Subject: IDL 5.0 widgets...
Posted by Stein Vidar Hagfors H on Fri, 13 Jun 1997 07:00:00 GMT
View Forum Message <> Reply to Message

Aaaahhh, once more this nice exercise (NOT!) of finding out
which widget programs need modifications to *keep* working
whenever RSI modifies their widget stuff.....

I've managed to make a pretty nice demonstration program that
illustrates the "growing widget" problem, and two related (I think)
completely weird phenomena.

The program below resembles in structure a pretty complex
program of mine. Basically, the program let's the user modify
a structure containing any number of elements (structure tags
containing various information), adding new elements and modifying
thir contents. The structure is displayed inside a scrolling
widget base, with the contents of each element inside it's own
base. Whenever a new element is added, the base containing the
list of elements is destroyed and a new list is generated.

Now, in IDL v 4.0.1, I had a lot of problems with Xlib errors
along the line of "Xlib sequence lost" that would freeze up the
display (killing IDL from another terminal seemed to be the
only cure at times).

I traced this back to the use of WIDGET_CONTROL,..,UPDATE=0/1 to
speed up the widget updates during the element list rebuild.
I managed (in a nonsensical way) to work around these errors by
adding and removing (almost at random in the end) UPDATE=0/1
statements (and a blank PRINT statement!) at various levels in the
rebuilding process - assuming that it was some Xlib buffer that
got full when updating too many widgets at once (or something
like that).

But now, what happens under version 5is that when I add an
element and rebuild the widget list, the scrollable widget base
seems to remember its appearance when created the first time,
blanking out all the widgets that extend outside the limits
of the original contents... and again I can trace it back to
the use of UPDATE=0/1 statements at various places....

The demonstration program starts off with a single element,
using no WIDGET_CONTROL,..,UPDATE=0/1 statements. Adding
elements (press "Add element & rebuild") *works*, but it's a
good demonstration of the ever growing widget sizes whenever
UPDATE is on during widget changes.

Now, the UPDATE=0/1 mechanism can be turned on at various levels:

The EVENT level will turn update on/off on the scrollable
widget base itself before and after an add & rebuild is done,
or whenever a Modify operation is done.

The LIST level will turn update on/off on the base containing
all the elements as children (i.e., the list base, which is
the direct child of the scrollable widget base), before and
after all element bases are created, or before and after all
elements are modified (during a Modify operation).

The ELEMENT level will turn update on/off on the new element
bases (which contain just one label in this example) before
and after the label (the element "contents") is created, or
just before/after the labels are modified.

Now, if you use UPDATE at the EVENT level OR at the LIST
level only, it's ok, except if I do this with my original,
more complex program under IDL 4.0.1, I get Xlib errors more
often....

If you use UPDATE at the ELEMENT level ONLY, the Add operation
works (though *slower* and more flickering), but the Modify
operation blanks out everything inside the scrollable widget base
except the space used by its child during its original
creation (during /realization, I guess..)

If you use UPDATE at the EVENT level AND at the LIST level,
both operations blank out everything but the space used
by the first child (after first showing the updated contents
in a flicker).

Likewise (with small modifications in the flicker etc) with
any combination of two or more updates used.

As an added bonus: Try adding & rebuilding a couple of times
with no use of the UPDATE on/off on any level. Then, flick the
UPDATE status of the TOP level base on and off. Voila, the
ever-growing widget problem fixed. BUT!! Do the same once more,
and the darn scrollbase blanks out all but the initial section
again...argh!!

My program has to be able to run under v 4.0.1 (actually, under
3.6.1c as well :-(so I'm reluctant about fiddling around any
more with the number of UPDATES (due to the Xlib sequence lost
problem), but I guess.... well... I already have lines checking
for v 4.0.1, so I could check for 5.0 as well... or something..

Oh, and I'm doing this on

!version = { alpha OSF unix 5.0 Apr 28 1997}
widget_info(/version) = { Motif OSF 1.2}

David, do you think you could forward this to RSI as a bug
report, so I can save myself the hassle of going through
our system administrator etc...?

Anyway, the demo program is included below.

