Subject: Re: Passing info and destroying widgets...

Posted by Struan Gray on Mon, 21 Jun 1999 07:00:00 GMT

View Forum Message <> Reply to Message

mirko_vukovic@notes.mrc.sony.com writes:

```
>
>   I wrote:
>>
>>      I'm currently playing with a refinement where all my widgets
>>  become objects, and I can invoke certain methods from dying child
>>  widgets, thus bypassing the event queue should I want to force a
>>  particular excecution order.  Looks cool.
>>
>
>  Cool?  Cool? you say???  It seems absolutely essential! Splendid idea!
>
>  (Cooling it down some), widgets are objects after all.  Please,
>  keep us posted.
```

The idea of a objectified widget I owe to Mark Rivers. Deja News has a thead with a neat discussion of his technique, plus a few refinements - search on his name and 'objects'. My widgets follow his scheme, with a few inherited properties that I like all my program-oriented objects have (such as a unified way of handling global and user preferences) and generalised information sharing/passing methods (the above, plus the ability to handle conventional events).

At present the parts work, but the whole looks like it's in the middle of open heart surgery. I'm building a disperse set of data-objects, widget-objects and plot/analysis-objects and at present I'm playing around with different ways of distributing basic behaviours among them. I'm not sure when it will be ready for public consumption, but I promise to make what I have freely available when it is.

Struan

Subject: Re: Passing info and destroying widgets...

Posted by davidf on Mon, 21 Jun 1999 07:00:00 GMT

View Forum Message <> Reply to Message

Liam Gumley (Liam.Gumley@ssec.wisc.edu) gives us an
example of a program that can record the last instance
of a button push in a non-blocking, non-modal widget
when he writes:

> "Robert S. Mallozzi" wrote:
>> I believe you must use XMANAGER in blocking mode for
>> this technique to work.
>
> Here's an example which works in non-blocking mode:

No question it works. But I would argue that it works
for all the wrong reasons and is a *terrible* programming
practice in almost every instance. I mean, you can write
an object method that returns a data pointer too, but
by doing so you violate every tenet of good object programming
practice, in which the data should be encapsulated and
unseen by the outside world. Sucking the pointer out of
a widget program, except perhaps in the hands of just the
best programmers, is a practice that is guaranteed, it
seems to me, to get most of the rest of us in a hell of
a lot of trouble.

If you are going to recommend this, at the very least
teach people how to use HEAP_GC at the same time because
I'll bet a ton of money there will be leaking memory
right and left!

As for me, I'm sticking to widget programs that clean
themselves up and don't leave the user holding the bag,
er, pointer. :-)

Cheers,

David

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

Subject: Re: Passing info and destroying widgets...
Posted by davidf on Mon, 21 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Robert King (kingrj2@bp.com) writes:

> I eventually created a separate event handler to destroy the widgets, eg
>

> Pro kill_widgets, event
>   Widget_Control, event.top,/DESTROY
> END
>
> This worked with no errors!
>
> I'd like to know if there is another way around this problem as it seems
> rather strange behavoir.

"Strange" is not the word that comes to my mind when something
works with no errors. I would think it is "strange" to destroy
the top-level base and then imagine you could stick something
in its non-existent user value. But, then, that's just me. :-)

I used to write elaborate work arounds for this problem,
sometimes using WIDGET_INFO to make sure the top-level base
is still living before I stuffed something into it:

```
   IF WIDGET_INFO(event.top, /Valid_ID) THEN $
      WIDGET_CONTROL, event.top, Set_UValue=info, /No_Copy
```

But now I *always* have a separate event handler for the
QUIT button that does nothing but destroy the top-level
base. As Struan notes, all your clean-up should be done
in a CLEANUP routine. Doing it anywhere else means you
are going to miss it at least half the time.

I like this object-like programming approach. It's simple,
it's elegant, and it works like a charm. :-)

Cheers,

David

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

Subject: Re: Passing info and destroying widgets...
Posted by Liam Gumley on Mon, 21 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

"Robert S. Mallozzi" wrote:
> I believe you must use XMANAGER in blocking mode for

> this technique to work.

