Subject: Widget layout in BASE to get table in IDL 4.0
Posted by nmw on Thu, 05 Oct 1995 07:00:00 GMT
View Forum Message <> Reply to Message

I have an application which used to work in IDL 3, but no longer produces sensible output in IDL 4.0.

What the routine is supposed to do is produce a table of widgets with the rows and columns nicely aligned. Using IDL 3 I could do this by creating a BASE widget and using the COLUMN attribute to tell it how many columns to use. Then I could fill the base with the widgets and the same number of widgets would be put into each column and each widget would be the same height, pretty much like a Motif RowColumn widget.

With IDL 4 this is no longer the case. The widgets are all packed into the BASE widget and it doesn't even put the same number in each column. The result is a free-form complete mess. I have tried creating additional BASE, /COLUMN=1 widgets of the main BASE and then putting the table widgets in these. This does allow me to put the correct number of widgets in each column, but the widgets in each column are of different heights so they don't align across. I cannot use the XSIZE and YSIZE attributes because they are ignored for ROW/COLUMN BASE widgets.

BTW, this is not regarded as a bug, but as a new *feature*. I have been told that this is the way it is now *meant* to work.

Does anyone have any idea how it might be possible to create a table of widgets of different types which are aligned to a grid? I don't want to have to use a Bulletin Board type BASE and specify the XSIZE, YSIZE, XOFFSET, and YOFFSET of every child because a) it's very tedious and requires alteration every time a new row or column is added, b) it only works on one screen with one font - a different screen or font requires all the values to be changed.

--
Nigel Wade, System Administrator, Ionospheric Physics Group,
University of Leicester, Leicester, LE1 7RH, UK
E-mail : nmw@ion.le.ac.uk
Phone : +44 (0)116 2523568, Fax : +44 (0)116 2523555

Subject: Re: widget layout
Posted by notspecified on Fri, 12 Jul 2002 17:42:47 GMT
View Forum Message <> Reply to Message

On Fri, 12 Jul 2002 13:20:03 -0400, Ted Cary <tedcary@yahoo.com> wrote:

> 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.
>

> geometry of its children in order to determine the layout, but this only

> 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.
>

Hmm, well, if there isn't a direct way of doing it, I guess you could
save that sort of information in the base's uvalue as part of the
state of the widget.

Matt Feinstein does not include his email address
in the text of usenet postings.

Harvard Law of Automotive Repair: Anything that goes away
by itself will come back by itself.

Subject: Re: widget layout
Posted by Ted Cary on Sat, 13 Jul 2002 23:03:53 GMT
View Forum Message <> Reply to Message

Matt Feinstein wrote:

>
> Hmm, well, if there isn't a direct way of doing it, I guess you could
> save that sort of information in the base's uvalue as part of the
> state of the widget.

Thanks, but that's what I used to do in this kind of situation. The routine
was conceived in the first place so that I would not have to resort to
storing widget geometry information in base uvalues, since keeping track of
such details during TLB resizing can be a pain for all but the most simple
GUIs. I was hoping for something more general that worked along the lines
of the WIDGET_INFO function.

TC

Subject: Re: widget layout
Posted by Matt Feinstein on Sun, 14 Jul 2002 17:10:08 GMT
View Forum Message <> Reply to Message

In article <3D30B1DA.E0041F96@yahoo.com>, Ted Cary <tedcary@yahoo.com>
wrote:

> Matt Feinstein wrote:
>
>>
>> Hmm, well, if there isn't a direct way of doing it, I guess you could
>> save that sort of information in the base's uvalue as part of the
>> state of the widget.
>
> Thanks, but that's what I used to do in this kind of situation. The routine
> was conceived in the first place so that I would not have to resort to
> storing widget geometry information in base uvalues, since keeping track of
> such details during TLB resizing can be a pain for all but the most simple
> GUIs. I was hoping for something more general that worked along the lines
> of the WIDGET_INFO function.
>
> TC
>

I admit-- I'm new to IDL, so let me continue in the 'why doesn't the
easy way work?' mode. Why can't you define a 'descendent' function of
widget_info() that wraps around widget_info(), adds a keyword, and does
what you want by looking at the base uvalue (where, I'm assuming, the
appropriate state information is stored)?

Matt

Subject: Re: widget layout
Posted by Mark Hadfield on Sun, 14 Jul 2002 21:49:04 GMT
View Forum Message <> Reply to Message

"Matt Feinstein" <mfein@clark.net> wrote in message
news:140720021309014241%mfein@clark.net...
> In article <3D30B1DA.E0041F96@yahoo.com>, Ted Cary
> <tedcary@yahoo.com> wrote:
>
>> Thanks, but that's what I used to do in this kind of situation.
>> The routine was conceived in the first place so that I would not
>> have to resort to storing widget geometry information in base
>> uvalues, since keeping track of such details during TLB resizing
>> can be a pain for all but the most simple GUIs. I was hoping for
>> something more general that worked along the lines of the

>> WIDGET_INFO function.
>
> I admit-- I'm new to IDL, so let me continue in the 'why doesn't the
> easy way work?' mode. Why can't you define a 'descendent' function
> of widget_info() that wraps around widget_info(), adds a keyword,
> and does what you want by looking at the base uvalue (where, I'm
> assuming, the appropriate state information is stored)?