Regards,

Stein Vidar

```
; Make an "element" on "base", with or without update on/off statements
;
PRO witest_make_element,base,update=update
  mybase = widget_base(base,/column,frame=4)

  IF keyword_set(update) THEN widget_control,mybase,update=0
  label = widget_label(mybase,value = 'adsfadfadfadf',dynamic_resize=1)
  IF keyword_set(update) THEN widget_control,mybase,update=1
END


; Make a list of N elements on the scrollable base "onbase", with
; or without using update on/off on the container base (called mybase
here)
;
FUNCTION witest_makelist,onbase,n,update1=update1,update2=update2
  mybase = widget_base(onbase,/column)

  IF keyword_set(update1) THEN widget_control,mybase,update=0
  FOR i = 0,n-1 DO BEGIN
    witest_make_element,mybase,update=update2
  END
  IF keyword_set(update1) THEN widget_control,mybase,update=1

  return,mybase
END

; Modify element with base id "ebase", with or without using update
on/off
;
PRO witest_modify_element,ebase,update=update
```

```
  IF keyword_set(update) THEN widget_control,ebase,update=0

  labelid = widget_info(ebase,/child)
  widget_control,labelid,get_value=val
  widget_control,labelid,set_value=val+'xx'

  IF keyword_set(update) THEN widget_control,ebase,update=1
END

; Modify all elements on the list base "listid", with or without
; using the modify on/off on the list base
;
PRO  witest_modify_list,listid,no,update1=update1,update2=update2

  IF keyword_set(update1) THEN widget_control,listid,update=0

  element_base = widget_info(listid,/child)
  FOR i = 0,no-1 DO BEGIN
    witest_modify_element,element_base,update=update2
    element_base = widget_info(element_base,/sibling)
  END

  IF keyword_set(update1) THEN widget_control,listid,update=1
END



;
; Event function for the widget test program
;
PRO witest_event,ev

  widget_control,ev.id,get_uvalue=uval
  widget_control,ev.top,get_uvalue=status

  CASE uval OF
  "EXIT":BEGIN
    widget_control,ev.top,/destroy
    return
    ENDCASE

  "ADD":BEGIN
    status.count = status.count + 1
    widget_control,status.listid,/destroy
    IF status.update0 THEN widget_control,status.listbase,update=0
    status.listid = witest_makelist(status.listbase,status.count,$
                                    update1=status.update1,$
                                    update2=status.update2)
    IF status.update0 THEN widget_control,status.listbase,update=1
```

```
    ENDCASE

  "MODIFY":BEGIN
    IF status.update0 THEN widget_control,status.listbase,update=0
    witest_modify_list,status.listid,status.count,$
      update1=status.update1,update2=status.update2
    IF status.update0 THEN widget_control,status.listbase,update=1
    ENDCASE

  "UPDATE0":BEGIN
    status.update0 = ev.select
    ENDCASE

  "UPDATE1":BEGIN
    status.update1 = ev.select
    ENDCASE

  "UPDATE2":BEGIN
    status.update2 = ev.select
    ENDCASE

  "TOPDATE":BEGIN
    widget_control,ev.top,update=ev.select
    ENDCASE

  END

  widget_control,ev.top,set_uvalue=status

END


PRO witest

  top = widget_base(/column)

  label = widget_label(top,value='adfadfadf')

  button = widget_button(top,value='Exit',uvalue='EXIT')
  button2 = widget_button(top,value='Add element &
rebuild',uvalue='ADD')
  button3 = widget_button(top,value='Modify',uvalue='MODIFY')

  nonex = widget_base(top,/nonexclusive,/column)

  update0 = widget_button(nonex,value='Use UPDATE at EVENT level',$
                          uvalue='UPDATE0')
```

```
  update1 = widget_button(nonex,value='Use UPDATE at LIST level',$
                    uvalue='UPDATE1')

  update2 = widget_button(nonex,value='Use UPDATE at ELEMENT level',$
                    uvalue='UPDATE2')

  nonex2 = widget_base(top,/nonexclusive,/column,frame=2)

  update = widget_button(nonex2,value='UPDATE status for TOP base',$
                   uvalue='TOPDATE')
  widget_control,update,set_button=1

  listbase = widget_base(top,y_scroll_size = 400,x_scroll_size=400)

  listid = witest_makelist(listbase,1)

  status = {count:1,$
          update0:0,$
          update1:0,$
          update2:0,$
          listid:listid,$
          listbase:listbase}

  widget_control,top,/realize

  widget_control,top,set_uvalue=status

  xmanager,'witest',top
END
```