Here's an example which works in non-blocking mode:

```
;---cut here---
PRO TEST_EVENT, EVENT

;- Get pointer from top level base, then the info structure

widget_control, event.top, get_uvalue=ptr
info = *ptr

;- Handle the widget which caused this event

widget_control, event.id, get_uvalue=name
case 1 of
  name eq 'Button 1' or name eq 'Button 2' : info.name = name
  else : widget_control, event.top, /destroy
endcase

;- Update the info structure

*ptr = info

END

 ;---------------------------------------------------------- -----

FUNCTION TEST, PTR

;- Create widgets

tlb = widget_base(/column)
but1 = widget_button(tlb, value='Button 1', uvalue='Button 1')
but2 = widget_button(tlb, value='Button 2', uvalue='Button 2')
but3 = widget_button(tlb, value='Done', uvalue='Done')
widget_control, tlb, /realize

;- Create info structure, and store pointer in top level base

info = {name:''}
ptr = ptr_new(info)
widget_control, tlb, set_uvalue=ptr

;- Manage events

xmanager, 'test', tlb, /no_block
```

```
;- Return pointer to caller

return, ptr

END
;---cut here---
```

Multiple instances can be invoked, e.g.

```
ptr1 = test()
ptr2 = test()
ptr3 = test()
```

Then to find the the last selected button of any of the instances of the
dialog,

```
info = *ptr1
print, info.name
```

--
Liam E. Gumley
Space Science and Engineering Center, UW-Madison
http://cimss.ssec.wisc.edu/~gumley

Subject: Re: Passing info and destroying widgets...
Posted by Liam Gumley on Mon, 21 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

"Robert S. Mallozzi" wrote:
> I believe you must use XMANAGER in blocking mode for
> this technique to work.

I've used this technique successfully in blocking and non-blocking
modes. As long as the main widget procedure creates a new pointer for
each invocation, there is no problem.

--
Liam E. Gumley
Space Science and Engineering Center, UW-Madison
http://cimss.ssec.wisc.edu/~gumley

Subject: Re: Passing info and destroying widgets...
Posted by mallors on Mon, 21 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

In article <376E5BF5.622A1540@ssec.wisc.edu>,
 Liam Gumley <Liam.Gumley@ssec.wisc.edu> writes:
> Dirk Fabian wrote:
>> Here's something i can't figure out... I'm following the advice of DF and
>> communicating between my widgets with info structures. All is well, but now i
>> want to pass the info structure from the event handler back to the parent widget
>> with
>>
>> WIDGET_CONTROL, event.top, SET_UVALUE=lines, /NO_COPY
>>
>> and then destroy the widget. But you can't do this, because WIDGET_CONTROL (i
>> think) dereferences event.top so that
>>
>> WIDGET_CONTROL, event.top, /DESTROY
>>
>> fails since it doesn't know where to look. (you can't even put in a dummy to hold
>> the event.top number, the widget itself is gone from that id)
>>
>> Unfortunately, you can't /DESTROY the top widget first and expect to set it's
>> UVALUE later, either. So what do i do here? I tried putting a flag in my
>> info structure to trigger the base widget destruction back in the widget
>> definition level (not in the event handler), but i can't figure out when the
>> program would be able to look at that newly inserted flag.
>
> Dirk,
>
> If I understand your question correctly, you are trying to figure out
> how to pass a value from a widget event manager back to the calling
> program (i.e. the one that invoked XMANAGER) after the top level widget
> has been destroyed. The answer in IDL5 is pointers.
>
> In a widget which does not need to return any information, you store the
> info structure in the top level base user value, e.g.
>
> ;- Create widgets...
>
> ;- Create info structure
> info = {name:'test', value:indgen(10)}
>
> ;- Store info structure in top level base
> widget_control, tlb, set_uvalue=info
>
> ;- Start XMANAGER...
>
> However when the event manager must pass back information to the program
> which invoked XMANAGER, use a pointer to store the info structure, and
> store the *pointer* in the top level base user value, e.g.
>

```
>  ;- Create widgets...
>
>  ;- Create info structure and store via pointer
>  info = {name:'test', value:indgen(10)}
>  ptr = ptr_new(/allocate_heap)
>  *ptr = info
>
>  ;- Store pointer in top level base
>  widget_control, tlb, set_uvalue=ptr
>
>  ;- Start XMANAGER...
>
>  and then in the event manager, get the contents of the info structure
>  from the pointer, e.g.
>
>  ;- Get pointer
>  widget_control, event.top, get_uvalue=ptr
>
>  ;- Get info structure
>  info = *ptr
>
>  When the top level base is destroyed, the *pointer* still exists, thus
>  in the calling program you can retrieve it's value.
```

I believe you must use XMANAGER in blocking mode for
this technique to work.

