Subject: Programmatic size adjustment of widgets under Windows vs. UNIX (was "Natural size of explicitly-sized widget")
Posted by Jean-Paul Davis on Wed, 29 Aug 2007 22:12:18 GMT
View Forum Message <> Reply to Message

The reason to start adjusting widget sizes is that, with more than a
single row or column, each containing a different number of elements,
the automatic layout starts to look pretty ugly, especially if you use
frames to visually group elements. Then add to this the fact that I'm
changing the number of elements inside some groups in response to user
interaction. Despite calling explicit sizing a bad decision, David
even admits to doing something like this in this post:
 http://groups.google.com/group/comp.lang.idl-pvwave/msg/a465 23cdb1ad08a7.

Okay, so there is no way to automatically obtain a "natural size"
property for a widget. I got everything to look nice under both
Windows and UNIX by careful programming to address the peculiarities.
Figuring it out was akin to pulling my teeth out, only it took longer.

Has anyone compiled a full, detailed description of widget geometry
behavior for both systems? I will share here what I needed to figure
out in order to get my application to behave the way I wanted to, in
case someone else finds it useful. For Windows I'm using XP SP 2, and
for UNIX I'm using Red Hat Enterprise Linux 4 U5. Presumably the
following behavior will be the same under any UNIX-based system since
it depends on IDL's interaction with Motif? The '*' character below
refers to either x or y.

-- in all cases, the *size tag returned by widget_info does NOT
include the widget's margin (this may actually be stated in the
documentation somewhere)

-- under Windows, using widget_control to set ysize for a base defined
with /column, or to set xsize for a base defined with /row, instead
changes the scr_ysize or scr_xsize tag, respectively, so that it
corresponds to the desired *size. Effectively, ysize for column bases
and xsize for row bases automatically keep track of the base's natural
size, with the base's actual current size given by scr_*size, which
includes the margins

-- under UNIX, scr_*size is equal to *size for all widgets except the
TLB (which has window dressing) or widgets using character units
instead of pixels for *size; determining the natural size requires
adding up sizes of the individual elements

-- under UNIX, the first level of widget bases down from the TLB
always size themselves to the TLB; the value of the *size tag for such
a base does not change unless widget_control passes a value larger

than the *size of the widest/tallest child of TLB, or, if operating on
the widest/tallest child widget, larger than the *size of the next-
widest/tallest child widget

-- under UNIX, using widget_control to change the value of an
unrealized label widget will change that widget's *size accordingly;
thus, if you want a label widget to start blank, you have to define it
with text, get its size, change the text to null string, then
explicitly set it to the saved original size before realizing

-- under UNIX, using widget_control to set *size for an unrealized
widget actually sets the *size tag equal to the keyword value plus
twice the current *pad value; in other words, the *size value passed
to widget_control excludes the padding.

-- under UNIX, once a widget hierarchy is realized, the above behavior
changes to the expected behavior (*size value passed to widget_control
includes the padding), and (get this!) the first time *size is changed
for any widget, all other widgets that had *size specified before the
hierarchy was realized suddenly change their values of *size to equal
the values originally passed to them by widget_control (instead of
passed value + padding).

For a demonstration of this last (very strange) behavior, try the
following code:

```
wtest = widget_base(/row)
wcol1 = widget_base(wtest, /column)
wcol2 = widget_base(wtest, /column)
wsub11 = widget_base(wcol1, /column, /frame)
wsub12 = widget_base(wcol1, /column, /frame)
wsub21 = widget_base(wcol2, /column, /frame)
wsub22 = widget_base(wcol2, /column, /frame)
w111 = widget_text(wsub11, xsize=15)
w112 = widget_button(wsub11, value='TESTING')
w121 = widget_label(wsub12, value='TESTING')
w122 = widget_text(wsub12, xsize=15)
w211 = widget_button(wsub21, value='TESTING')
w212 = widget_button(wsub21, value='TESTING')
w221 = widget_text(wsub22, xsize=15)
w222 = widget_text(wsub22, xsize=15)

gsub11 = widget_info(wsub11, /geometry)
print, 'NATURAL PRE-REALIZED WSUB11: ysize = ', gsub11.ysize, 'ypad =
', $
    gsub11.ypad, 'margin = ', gsub11.margin,
format='(A0,I0,3X,A0,I0,3X,A0,I0)'
gsub21 = widget_info(wsub21, /geometry)
```

```
print, 'NATURAL PRE-REALIZED WSUB21: ysize = ', gsub21.ysize, 'ypad =
', $
      gsub21.ypad, 'margin = ', gsub21.margin,
format='(A0,I0,3X,A0,I0,3X,A0,I0)'

widget_control, wsub11, ysize = gsub11.ysize + 20
widget_control, wsub21, ysize = gsub21.ysize + 40

gsub11 = widget_info(wsub11, /geometry)
print, 'MODIFIED PRE-REALIZED WSUB11: ysize = ', gsub11.ysize, 'ypad =
', $
      gsub11.ypad, 'margin = ', gsub11.margin,
format='(A0,I0,3X,A0,I0,3X,A0,I0)'
gsub21 = widget_info(wsub21, /geometry)
print, 'MODIFIED PRE-REALIZED WSUB21: ysize = ', gsub21.ysize, 'ypad =
', $
      gsub21.ypad, 'margin = ', gsub21.margin,
format='(A0,I0,3X,A0,I0,3X,A0,I0)'

widget_control, wtest, /realize
widget_control, wsub11, ysize = gsub11.ysize - 10

gsub11 = widget_info(wsub11, /geometry)
print, 'REMODIFIED POST-REALIZED WSUB11: ysize = ', gsub11.ysize,
'ypad = ', $
      gsub11.ypad, 'margin = ', gsub11.margin,
format='(A0,I0,3X,A0,I0,3X,A0,I0)'
gsub21 = widget_info(wsub21, /geometry)
print, 'REMODIFIED POST-REALIZED WSUB21: ysize = ', gsub21.ysize,
'ypad = ', $
      gsub21.ypad, 'margin = ', gsub21.margin,
format='(A0,I0,3X,A0,I0,3X,A0,I0)'
```

On my Linux system this produces the following output:

```
NATURAL PRE-REALIZED WSUB11: ysize = 65   ypad = 3   margin = 2
NATURAL PRE-REALIZED WSUB21: ysize = 59   ypad = 3   margin = 2
MODIFIED PRE-REALIZED WSUB11: ysize = 91   ypad = 3   margin = 2
MODIFIED PRE-REALIZED WSUB21: ysize = 105   ypad = 3   margin = 2
REMODIFIED POST-REALIZED WSUB11: ysize = 81   ypad = 3   margin = 2
REMODIFIED POST-REALIZED WSUB21: ysize = 99   ypad = 3   margin = 2
```

Note how setting ysizes to 85 and 99 for the unrealized widgets
actually gave ysizes of 91 and 105, respectively (2*ypad was added to
ysize, which should normally include the padding). After realizing
the widgets, setting the ysize of wsub11 to 81 actually produces 81,
but also changes the ysize of wsub21 to 99. To verify that ypad is
the relevant parameter, change its value in the widget creation

calls. I would be curious to know if this behavior does NOT occur
under someone else's UNIX system.

For my application, I determined natural sizes by adding up components
(no references to scr_*size), and only had to include system-dependent
code (based on !VERSION.OS_FAMILY system variable) for sizing of non-
realized widgets.

Jean-Paul Davis