Subject: Persistent Object Graphics Not Persistent in Resizeable Windows
Posted by davidf on Fri, 06 Jun 1997 07:00:00 GMT
View Forum Message <> Reply to Message

Folks,

I've run into a problem in which I lose all my axes labeling
if I resize my draw widget object window before evoking the
Draw method of the window. Here is a bug report and some
sample code to see the problem. (Those of you who want to
see how to add axes to a surface plot, take note. :-))

Cheers,

David

********** **********

Name: David Fanning
E-mail Address: davidf@dfanning.com
IDL version: { x86 Win32 Windows 5.0 Apr 28 1997}
Platform and OS: WindowsNT 4.0, Service Pack 1

Description of Problem: ----------

"Persistent" object graphics are not very persistent if
you resize the object window before you evoke the Draw
method. Here I am losing the axes labeling when the window
is resized.

Example Code: ----------

```
Pro XSurface_Cleanup, tlb

    ; Come here when program dies. Free all created objects.

Widget_Control, tlb, Get_UValue=info
Obj_Destroy, info.thisContainer
END
 ;---------- --------


PRO XSurface_Resize, event

    ; The only events generated by this simple program are resize
    ; events, which are handled here.

    ; Get the info structure.
```

```
Widget_Control, event.top, Get_UValue=info, /No_Copy

    ; Resize the draw widget.

 ;8888888888888888888888888888888888888888888888888888888888 888
; Here is the problem. Comment this line out and uncomment the
; two below it to see that the problem comes from resizing the
; window.

info.thisWindow->SetProperty, Dimension=

;info.thisWindow->erase, Color=
;Wait, 1
 ;8888888888888888888888888888888888888888888888888888888888 888

    ; Redisplay the graphic.

info.thisWindow->Draw, info.thisView

    ;Put the info structure back.

Widget_Control, event.top, Set_UValue=info, /No_Copy
END
 ;------------------------------------------------------------ --------


PRO XSurface, data

    ; Need some data.

Catch, error
IF error NE 0 THEN BEGIN  ; Can't find LoadData.
   data = DIST(40)
   GOTO, CreateView
ENDIF

IF N_Params() EQ 0 THEN   data = LoadData(2)

CreateView:
Catch, /Cancel

    ; Create a view. Use RGB color. Charcoal background.
    ; The viewplane rectangle extends from -1 to 1 in XY directions.

thisView = OBJ_NEW('IDLgrView', Color=, Viewplane_Rect=)

    ; Create a model and add it to the view.
```

```
thisModel = OBJ_NEW('IDLgrModel')
thisView->Add, thisModel

    ; Create a wire mesh surface. Make it yellow.

thisSurface = OBJ_NEW('IDLgrSurface', data, Color=)

    ; Create axes for the surface. Color them green.

xAxis = Obj_New("IDLgrAxis", 0, color=, Ticklen=0.1)
yAxis = Obj_New("IDLgrAxis", 1, color=, Ticklen=0.1)
zAxis = Obj_New("IDLgrAxis", 2, color=, Ticklen=0.1)

    ; Add the surface and axes to the model.

thisModel->Add, thisSurface
thisModel->Add, xAxis
thisModel->Add, yAxis
thisModel->Add, zAxis

    ; Get the data ranges for the surface.

thisSurface->GetProperty,XRange=xrange,YRange=yrange,ZRange=zrange

    ; Set the range and location of the axes. In this case,
    ; we are scaling the data into -0.5 to 0.5, so that even
    ; when the surface is rotated, it stays inside the -1 to 1
    ; viewing rectangle. Note that not all values in the Location
    ; keyword are used. (I've put really large values into the
    ; positions that are not being used to demonstate this.) For
    ; example, with the X axis only the Y and Z locations are used.

xAxis->SetProperty, Range=xrange, Location=
yAxis->SetProperty, Range=yrange, Location=
zAxis->SetProperty, Range=zrange, Location=

    ; Scale the surface and axes into the range -0.5 to 0.5.

xs = [-0.5, 1/(xrange-xrange)]
ys = [-0.5, 1/(yrange-yrange)]
zs = [-0.5, 1/(zrange-zrange)]
thisSurface->SetProperty,XCoord_Conv=xs, YCoord_Conv=ys, ZCoord_Conv=zs
xAxis->SetProperty, XCoord_Conv=xs
yAxis->SetProperty, YCoord_Conv=ys
zAxis->SetProperty, ZCoord_Conv=zs

    ; Rotate model to the standard view.
```

```
thisModel->Rotate,, -90  ; To get the Z-axis vertical.
thisModel->Rotate,,  30  ; Rotate it slightly to the right.
thisModel->Rotate,[1,0,0],  30  ; Rotate it down slightly.

    ;Create the widgets to view the surface.

tlb = Widget_Base(Title='Surface Example', Column=1, TLB_Size_Events=1)
drawID = Widget_Draw(tlb, XSize=300, YSize=300, Graphics_Level=2,
Retain=2)
Widget_Control, tlb, /Realize

    ; Get the window destination object.

Widget_Control, drawID, Get_Value=thisWindow

    ; Draw the view in the window.

thisWindow->Draw, thisView

   ; Create a container object to hold all the other
   ; objects. This will make it easy to free all the
   ; objects when we are finished with the program.

thisContainer = Obj_New('IDLgrContainer')

   ; Add the view to the container.

thisContainer->Add, thisView

   ; Create an INFO structure to hold needed program information.

info = { thisContainer:thisContainer, $    ; The object container.
        thisWindow:thisWindow, $          ; The destination window
object.
        thisView:thisView }               ; The view object.

   ; Store the info structure in the UValue of the TLB.

Widget_Control, tlb, Set_UValue=info, /No_Copy

   ; Call XManager. Set a cleanup routine so the objects
   ; can be freed upon exit from this program.

XManager, 'xsurface', tlb, Cleanup='XSurface_Cleanup', /No_Block, $
   Event_Handler='XSurface_Resize'
END
 ;------------------------------------------------------------ --------
```

Known Workarounds or Fixes: ----------

Don't resize windows or store all object components in the info
structure so the object can be re-built each time.

[RSI Technical Support Response: ----------

Pending.

--
David Fanning, Ph.D.
Fanning Software Consulting
Customizable IDL Programming Courses
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com
IDL 5 Reports: http://www.dfanning.com/documents/anomaly5.html