Regards,

-bob

```
--
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ~~~~~~~~~
Robert S. Mallozzi                            256-544-0887
                                   Mail Code SD 50
Work: http://gammaray.msfc.nasa.gov/     Marshall Space Flight Center
Play: http://cspar.uah.edu/~mallozzir/          Huntsville, AL 35812
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ~~~~~~~~~
```

Subject: Re: Passing info and destroying widgets...
Posted by mirko_vukovic on Mon, 21 Jun 1999 07:00:00 GMT

View Forum Message <> Reply to Message

In article <7klp10$o6t$1@news.lth.se>,
Struan Gray <struan.gray@sljus.lu.se> wrote:

>
> I'm currently playing with a refinement where all my widgets
> become objects, and I can invoke certain methods from dying child
> widgets, thus bypassing the event queue should I want to force a
> particular excecution order. Looks cool.
>

Cool? Cool? you say??? It seems absolutely essential! Splendid idea!

(Cooling it down some), widgets are objects after all. Please, keep us posted.

Mirko

Sent via Deja.com http://www.deja.com/
Share what you know. Learn what you don't.

Subject: Re: Passing info and destroying widgets...
Posted by Struan Gray on Mon, 21 Jun 1999 07:00:00 GMT

View Forum Message <> Reply to Message

Liam.Gumley@ssec.wisc.edu writes:

> If I understand your question correctly, you are trying
> to figure out how to pass a value from a widget event
> manager back to the calling program (i.e. the one that
> invoked XMANAGER) after the top level widget has been
> destroyed. The answer in IDL5 is pointers.

Or objects. Or handles (well *I* like 'em).

I have a number of browser widgets dedicated to a particular filetypes which I call from other widgets and from the command line. When they die, their cleanup routine looks to see if the top level base had a group leader. If it did, and information needs to be passed upwards, the widget sends a custom event to the group leader which contains the relevant data. If not, the dying widget pops up a dialog asking if the user wants to keep the data in memory, and if the answer is yes, prints out the object/handle/pointer id by which the data can be globally accessed.

I'm currently playing with a refinement where all my widgets become objects, and I can invoke certain methods from dying child widgets, thus bypassing the event queue should I want to force a particular excecution order. Looks cool.

As a rule I like to put all such code in a cleanup routine rather than in the event handler. I've been caught out too many times by the multitude of ways that widgets can be killed other than through an author-defined 'quit' button - for example, via the window manager, from XTOOL and when a group-leader dies.

Struan

Subject: Re: Passing info and destroying widgets...
Posted by Liam Gumley on Mon, 21 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Dirk Fabian wrote:
> Here's something i can't figure out... I'm following the advice of DF and
> communicating between my widgets with info structures. All is well, but now i
> want to pass the info structure from the event handler back to the parent widget
> with
>
> WIDGET_CONTROL, event.top, SET_UVALUE=lines, /NO_COPY
>
> and then destroy the widget. But you can't do this, because WIDGET_CONTROL (i
> think) dereferences event.top so that
>
> WIDGET_CONTROL, event.top, /DESTROY
>
> fails since it doesn't know where to look. (you can't even put in a dummy to hold
> the event.top number, the widget itself is gone from that id)
>
> Unfortunately, you can't /DESTROY the top widget first and expect to set it's
> UVALUE later, either. So what do i do here? I tried putting a flag in my
> info structure to trigger the base widget destruction back in the widget
> definition level (not in the event handler), but i can't figure out when the
> program would be able to look at that newly inserted flag.

Dirk,

If I understand your question correctly, you are trying to figure out how to pass a value from a widget event manager back to the calling program (i.e. the one that invoked XMANAGER) after the top level widget has been destroyed. The answer in IDL5 is pointers.

In a widget which does not need to return any information, you store the
info structure in the top level base user value, e.g.

```
;- Create widgets...

;- Create info structure
info = {name:'test', value:indgen(10)}

;- Store info structure in top level base
widget_control, tlb, set_uvalue=info

;- Start XMANAGER...
```

However when the event manager must pass back information to the program
which invoked XMANAGER, use a pointer to store the info structure, and
store the *pointer* in the top level base user value, e.g.

```
;- Create widgets...

;- Create info structure and store via pointer
info = {name:'test', value:indgen(10)}
ptr = ptr_new(/allocate_heap)
*ptr = info

;- Store pointer in top level base
widget_control, tlb, set_uvalue=ptr

;- Start XMANAGER...
```

and then in the event manager, get the contents of the info structure
from the pointer, e.g.

```
;- Get pointer
widget_control, event.top, get_uvalue=ptr

;- Get info structure
info = *ptr
```

When the top level base is destroyed, the *pointer* still exists, thus
in the calling program you can retrieve it's value.

Cheers,
Liam.

