Subject: Child widget group leader of its own TLB?
Posted by nhbkmich on Mon, 30 Jun 1997 07:00:00 GMT
View Forum Message <> Reply to Message

Dear IDL experts,

is it possible (allowed) to make a child widget a group leader of its own
top level base?
For me (I'm running IDL 4.0.1 under HP-UX 10.20) this works fine, as long as
the child widget is the last one in the hierachy. But if it isn't...
Consider the following example:

```
PRO GroupTest1, Id = Id
  Top = WIDGET_BASE(/ROW)
  Text = WIDGET_TEXT(Top)
  Exit = WIDGET_BUTTON(Top, VALUE = 'Dummy')
  WIDGET_CONTROL, Top, GROUP_LEADER = Text, /REALIZE
  Id = Text
END
```

The text widget is the one, whose id is passed outside and which is supposed
to act as group leader. Go ahead:

```
IDL> grouptest1, id = id
IDL> widget_control, id, /destroy
% X windows protocol error:  (BadWindow (invalid Window parameter)).
```

Apparently, the window manager has something to complain about. Sometimes
- not reproducible for me - this even crashes IDL completely. I've fiddled
around with different ways to avoid this error. It disappears when updating
is switched off before destroying the text widget. But, according to the
manual, manipulating the update state has only effect on Motif based window
systems. So that solution would be rather unportable.

I ended up with a kind of group leader cascade:

```
PRO GroupTest1, Id = Id
  Top = WIDGET_BASE(/ROW)
  Text = WIDGET_TEXT(Top)
  Exit = WIDGET_BUTTON(Top, GROUP_LEADER = Text, VALUE = 'Dummy')
  WIDGET_CONTROL, Top, /REALIZE
  WIDGET_CONTROL, Top, GROUP_LEADER = Exit
  Id = Text
END
```

This method is meant to ensure, that there is no widget behind "Text" when
"Top" is destroyed. This appears to work, but is it reliable?
Is there a defined order of evaluation in such a cascade?

Any help appreciated
Michael
--
****
Michael Steffens Institut f. Meteorologie u. Klimatologie
Tel.: +49-511-7624413 Universitaet Hannover
email: Michael.Steffens@mbox.muk.uni-hannover.de Herrenhaeuser Str. 2
steffens@muk.uni-hannover.de D-30419 Hannover

PGP fingerprint = FA BE 6C 1C F6 C3 EC 33 DD 42 6B 7F DE CF 84 B8

Subject: Re: Child widget group leader of its own TLB?
Posted by David Foster on Thu, 03 Jul 1997 07:00:00 GMT
View Forum Message <> Reply to Message

nhbkmich@rrzn-user.uni-hannover.de wrote:
>
> Dear IDL experts,
>
> is it possible (allowed) to make a child widget a group leader of its own
> top level base?
> For me (I'm running IDL 4.0.1 under HP-UX 10.20) this works fine, as long as
> the child widget is the last one in the hierachy. But if it isn't...
> Consider the following example:
>
> PRO GroupTest1, Id = Id
> Top = WIDGET_BASE(/ROW)
> Text = WIDGET_TEXT(Top)
> Exit = WIDGET_BUTTON(Top, VALUE = 'Dummy')
> WIDGET_CONTROL, Top, GROUP_LEADER = Text, /REALIZE
> Id = Text
> END
>
> The text widget is the one, whose id is passed outside and which is supposed
> to act as group leader. Go ahead:
>
> IDL> grouptest1, id = id
> IDL> widget_control, id, /destroy
> % X windows protocol error: (BadWindow (invalid Window parameter)).
>
> This method is meant to ensure, that there is no widget behind "Text" when
> "Top" is destroyed. This appears to work, but is it reliable?
> Is there a defined order of evaluation in such a cascade?
>

Michael -

I think you may be confused about the concept of GROUP_LEADER and
it's effect on widget destruction. When you create a widget heirarchy,
any and all widgets that are "below" the TLB will be destroyed
automatically when you destroy the TLB. By "below" I mean that you can
trace back from a widget up through the heirarchy and get to the
TLB. So there is no need for a cascade as you mention above.

