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:1407200213090142 41%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