Subject: Tips on Window Resize
Posted by jbob on Tue, 19 Nov 1996 08:00:00 GMT
View Forum Message <> Reply to Message

I regularly have a situation where I need to destroy and re-create a window, such as for resizing. The following is a summary of information about IDL window position and size as it pertains to window resizing.

The following information is either somewhere in the IDL documentation or is straightforward to figure out. However, I found that it was very time consuming to get all the details straight. I hope this note will save someone a bit of time.

Note: This summary may only apply to my specific setup: Sun SPARCstation running Solaris 2.5.1 and IDL Version 4.0.1b. It works using either OpenWindows 3.5.1 or Common Desktop Environment 1.0.2.

Tips on IDL Window Resize
11/19/96

My Situation

Most of my windows are laid out in rows (ie. the number of columns = 1). I need to "recommend" placement, since I have control panels that I want in one area of the screen and data display windows in another area. So most of my windows are created with:

```
main_base = WIDGET_BASE(TITLE="My Title", /TLB_SIZE_EVENTS, $
                  /COLUMN, XOFFSET=xoffset, YOFFSET=yoffset)
```

And most of the sub-bases are created with:

```
sub1_base = WIDGET_BASE(main_base)
sub2_base = WIDGET_BASE(main_base)
... etc. ...
```

The Coordinate System

Some of the IDL position information is with respect to the upper left corner as (0,0) and others with respect to the lower left corner as (0,0). I found the most useful information to be with respect to the upper left corner as (0,0), so I decided to forget about commands that use the lower left corner as (0,0), for example the DEVICE command. I think all of the information defaults to device coordinates, or in other words pixels.

Default Values

One of the problems I ran into was determining default values for the needed parameters.

The window frame created around each "main_base" has the following sizes:

```
win_frame_top  = 25 pixels
win_frame_edge =  5 pixels    ; All but top edge.
```

The documentation indicates there is a padding (xpad & ypad) between a widget and the edge of its base. By trial & error I found that:

```
Default xpad = Default ypad = 3 pixels
```

I found that all other applicable parameters, such as MARGIN, default to zero and are not needed for this note.

Re-Create Window at Same Position

To determine the current position, use WIDGET_INFO on the main_base with the GEOMETRY keyword:

```
main_geom = WIDGET_INFO(main_base, /GEOMETRY)
```

This does NOT include the frame, so that destroying and re-creating a window in the same location involves calculating xoffset and yoffset for the WIDGET_BASE statement (see example near the top of this note) for the "main_base" as:

```
xoffset = main_geom.xoffset - win_frame_edge
yoffset = main_geom.yoffset - win_frame_top
```

Re-Size Window and Sub-Base

Usually the size of one of my sub-bases can vary, for example a graphics draw area, and the size of the other sub-bases are fixed, like rows of buttons. So now we need to determine the size of the variable-sized graphics draw area.

For example, assume only one sub-base initially. Assume it is a graphics draw area of the form:

```
draw_base   = WIDGET_BASE(main_base)
draw_widget = WIDGET_DRAW(draw_base, XSIZE=xsize, YSIZE=ysize)
```

The resize EVENT structure contains the new main_base size and it does not include the frame. So xsize and ysize used in the above WIDGET_DRAW command are:

```
xsize = EVENT.X - (2 * xpad)
ysize = EVENT.Y - (2 * ypad)
```

If a second, fixed sub-base is added to the window, say as a row of buttons above the graphics draw area, then take the size of its base into effect. First determine its size with:

```
sub1_geom = WIDGET_INFO(sub1_base, /GEOMETRY)
```

Now the xsize and ysize equations used to size the graphics draw area become (again, this example has the sub-bases as rows in one column, so only the Y size of the button sub-base affects the graphics draw widget size):

```
xsize = EVENT.X - (2 * xpad)
ysize = EVENT.Y - (2 * ypad) - sub1_geom.scr_ysize
```

Test for Resize Event

One may need to test for the existence of a resize event, for example one may want to use the same event handler for more than one type of event. In this case, the !D system variable can be used for the "old" or previous size of the graphics draw area. Using "xsize" and "ysize" from above:

```
IF ((xsize NE !D.X_SIZE) OR (ysize NE !D.Y_SIZE)) THEN BEGIN
   ; Then a resize event occurred....
ENDIF
```

Please report errors or omissions to:

J. Bob Brown jbob@snap.med.ge.com
Consulting for GE Medical Systems
For ID only -- standard disclaimers apply.

"Of course that's just my opinion. I could be wrong."
-Dennis Miller
