Subject: Multiple widgetized windows in one application
Posted by bowman on Fri, 23 Apr 1999 07:00:00 GMT
View Forum Message <> Reply to Message

I'm trying to write a widget program that has multiple widgetized
windows. I have done the following (in outline form)

```
PRO MANIFOLD

base_id = WIDGET_BASE(TITLE='Manifolds')
  ... create subwidgets for this window

display_1_id = WIDGET_BASE(TITLE = 'Display 1', GROUP_LEADER = base_id)
  ... create subwidgets for this window

display_2_id = WIDGET_BASE(TITLE = 'Display 2', GROUP_LEADER = base_id)
  ... create subwidgets for this window

WIDGET_CONTROL, display_1_id, /REALIZE
WIDGET_CONTROL, display_2_id, /REALIZE
WIDGET_CONTROL, base_id,      /REALIZE

XMANAGER, 'MANIFOLD', display_1_id, /JUST_REG
XMANAGER, 'MANIFOLD', display_2_id, /JUST_REG
XMANAGER, 'MANIFOLD', base_id
```

The 'Display 1' and 'Display 2' windows have identical widget structures
(but different widget id's, of course).

I have a single event handling routine, MANIFOLD_EVENT

The program works, mostly. All three windows are realized. Events in
base_id and display_1_id windows are passed to MANIFOLD_EVENT. For some
reason, however, no events in 'Display 2' are passed to MANIFOLD_EVENT. I
know this because I print out part of every event that comes to
MANIFOLD_EVENT.

I must say that the documentation of GROUP_LEADER, JUST_REG, and NO_BLOCK
may have reached a new high in IDL documentation obscurity, approaching
the PS and X device descriptions. I may not be using them correctly. I
don't need to handle command line input.

Am I doing something wrong here?

Thanks, Ken

--
Dr. Kenneth P. Bowman, Professor 409-862-4060
Department of Meteorology 409-862-4466 fax
Texas A&M University bowmanATcsrp.tamu.edu
College Station, TX 77843-3150 Replace AT with @

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.]

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

> 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

Well, typo, after all! Told ya!
GROUP_LEADER : the only use I have for it is to kill a dependent top-level base, like a modal dialog.
JUST_REG : I have never used it and from the description I see no use for it in what I am doing.
EVENT_PRO: This is just redirector for the event handling procedure, to let you bypass the default "your_pro_EVENT" procedure name. XMANAGER still provides event handling (unless you are advanced enough to use WIDGET_EVENTdirectly, which I never needed).

> 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?

In my first useful widget application (not big, <1500 lines of code) I used one event handling procedure. As I expanded it, it quickly grew out of reasonable limits, it became difficult to find and change things. Now I have separate event handling routines for logically related tasks (modes of application operation: data loading, application setup, integration, time series analyses, etc.). This

way event handling parts don't screw up each other and it is easy to expand one of them without going through a lot of code.
Cheers,
Pavel

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

In article <MPG.118d4a2f510ea10598977b@news.frii.com>, davidf@dfanning.com (David Fanning) wrote:

> Kenneth P. Bowman (bowman@null.edu) writes:
>
>> I'm trying to write a widget program that has multiple widgetized
>> windows.
>>
>> The program works, mostly. All three windows are realized. Events in
>> base_id and display_1_id windows are passed to MANIFOLD_EVENT. For some
>> reason, however, no events in 'Display 2' are passed to MANIFOLD_EVENT. I
>> know this because I print out part of every event that comes to
>> MANIFOLD_EVENT.
>>
>> Am I doing something wrong here?
>
> I don't see anything manifestly wrong. I wrote a simple test
> program putting your principles to work. The program, named TEST,
> works perfectly. I include it below.
>
> You don't mention what version of IDL you are running,
> but mine runs fine in IDL 5.2 on Windows NT.

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?

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?

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?

Regards, Ken

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

--

Dr. Kenneth P. Bowman, Professor 409-862-4060
Department of Meteorology 409-862-4466 fax
Texas A&M University bowmanATcsrp.tamu.edu
College Station, TX 77843-3150 Replace AT with @

Subject: Re: Multiple widgetized windows in one application
Posted by davidf on Thu, 29 Apr 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Stein Vidar Hagfors Haugan (steinhh@ulrik.uio.no) writes:

> Most widget programs that I've written have been using a
> single event handling routine, structured along the lines
> of:

I'm not going to get into a pissing contest with Stein Vidar over how to write IDL widget programs. Lord knows, I'm smart enough to know when I'm over-matched even if I'm not smart enough to keep my mouth shut all the time.

But I will say this. If you know what you are doing, you can pretty much write widget programs any way you please. If you need evidence, open up just about any program in the lib directory and look at that spaghetti code. Older programs written by David Stern (the founder of RSI) tend to use a lot of WIDGET_EVENT calls. I say, the more power to him. David CLEARLY knows what he is doing.

But I would argue that if any of us run-of-the-mill hackers put a WIDGET_EVENT command in our program, then we have written a program that pretty much ought to be shot.

For most of us (who are not programmers, but scientists who have a job to do), anything that makes our programs easier to build and easier to maintain has to be a positive development. In my experience teaching a LOT of people to write IDL programs, I've found that having many event handlers rather than one big event handler leads to significantly better programs.

I'm not saying it has to be this way. I'm saying

that until you know exactly what you are doing, you
are more likely to write better programs this way
than any other way.

Some of you have heard me tell the Barry Lopez
story "Directions" in my classes:

"You may have come across a detailed set of instructions,
a map...At first glance it seems excellent, the best
a man is capable of. But your confidence is misplaced.
Throw it out. It is too thin, it's not the sort of map
that can be followed by a man who knows what he is doing.
The coyote...even the crow...will regard it with suspicion."

A man (or woman) who knows what they are doing will find
the right way to write a widget program *without* a map
and without advice from so-called IDL experts. :-)

Cheers,

David

--
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

Subject: Re: Multiple widgetized windows in one application
Posted by steinhh on Thu, 29 Apr 1999 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:

> 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.

Most widget programs that I've written have been using a single event handling routine, structured along the lines of:

```
PRO program_event,ev
  widget_control,ev.top,get_uvalue=info,/no_copy
  widget_control,ev.id,get_uvalue=uval

  CASE uval OF

  "QUIT":BEGIN
    widget_control,ev.top,/destroy
    return
    ENDCASE

  "SAYHELLO":program_sayhello,info,ev

  "DRAW":program_draw,info,ev
  :
  :
  END

  widget_control,ev.top,set_uvalue=info,/no_copy
END
```

Most of these were written before pointers were invented, hence the /no_copy switch to avoid memory copying when fetching the info structure. This also creates the need to put the info structure back.

For complicated tasks I write separate *subroutines*, but the events go through the same handler.

This is to keep things simple, and to minimize the number of places where things can go wrong, like forgetting to put back the info structure, doing "refresh" operations if necessary etc..

It also makes it easier to put in a single line like

```
  help,info,ev,/structure
```

in just *one* place to do debugging of the event handling.

In my opinion, if an event influences or uses the state of the application (i.e. the info structure) directly, it should go through the same event handler as the

others.

If it *doesnt* influence or use the info structure, make
a compound widget!

So, in my opinion, the event_pro/event_func mechanism should
only be used when implementing compound widgets.

And if you find yourself using the event_pro/event_func
for groups of widgets controlling a subsystem of your
event application, I'd say chances are that you might be
better off implementing a compound widget - if nothing
else it will at least exercises the brain into thinking
object orientation (the compound widget being the object).

Regards,

Stein Vidar