--
Liam E. Gumley
Space Science and Engineering Center, UW-Madison
http://cimss.ssec.wisc.edu/~gumley

Subject: Re: Passing info and destroying widgets...
Posted by Robert King on Mon, 21 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Martin,

Thanks for the help, it's obvious now! I was thinking in a short-sighted
way - although the widgets don't exist after that point the program does
continue of course..

Regards,
Robert

> seems to me as if the problem with your routine lies in the statement
> AFTER the case construct: there you make another reference to event.top
> which no longer exists. What you could do is:
> [...]

Subject: Re: Passing info and destroying widgets...
Posted by Martin Schultz on Mon, 21 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Robert King wrote:
```
>
> Hi Dirk,
>
> I think that I came across this exact same problem only yesterday! I was
> using one event handler to manage all my events including exit/quit events
> where the widget hierarchy is destroyed, i.e. something like
>
> ;---------------------------------------------
>
> Pro my_event_handler, event
>   Widget_Control, event.top, GET_UVALUE=info, /NO_COPY
>   Widget_Control, event.id , GET_UVALUE=button
>
>   CASE button OF
>     'Open' :BEGIN
>               ....
>             END
>     'Save' :BEGIN
>               ....
>             END
>     'Quit' :BEGIN
>                SET_UVALUE=info,/NO_COPY
>               Widget_Control, event.top,/DESTROY
>             END
>     ENDCASE
>     Widget_Control, event.top, SET_UVALUE=info, /NO_COPY
>  END
>
>  [...]
```

seems to me as if the problem with your routine lies in the statement AFTER the case construct: there you make another reference to event.top which no longer exists. What you could do is:

```
    WIDGET_ACTIVE=1
    CASE button OF
      ...
    'Quit' :BEGIN
          Widget_Control, event.top, /DESTROY
          WIDGET_ACTIVE=0
          END
    ENDCASE

    IF (WIDGET_ACTIVE) THEN $
       Widget_Control, event.top, SET_UVALUE=info, /NO_COPY
```

It just doesn't make sense to set a UVALUE in a widget that no longer exists.

Regards,
Martin

```
--

 |||||||||||||||\\\\\\\\\\\\\-------------------//////////////// //||||||||||||||||
```

Martin Schultz, DEAS, Harvard University, 29 Oxford St., Pierce 109,
Cambridge, MA 02138   phone (617) 496 8318   fax (617) 495 4551
e-mail mgs@io.harvard.edu   web http://www-as/people/staff/mgs/

Subject: Re: Passing info and destroying widgets...
Posted by Robert King on Mon, 21 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Hi Dirk,

I think that I came across this exact same problem only yesterday! I was using one event handler to manage all my events including exit/quit events where the widget hierarchy is destroyed, i.e. something like

```
;---------------------------------------------

Pro my_event_handler, event
  Widget_Control, event.top, GET_UVALUE=info, /NO_COPY
  Widget_Control, event.id , GET_UVALUE=button

  CASE button OF
    'Open' :BEGIN
            ....
          END
    'Save' :BEGIN
            ....
          END
    'Quit' :BEGIN
            Widget_Control, event.top, SET_UVALUE=info,/NO_COPY
            Widget_Control, event.top,/DESTROY
          END
   ENDCASE
   Widget_Control, event.top, SET_UVALUE=info, /NO_COPY
END
```

I eventually created a separate event handler to destroy the widgets, eg

```
Pro kill_widgets, event
  Widget_Control, event.top,/DESTROY
END
```

This worked with no errors!

I'd like to know if there is another way around this problem as it seems
rather strange behavoir.

Regards,
Robert

Dirk Fabian <dirk@uwast.astro.wisc.edu> wrote in article
<7kkn9k$sag$1@news.doit.wisc.edu>...
>
> Here's something i can't figure out... I'm following the advice of DF and
> communicating between my widgets with info structures. All is well, but
now i
> want to pass the info structure from the event handler back to the parent
widget
> with

>
> WIDGET_CONTROL, event.top, SET_UVALUE=lines, /NO_COPY
>
> and then destroy the widget. But you can't do this, because
WIDGET_CONTROL (i
> think) dereferences event.top so that
>
> WIDGET_CONTROL, event.top, /DESTROY
>
> fails since it doesn't know where to look. (you can't even put in a
dummy to hold
> the event.top number, the widget itself is gone from that id)
>
> Unfortunately, you can't /DESTROY the top widget first and expect to set
it's
> UVALUE later, either. So what do i do here? I tried putting a flag in
my
> info structure to trigger the base widget destruction back in the widget
> definition level (not in the event handler), but i can't figure out when
the
> program would be able to look at that newly inserted flag.
>
> Thanks for your help. - Dirk
>
>
>