Isn't it a pity that widgets are not objects? Then one could simply
subclass the widget base, add an appropriate tag to the class
structure and add corresponding keywords to the Init, GetProperty &
SetProperty methods.

Actually, several people have implemented the "widgets-as-objects"
idea. I have a widget-base class in my Motley library. It doesn't
currently store the row/column info, but it would be straightforward
to extend it so it did.

Final comment: if the base is a child then the base's UVALUE is not an
ideal place to store info relating to the base's own structure,
because the UVALUE may be needed by other widgets further up in the
heirarchy. In IDL 5.5 the RSI developers inadvertently provided a
solution: create an invisible, unrealised context menu as the first
child of the base and store data there!

--
Mark Hadfield "Ka puwaha te tai nei, Hoea tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)

Subject: Re: widget layout
Posted by Ted Cary on Sun, 14 Jul 2002 22:11:47 GMT
View Forum Message <> Reply to Message

"Matt Feinstein" <mfein@clark.net> wrote in message
news:14072002130901424 1%mfein@clark.net...

> I admit-- I'm new to IDL, so let me continue in the 'why doesn't the
> easy way work?' mode. Why can't you define a 'descendent' function of
> widget_info() that wraps around widget_info(), adds a keyword, and does
> what you want by looking at the base uvalue (where, I'm assuming, the
> appropriate state information is stored)?
>
> Matt

The easy way does work, it is just not as general as I would like. When I

write that I want a method that works like WIDGET_INFO, I mean that the method should work on *any* realized widget base, not just those that were initialized with certain uvalues. WIDGET_INFO would not be useful in its own right if it only worked on widgets with state information stored in uvalues; it would be functionally equivalent to WIDGET_CONTROL, GET_UVALUE=. In fact, the uvalue method worked for me in the past, but it was not easy. For every resizable widget base, I wrote code to set geometry information in its uvalue. Then in a TLB resize event handler I would write more code to get and manipulate the info in this uvalue. It seems simple, but this is very annoying and time-consuming programming, as there are several bugs and special cases that have to be handled. Just look at some past posts on "TLB Widget Resizing."

I got tired of doing all this every time I wanted a resizable TLB. These days I use a "TLB_Resizer" helper object that automatically keeps track of everything. What used to be a hassle is now reduced to a few lines of code: "resizer=OBJ_NEW('TLB_RESIZER', TLB)", etc. What bothers me about this routine is the indirect way it determines the layouts enforced by base children of a TLB, by deriving the information from their WIDGET_INFO geometry structure, but still this seems a lot better than going back to manually storing layout type in uvalues for every base in the widget hierarchy. A transparent alternative would be better than one that requires more coding.

Finally, since this is really one of my few useful ideas, I was thinking about making it available to the IDL community along with some other stuff I 'm ironing out. My resizer's utility would be limited if every programmer had to store base layout information in uvalues when creating widget hierarchies; it even limits what type of uvalue the programmers can use! I want the helper object to continue to work like it does now: if you have a widget program, add a line or two of code to the widget definition module, and suddenly everything is dynamically resizable, no matter how complex the hierarchy is. This is the way widget resizing should be-it is not a problem worth all these posts :).

TC

Subject: Re: widget layout
Posted by David Fanning on Mon, 15 Jul 2002 22:24:47 GMT
View Forum Message <> Reply to Message

Ted Cary (tedcary@yahoo.com) writes:

> Finally, since this is really one of my few useful ideas, I was thinking
> about making it available to the IDL community along with some other stuff I
> 'm ironing out. My resizer's utility would be limited if every programmer
> had to store base layout information in uvalues when creating widget

> hierarchies; it even limits what type of uvalue the programmers can use! I
> want the helper object to continue to work like it does now: if you have a
> widget program, add a line or two of code to the widget definition module,
> and suddenly everything is dynamically resizable, no matter how complex the
> hierarchy is. This is the way widget resizing should be-it is not a problem
> worth all these posts :).

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. This is really
a powerful way of programming, because we have figured
out how integrate widget events into our object hierarchy.
Events propagate up an object hierarchy in just the same
way they propagate up a widget hierarchy. In fact, widget
events can be sent to objects that are not even widgets!

In any case, our widget objects are all based on
low-level objects that give them the ability to
report what they are doing, self-document themselves,
reference count themselves so they are not destroyed
until they are no longer needed, etc. Putting the
automatic resizing feature into one of these lower-level
objects would be simple, I think, and -- as you say --
would give all the widget objects the ability to
resize themselves automatically.

If you care to share your code, I have a ready-made
system to test it out. :-)

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