlb = Widget_Base(XOffset=50, Title='Manifolds')
draw = Widget_Draw(tlb, XSize=200, YSize=200, $
  UValue='Manifold', Button_events=1)

tlb1 = Widget_Base(XOffset=150, Group_Leader=tlb, Title='Display 1')
draw1 = Widget_Draw(tlb1, XSize=200, YSize=200, $
  UValue='Display 1', Button_events=1, Event_Pro='Test_Event')

tlb2 = Widget_Base(XOffset=250, Group_Leader=tlb, Title='Display 2')
draw2 = Widget_Draw(tlb2, XSize=200, YSize=200, $
  UValue='Display 2', Button_events=1, Event_Pro='Test_Event')

Widget_Control, tlb, /Realize
Widget_Control, tlb1, /Realize
```

```
Widget_Control, tlb2, /Realize

XManager, 'test', tlb, EVENT_HANDLER='Test_Event'
END
```

--
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

[Note: This follow-up was e-mailed to the cited author.]
