Subject: Re: Widget_base woes
Posted by David Fanning on Thu, 24 Nov 2005 15:28:36 GMT
View Forum Message <> Reply to Message

Peter Albert writes:

> The problem with widgets is that the ROW
> and COLUMN keywords just work weired. Setting ROW=n will create n rows
> in the end, however, subseqent elements are put in subsequent columns.
> Just an example:
>
> base=widget_base(row=3)
> a=widget_button(base, value="a")
> b=widget_button(base, value="b")
> c=widget_button(base, value="c")
> d=widget_button(base, value="d")
> widget_control, base, /real
>
> will create a widget which looks loke this:
>
> a b
> c
> d
>
> So we have three rows, as we defined, but not that the buttons were
> filled in row after row, that woul have been too easy ...

I agree that setting COLUMN and ROW keywords to something
other than 1 results in screwy results most of the time,
but I think the algorithm it uses for layout is a reasonable
algorithm. It seems to be something like this for a ROW base:

1. Calculate the X size of the widgets that are going to go
   into this base.
2. Maintaining their creation order (as I do everywhere else),
   put them in X rows in such a way that I get approximately
   X equal-length rows.

The problems comes about when you put things in the base that
do not have identical sizes. Since widgets are discrete things
and cannot be divided, the layout can be surprising. And it will
also be affected (usually in a negative way) by other things around
it.

In addition to sticking to COLUMN=1 and ROW=1 keywords, I sometimes
use the GRID keyword, too. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
