Subject: Re: Widget layout in BASE to get table in IDL 4.0
Posted by Josh Stillerman on Fri, 06 Oct 1995 07:00:00 GMT
View Forum Message <> Reply to Message

nmw@ion.le.ac.uk (Nigel Wade) wrote:
> I have an application which used to work in IDL 3, but no longer produces
> sensible output in IDL 4.0.
>
> What the routine is supposed to do is produce a table of widgets with the
> rows and columns nicely aligned. Using IDL 3 I could do this by creating
> a BASE widget and using the COLUMN attribute to tell it how many columns
> to use. Then I could fill the base with the widgets and the same number
> of widgets would be put into each column and each widget would be the same
> height, pretty much like a Motif RowColumn widget.
>
> With IDL 4 this is no longer the case. The widgets are all packed into
> the BASE widget and it doesn't even put the same number in each column.
> The result is a free-form complete mess. I have tried creating additional
> BASE, /COLUMN=1 widgets of the main BASE and then putting the table widgets
> in these. This does allow me to put the correct number of widgets in each
> column, but the widgets in each column are of different heights so they
> don't align across. I cannot use the XSIZE and YSIZE attributes because
> they are ignored for ROW/COLUMN BASE widgets.
>
> BTW, this is not regarded as a bug, but as a new *feature*. I have been
> told that this is the way it is now *meant* to work.
>
> Does anyone have any idea how it might be possible to create a table of
> widgets of different types which are aligned to a grid? I don't want to
> have to use a Bulletin Board type BASE and specify the XSIZE, YSIZE,
> XOFFSET, and YOFFSET of every child because a) it's very tedious and
> requires alteration every time a new row or column is added, b) it only
> works on one screen with one font - a different screen or font requires
> all the values to be changed.
>
> --
> Nigel Wade, System Administrator, Ionospheric Physics Group,
> University of Leicester, Leicester, LE1 7RH, UK
> E-mail : nmw@ion.le.ac.uk
> Phone : +44 (0)116 2523568, Fax : +44 (0)116 2523555
>
I made a compound widget (cw_base) which operates sort of like the bases in 3.x
To use it you need to do 2 things:

1) replace the widget_base with cw_base
2) disable updates when you are creating the base and its children.

Here is the code

```
>>>> >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> >>>>>
; Copyright (c) 1995, M.I.T.
;       Unauthorized reproduction prohibited.
;+
; NAME: CW_BASE
;
; PURPOSE: A base widget which lays itself out correctly in rows
;
; CATEGORY:
;       Compound widgets.
;
; CALLING SEQUENCE:
;       widget = CW_BASE([parent][,row = row] [,column=column] [any widget_base keywords])
;
; INPUTS:
;       PARENT - The ID of the parent widget.  if not present this is a top level base
;
; KEYWORD PARAMETERS:
;       row = row  - number of rows this base should have
;       column = column  - number of columns this base should have
;       any widget_base keywords -  _EXTRA=e
;
; OUTPUTS:
;       The ID of the created widget is returned.
;
; COMMON BLOCKS:
;       None.
;
; SIDE EFFECTS:
;
; PROCEDURE:
;
; MODIFICATION HISTORY:
;    8/16/95  JAS original version
;-
;**************************************************
; fixup a row base by makeing sure they all have
; the same width.  set it to the maximum width
;**************************************************
pro fix_rows, w
  width = 0
  c = widget_info(w, /child)
  while c ne 0 do begin
    geo = widget_info(c, /geometry)
    if (geo.scr_xsize gt width) then width = geo.scr_xsize
    c = widget_info(c, /sibling)
  endwhile
  c = widget_info(w, /child)
```

```
  while c ne 0 do begin
    widget_control, c, scr_xsize = width
    c = widget_info(c, /sibling)
  endwhile
end

;************************************************
; fixup a column base by makeing sure they all have
; the same height.  set it to the maximum height
;************************************************
pro fix_columns, w
  height = 0
  c = widget_info(w, /child)
  while c ne 0 do begin
    geo = widget_info(c, /geometry)
    if (geo.scr_ysize gt height) then height = geo.scr_ysize
    c = widget_info(c, /sibling)
  endwhile
  c = widget_info(w, /child)
  while c ne 0 do begin
    widget_control, c, scr_ysize = height
    c = widget_info(c, /sibling)
  endwhile
end
;************************************************************ *
;  create a base with one of the above procecdures as its
;  notify_realize
;************************************************************ *
function cw_base, parent, row=row, column=column, _extra=e
  psz = size(parent)
  rsz = size(row)
  if rsz(1) ne 0 then begin
    if (psz(1) ne 0) then $
      base = widget_base(parent, row=row, notify_realize='fix_rows', _extra=e) $
    else $
      base = widget_base(row=row, notify_realize='fix_rows', _extra=e)
  endif else begin
    csz = size(column)
    if csz(1) ne 0 then begin
      if (psz(1) ne 0) then $
        base = widget_base(parent, column=column, notify_realize='fix_columns', _extra=e) $
      else $
        base = widget_base( column=column, notify_realize='fix_columns', _extra=e)
    endif else begin
      if (psz(1) ne 0) then $
        base = widget_base(parent, _extra=e) $
      else $
        base = widget_base(_extra=e)
```

```
    endelse
  endelse
  return, base
end
```

>>>> >> >>>>>>
I hope this helps,
Josh Stillerman