Subject: widget layout
Posted by Ted Cary on Fri, 12 Jul 2002 17:20:03 GMT
View Forum Message <> Reply to Message

Hello all,

Is there a way to query a realized widget base to see what kind of
layout it enforces? I want a function that tells me if a base uses a
row, a column, or a bulletin board layout. If this information can be
obtained directly with WIDGET_INFO or WIDGET_CONTROL, I could not tell
from reading the help files.

Iï¿½ve tried analyzing the geometry of the base with respect to the
geometry of its children in order to determine the layout, but this only
works if there is already more than one child. Itï¿½s occurred to me to
add and then delete a "small" child (maybe a null string widget label)
to see which dimension of the base expands, but the problems with this
are obvious. Probably I am missing something.

Any help is appreciated.

TC

Subject: Re: widget layout
Posted by Ted Cary on Tue, 16 Jul 2002 17:41:45 GMT
View Forum Message <> Reply to Message

"David Fanning" <david@dfanning.com> wrote in message
news:MPG.179cf996c74af3a698992e@news.frii.com...
>
> This automatic resizing idea is interesting to me.
> I have a library of "widget objects", which wrap all
> the IDL widgets up into object wrappers. Then, rather
> then creating a widget hierarchy, you create an object
> hierarchy, based on IDL container objects.

> Putting the automatic resizing feature into one of these lower-level
> objects would be simple, I think.

Actually, I was thinking of implementing something like your object
container widget hierarchy specifically for my resizer helper object,
although it seemed like a lot of work for just this this routine.
Typically, I had not realized the full potential of the method, although I
now use object widgets frequently and even remember you hinting about your
project before.

The way the resizer is supposed to work is by searching a widget hierarchy

and finding all the resizable widgets in the TLB--by default, draw widgets. It resizes their windows around the other non-resizable realized widgets, whose dimensions remain fixed. The resizing depends upon the layout of the draw widgets in a base, which is why I needed to know this information. For example, if a TLB with three draw widgets in a column is resized, the default behavior is to distribute the change in base height evenly among the three draw windows. Sizing of TLBs with menubars is handled automatically, and there are options to fix individual draw widget window dimensions as well as to set maximum and minimum dimensions.

The problem with the routine is the "searching the widget hierarchy" part, since to be completely general it needs to find all resizable widgets and treat them differently according to the layouts of their bases. I used to manually call the object with the resizable IDs and layout info, but it's neater if it figures this information out for itself using just the TLB ID. Now I use WIDGET_INFO with /SIBLING and /CHILD to search through the hierarchy rooted at the TLB. The next step is to create some kind of object tree structure with "resizer" nodes corresponding to every resizable widget. A resize at the TLB root node should then work its up the tree to all the subnodes.

Not surprisingly this last part is not working well yet. Probably this would be a lot easier if instead of constructing an object tree hierarchy corresponding to an already-realized widget, the resizer could query a widget hierarchy like yours, that was realized as an object tree in the first place. But aren't the widget hierarchies in IDL essentially stored internally as object trees anyway? If only there were just a WIDGET_WHERE function and a truly comprehensive WIDGET_GET_PROPERTY....

Anyway, what I can show you now is how the individual TLB resizer node works. If you already have the widget tree structure, maybe it's all you need, and probably you can improve upon it. In fact, the TLB node works fine for my typical application--some draw widgets in a TLB, resized around some labels and buttons, etc.... I'll remove the "print" and "help" statements and add some comments first, so give me a few days.

TC

Subject: Re: widget layout
Posted by David Fanning on Tue, 16 Jul 2002 19:22:55 GMT
View Forum Message <> Reply to Message

Ted Cary (tedcary@yahoo.com) writes:

> Actually, I was thinking of implementing something like your object
> container widget hierarchy specifically for my resizer helper object,
> although it seemed like a lot of work for just this this routine.

I can tell you that wrapping widgets up in objects is
a hell of a lot of work! :-)

Although, to be fair, you only have to do it once. But
there are more decisions to be made then you would have
initially thought, and there is a lot of hemming and hawing
about where keywords should be implemented, etc. We have
opted for a subset of functionality rather than trying for
absolutely everything possible.

But this automatic resizing of widgets is attractive to
us because it seems relatively simple to implement with
our current object structure.

> The way the resizer is supposed to work is by searching a widget hierarchy
> and finding all the resizable widgets in the TLB--by default, draw widgets.

This is quite easy in an object containment hierarchy, because
a GET method can easily find an object of any "type" included in
the container.

> It resizes their windows around the other non-resizable realized widgets,
> whose dimensions remain fixed. The resizing depends upon the layout of the
> draw widgets in a base, which is why I needed to know this information.
> For example, if a TLB with three draw widgets in a column is resized, the
> default behavior is to distribute the change in base height evenly among the
> three draw windows. Sizing of TLBs with menubars is handled automatically,
> and there are options to fix individual draw widget window dimensions as
> well as to set maximum and minimum dimensions.

The object widget INIT method knows exactly how baseWidget objects are
laid out, so setting up the "resizer" portion of the widget object
should be trivial. A simple keyword could mark the widget object as
potentially "resizeable".

> The problem with the routine is the "searching the widget hierarchy" part,
> since to be completely general it needs to find all resizable widgets and
> treat them differently according to the layouts of their bases.

A "resize" request could simply bubble down the containment hierarchy,
with each object passing the message on to its children, who could
respond or not, depending upon their internal resize information. The
message could become increasingly detailed as it travels down the
hierarchy, so that a widget would know exactly what it was suppose to
do.

> But aren't the widget hierarchies in IDL essentially stored

> internally as object trees anyway? If only there were just a WIDGET_WHERE
> function and a truly comprehensive WIDGET_GET_PROPERTY....

In our system, GetProperty requests can bubble up to parent containers
if they can't be fulfilled by the object who is initially requested to
supply the information. I think that is what you mean here. It is a
powerful concept.

(I'm always a bit confused if we are bubbling "up" or "down". This
is one of the reasons I haven't written an object book yet. I'm sure
to get a great deal of criticism for the way I think about these things.
I think [but am not absolutely sure] that I am internally consistent,
anyway. Sigh...)

> Anyway, what I can show you now is how the individual TLB resizer node
> works. If you already have the widget tree structure, maybe it's all you
> need, and probably you can improve upon it. In fact, the TLB node works
> fine for my typical application--some draw widgets in a TLB, resized around
> some labels and buttons, etc.... I'll remove the "print" and "help"
> statements and add some comments first, so give me a few days.

I look forward to seeing it.

Cheers,

David

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