Subject: Re: Obtain all child widgets
Posted by DavidF[1] on Mon, 20 Aug 2012 18:32:05 GMT
View Forum Message <> Reply to Message

Russel writes:

> Ok, this must be easy and I'm just having "one of those days". I have a widget hierarchy, and it creates a series of buttons using the /menu feature in widget_button. But, I'd like to get the widget IDs of all the children. Suppose:
>
>
>
> base=widget_base(/col)
> but=widget_button(base,value='top',/menu)
> but1=widget_button(but,value='First level',/menu)
> b11=widget_button(but1,value='First button',uval='firstbutton',group=but)
> b12=widget_button(but1,value='Second button',uval='secondbutton',group=but)
> but2=widget_button(but,value='First level',/menu)
> b21=widget_button(but2,value='First button',uval='firstbutton',group=but)
> b22=widget_button(but2,value='Second button',uval='secondbutton',group=but)
>
> Now, I'd like to call something which will take as input the variable 'but' (as above) and return (in this example) a four element vector of widget IDs. I know widget_info(/child) but that only returns the first child. I suppose I could "nest" the group leaders, where the current button has the previous as a leader. Then just recursively call widget_info(/child), but that seems like a lot of work and requires the children be "serially" listed.
>
> I've scoured the widget_info and widget_control pages. It really feels like this should not only be possible, but pretty simple. I must be missing something...

You aren't missing anything. :-)

> Any ideas?

Take all those keystrokes where you are typing "group=but" (not needed)
and use them to save the button IDs you are interested in retaining into
the info structure of your program. Or, not. I haven't used
button IDs in 10 years, probably. I just use the button value
in my button event handlers. Makes the code *much* easier to read! ;-)

Cheers,

David

Subject: Re: Obtain all child widgets
Posted by Russell Ryan on Mon, 20 Aug 2012 20:14:23 GMT

On Monday, August 20, 2012 2:32:05 PM UTC-4, Coyote wrote:
> Russel writes:
>
>
>
>> Ok, this must be easy and I'm just having "one of those days". I have a widget hierarchy, and it creates a series of buttons using the /menu feature in widget_button. But, I'd like to get the widget IDs of all the children. Suppose:
>
>>
>
>>
>
>>
>
>> base=widget_base(/col)
>
>> but=widget_button(base,value='top',/menu)
>
>> but1=widget_button(but,value='First level',/menu)
>
>> b11=widget_button(but1,value='First button',uval='firstbutton',group=but)
>
>> b12=widget_button(but1,value='Second button',uval='secondbutton',group=but)
>
>> but2=widget_button(but,value='First level',/menu)
>
>> b21=widget_button(but2,value='First button',uval='firstbutton',group=but)
>
>> b22=widget_button(but2,value='Second button',uval='secondbutton',group=but)
>
>>
>
>> Now, I'd like to call something which will take as input the variable 'but' (as above) and return (in this example) a four element vector of widget IDs. I know widget_info(/child) but that only returns the first child. I suppose I could "nest" the group leaders, where the current button has the previous as a leader. Then just recursively call widget_info(/child), but that seems like a lot of work and requires the children be "serially" listed.
>
>>
>
>> I've scoured the widget_info and widget_control pages. It really feels like this should not only be possible, but pretty simple. I must be missing something...
>
>
>
> You aren't missing anything. :-)

>
>
>
>> Any ideas?
>
>
>
> Take all those keystrokes where you are typing "group=but" (not needed)
>
> and use them to save the button IDs you are interested in retaining into
>
> the info structure of your program. Or, not. I haven't used
>
> button IDs in 10 years, probably. I just use the button value
>
> in my button event handlers. Makes the code *much* easier to read! ;-)
>
>
>
> Cheers,
>
>
>
> David

Thanks David,
I guess I don't understand your question. I do process the events in the event handler by querying the structure name and the UNAME from the event ID. I sort the event out two case statements, first to process which type of event (e.g. WIDGET_BUTTON) then second to process which widget was activated (e.g. this button vs. that button). All the while, I store some basic information in the UVALUE for each button. Now at some point in the code, I'd like to collect the UVALUE data from all buttons which are in the menu hierarchy (ie. those which are children of the first call to widget_button). Of course, i can store that data in a state/info structure (truth be told, this widget is my first foray into object widgets, so it'd be the self structure), but I expect the number of buttons to change as the code matures. So, I don't like the idea of sticking things in the state vector (as I plan to tweak the menu list extensively, then I'll have to constantly update the define procedure).

