

---

Subject: Re: widget\_problem

Posted by [Paul Van Delst\[1\]](#) on Wed, 23 Jul 2008 14:34:13 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

d.poreh@gmail.com wrote:

> On Jul 22, 5:52 pm, d.po...@gmail.com wrote:

>> On Jul 17, 8:32 pm, Paul van Delst <Paul.vanDe...@noaa.gov> wrote:

>>

>>

>>

>>> Justus Skorps wrote:

>>>> > Justus

>>>> > stil can not fix it. i have Liam E.Gumley's book but .....

>>>> > Cheers

>>>> > Dave

>>>> after u load your arrays (lets call them A) store them with

>>>> widget\_control, event.top, set\_uvalue=A, /nocopy

>>>> You have to change 'event.top' that it fits your program...

>>>> In your second button you can now load the arrays with

>>>> widget\_control, event.top, get\_uvalue=A, /nocopy

>>> but don't forget to put them back into your uvalue when you're done!

>>>> It is useful to

>>>> -store the data in the main widget

>>>> -use a structure to store every data you want

>>> I also tend to use procedures to get the Info state:

>>> ; Routine to get the Info state

>>> PRO GetState, ID, Info, No\_Copy = No\_Copy

>>> ; -- Get pointer

>>> WIDGET\_CONTROL, ID, GET\_UVALUE = InfoPtr

>>> IF ( PTR\_VALID( InfoPtr ) EQ 0 ) THEN \$

>>> MESSAGE, 'State Information pointer is invalid'

>>> ; -- Get state information structure

>>> IF ( N\_ELEMENTS( \*InfoPtr ) EQ 0 ) THEN \$

>>> MESSAGE, 'State information structure is undefined'

>>> IF ( KEYWORD\_SET( No\_Copy ) ) THEN BEGIN

>>> Info = TEMPORARY( \*InfoPtr )

>>> ENDIF ELSE BEGIN

>>> Info = \*InfoPtr

>>> ENDELSE

>>> IF ( Info.Debug EQ 1 ) THEN PRINT, 'GetState'

>>> END

>>> and to set the info state

>>> ; Routine to set the Info state

>>> PRO SetState, ID, Info, No\_Copy = No\_Copy

>>> ; -- Get pointer

>>> WIDGET\_CONTROL, ID, GET\_UVALUE = InfoPtr

>>> IF ( PTR\_VALID( InfoPtr ) EQ 0 ) THEN \$

>>> MESSAGE, 'State information pointer is invalid'

```

>>> ; -- Set state information structure
>>> IF ( N_ELEMENTS( Info ) EQ 0 ) THEN $
>>>   MESSAGE, 'State information structure is undefined'
>>> IF ( KEYWORD_SET( No_Copy ) ) THEN BEGIN
>>>   *InfoPtr = TEMPORARY( Info )
>>> ENDIF ELSE BEGIN
>>>   *InfoPtr = Info
>>> ENDELSE
>>> IF ( (*InfoPtr).Debug EQ 1 ) THEN PRINT, 'SetState'
>>> END
>>> My widget event handlers then do something like:
>>> FUNCTION ComponentTest_LogLin_Event, Event
>>> ; -- Get main info state
>>>   GetState, Event.Top, Info
>>> ; -- Print debug statement if required
>>>   IF ( Info.Debug EQ 1 ) THEN PRINT, 'ComponentTest_LogLin_Event'
>>> ; -- Set the selected variable number index
>>>   Info.LogLin_Index = Event.Value
>>> ; -- Save info state
>>>   SetState, Event.Top, Info
>>> ; -- Display the result
>>>   ComponentTest_Display, Event.Top
>>>   RETURN, 0
>>> END
>>> Note how I call GetState and then SetState. Because I tend not to use /no_copy, it's
>>> really only an issue when I update a component of the info state (like in my example
>>> above). But, if you *do* use /no_copy, then I think you have to call SetState again to
>>> replace the info pointer.
>>> cheers,
>>> paulv
>> Thanks Justus and Paulv
>> You help me very much. And it was very useful.
>> Cheers
>> Dave
>
> Hi Justus
> I encounter a now problem:
> Say I have set and get 2 arrays by this method:
> widget_control, event.top, set_uvalue=A, /nocopy
>  $\frac{1}{2}$ 
> widget_control, event.top, get_uvalue=A, /nocopy
>
> and from another button:
> widget_control, event.top, set_uvalue=B, /nocopy
>  $\frac{1}{2}$ 
> widget_control, event.top, get_uvalue=B, /nocopy
>
>

```

```

>
> Now I want to get both A&B in another button like this:
>
>
> widget_control, event.top, get_uvalue=A, /nocopy
> widget_control, event.top, get_uvalue=B, /nocopy
>
> But widget just accepts one array. How I can take this two arrays
> simultaneously in another button?

```

Don't store A and B separately. Store them both. When you create the widget hierarchy, do something like,

```

Info = {A:PTR_NEW(/ALLOCATE_HEAP), $
        B:PTR_NEW(/ALLOCATE_HEAP) }
InfoPtr = PTR_NEW( Info )
WIDGET_CONTROL, Top_Level_Base_ID, SET_UVALUE = InfoPtr

```

This assumes you don't know the sizes of A and B ahead of time, hence the PTR\_NEW(/ALLOCATE\_HEAP). Also, if you store a structure, then you can easily add more data to your widget info state as your add more functionality to your GUI application, but without breaking existing stuff.

Then when you need to store them in an event handler:

```

GetState, Event.Top, Info
*Info.A = array_with_Adata
*Info.B = array_with_Bdata
SetState, Event.Top, Info

```

and also get them both later on:

```

GetState, Event.Top, Info
X = *Info.A
Y = *Info.B

```

This type of thing is covered frequently in both Liam's and David's books - as well as numerous examples on David's websites. FWIW my methodology is shamelessly nicked (with some minor alterations) from Liam's book section 9.5 "A GUI Application".

cheers,

paulv

---