The GROUP_LEADER keyword is intended for situations where you have
one widget program up and running, and then you with to call another
widget (a program or a modal popup) from within that first program.
In the second program you specify the TLB of the first widget as the
GROUP_LEADER. When you do this, the second widget will be destroyed
if the first is destroyed. You can have a whole series of programs
up, and have them all destroyed at once. This is also useful when
a program has several (or many!) widgets that all must
go when the parent widget is destroyed.

Hope this helps.

Dave
--

~~~~~~ ~~~~~~~
David S. Foster Univ. of California, San Diego
Programmer/Analyst Brain Image Analysis Laboratory
foster@bial1.ucsd.edu Department of Psychiatry
(619) 622-5892 8950 Via La Jolla Drive, Suite 2200
La Jolla, CA 92037
~~~~~~ ~~~~~~~

"I have this theory that if we're told we're bad,
then that's the only idea we'll ever have.
But maybe if we are surrounded in beauty,
someday we will become what we see." - Jewel Kilcher

Subject: Re: Child widget group leader of its own TLB?
Posted by nhbkmich on Fri, 04 Jul 1997 07:00:00 GMT
View Forum Message <> Reply to Message

David Foster (foster@bial1.ucsd.edu) wrote:
: Michael -

: I think you may be confused about the concept of GROUP_LEADER and
: it's effect on widget destruction. When you create a widget heirarchy,
: any and all widgets that are "below" the TLB will be destroyed
: automatically when you destroy the TLB. By "below" I mean that you can

: trace back from a widget up through the heirarchy and get to the
: TLB. So there is no need for a cascade as you mention above.

In the actual case, there is some need. The widget application passes
the Id of a (child) text widget outside, rather than its top level Id.
The reason is, that I want to be able to manipulate this text widget from
outside in more ways than possible by FUNC_GET_VALUE and PRO_SET_VALUE.
Things like dis- or enabling editing, dis- or enabling events created by
the text widget or establishing a separate event handler.
But one possible manipulation is also destroying the text widget. In
this case, there would only remain a kind of ruin of a mini window frame
containing meaningless buttons and labels.
I intended the whole application to quit in such a situation as if "Done"
had been pressed.

: The GROUP_LEADER keyword is intended for situations where you have
: one widget program up and running, and then you with to call another
: widget (a program or a modal popup) from within that first program.
: In the second program you specify the TLB of the first widget as the
: GROUP_LEADER. When you do this, the second widget will be destroyed
: if the first is destroyed. You can have a whole series of programs
: up, and have them all destroyed at once. This is also useful when
: a program has several (or many!) widgets that all must
: go when the parent widget is destroyed.

I know, this is the most common use of widget groups. But is it restricted
to this use? Neither the online help nor the printed manuals state something
like that. Furthermore, wouldn't this imply, that only top level bases can
reasonably be group leaders as well as _group members_? If so, why do all
widget creation functions, including WIDGET_TEXT, WIDGET_BUTTON, WIDGET_LABEL
etc., which can never create top level widgets, accept the GROUP_LEADER
keyword?

: Hope this helps.

Yes it does. Maybe my goal is a bit too exotic. But I'm not yet really
convinced :-)

Michael
--
********** **********
Michael Steffens Institut f. Meteorologie u. Klimatologie
Tel.: +49-511-7624413 Universitaet Hannover
email: Michael.Steffens@mbox.muk.uni-hannover.de Herrenhaeuser Str. 2
steffens@muk.uni-hannover.de D-30419 Hannover

PGP fingerprint = FA BE 6C 1C F6 C3 EC 33 DD 42 6B 7F DE CF 84 B8