I was sorta thinking like the /FIND_BY_UNAME or /CHILD feature in widget_info(). As I understand it, t=widget_info(event.id,/child) returns the widget ID of the first child. I'd like it to return the widget ID of all the children. I can believe I don't need the group_leader keyword when building the widget, but I thought then I'd just have to find all widgets which have that same group_leader value. Another purpose I can envision, is to find which widget IDs have been assigned. I realize that, as you create a new widget, IDL will use the next valid widget ID. And, you can use widget_info(/valid_id) to test if a given long integer is assigned to an existing widget, but you'd need to know a list of long integers to test. However, in most cases you do know the ID of the TLB, so this routine would return the list of widget IDs which are children of this TLB --- then any long not in this list is available for a new widget assignment. Granted, I've never encountered

this problem, but it's the same idea...

I think I can work around the lack of a /ALL_CHILDREN like keyword, but now my curiosity is fully piqued by this.

Russell

Subject: Re: Obtain all child widgets
Posted by DavidF[1] on Mon, 20 Aug 2012 20:52:17 GMT
View Forum Message <> Reply to Message

Russel writes:

> I guess I don't understand your question. I do process the events in the event handler by querying the structure name and the UNAME from the event ID. I sort the event out two case statements, first to process which type of event (e.g. WIDGET_BUTTON) then second to process which widget was activated (e.g. this button vs. that button).

I send all my button events to the same event handler. That way
I don't have to sort out whether the event is a button event. It
wouldn't be there otherwise! (Actually, I use groups of related buttons,
but you get the idea.)

Then, I just get the VALUE of the button and branch on that:

```
Widget_Control, event.id, Get_Value=buttonValue
CASE buttonValue OF
```

I can see at a glance which button event I am working on.

> All the while, I store some basic information in the UVALUE for each button. Now at some point in the code, I'd like to collect the UVALUE data from all buttons which are in the menu hierarchy (ie. those which are children of the first call to widget_button). Of course, i can store that data in a state/info structure (truth be told, this widget is my first foray into object widgets, so it'd be the self structure), but I expect the number of buttons to change as the code matures. So, I don't like the idea of sticking things in the state vector (as I plan to tweak the menu list extensively, then I'll have to constantly update the define procedure).

This is probably a matter of style I guess. But, I prefer to have
all my program information in one place, not scattered all to hell
and gone over my program interface. :-)

> I was sorta thinking like the /FIND_BY_UNAME or /CHILD feature in widget_info(). As I understand it, t=widget_info(event.id,/child) returns the widget ID of the first child. I'd like it to return the widget ID of all the children. I can believe I don't need the group_leader keyword when

building the widget, but I thought then I'd just have to find all widgets which have that same group_leader value. Another purpose I can envision, is to find which widget IDs have been assigned. I realize that, as you create a new widget, IDL will use the next valid widget ID. And, you can use widget_info(/valid_id) to test if a given long integer is assigned to an existing widget, but you'd need to know a list of long integers to test. However, in most cases you do know the ID of the TLB, so this routine would return the list of widget IDs which are children of this TLB --- then any long not in this list is available for a new widget assignment. Granted, I've never encountered this problem, but it's the same idea...

Ugh! Sounds ugly. I like to write things as simply as possible.
That way I don't have to think too much when I am trying to figure
out what the hell I was thinking months ago!

> I think I can work around the lack of a /ALL_CHILDREN like keyword, but now my curiosity is fully piqued by this.

I hope it is piqued in a "direct and simple" direction. ;-)

Cheers,

David
