Subject: Re: Object Graphics Printing
Posted by Rick Towler on Wed, 23 Oct 2002 16:50:16 GMT
View Forum Message <> Reply to Message

"Ben Tupper" <btupper@bigelow.org> wrote
>
> I enjoy working with objects graphics when I can... until somebody
> wants a printout. Then I hate the dang stuff. It *always* seems
> like such a tinker-with-it-until-it-works experience. Hmph!
>
> I have been daydreaming (again) about a generic tool much like
> FSC_PSCONFIG, that is, an interactive tool for placing/sizing object
> graphics on a printed page.
>
> I gather from the object printing examples in 'What's New in IDL 5.5'
> the only things that need to be modified are
>
> (1) the LOCATION property of the VIEW(s) to be drawn
>
> and
>
> (2) the SCALE properites of any of the MODELS contained in the
> VIEW(s).
>
>
> It appears that one only needs to ascertain and save the initial
> properties of the view(s) and the models contained in the view(s).
> Then, relocate the views and rescale the models for the printer.
> Next, invoke the printer's draw method. Finally, restore the original
> properties of the view(s) and models. Piece of cake, maybe.
>
>
> I find the example a bit hard to fathom because information about the
> object graphics window is used to scale/locate the object graphics
> onto the printer page. It makes it seem complicated and a bit
> adhoc... precisely my own experience.
>
>
> My questions are...
>
> (1) is it possible to properly size/place object graphics on a printer
> page given *only* the VIEW(s) and the PRINTER object?

Yes.

Look in $RSI_DIR/Examples/Visual/Utility you'll find a program called
set_view.pro which will set your view accordingly based on the destination
device (printer or window) you provide. I haven't actually tried it (I just

found it a day or so ago) but it looks promising.

> (2) If it is possible to size/locate object graphics in a general way,
> can the concept be 'enlarged' to encompass any destination device
> (like the clipboard or buffer)?

Why is it whenever I think I am the first person to stumble upon some trick in IDL I learn that I am years behind...

While digging thru idl.dll the other day I found the methods and keywords for IDLgrGraphic which is the superclass of all graphic atoms. Three keywords for the GetProperty method were really interesting: XRANGE, YRANGE, ZRANGE. Since I had been mulling about just this very thing I was quite excited and wrote a nice little recursive function which returns the axis aligned bounding box of a model hierarchy. Exactly what you are looking for.

After doing this I stumbled upon the aforementioned set_view.pro and then get_bounds.pro (in the same directory). Get_bounds.pro performs a similar function although I haven't studied it. So much for originality. I still like my version better though :) You are welcomed to it, although I need to fix one feature.

You would still have to deal with scaling your models to fit the perspective of your printed page but you should have the information you need. I would probably create a second view specifically for printing and an additional model that I could add an alias of my main model tree to. That way you can scale that model without distorting your scene in other views.

Happy printing...

-Rick

Subject: Re: Object Graphics Printing
Posted by btupper on Thu, 24 Oct 2002 23:51:22 GMT
View Forum Message <> Reply to Message

On Wed, 23 Oct 2002 09:50:16 -0700, "Rick Towler"
<rtowler@u.washington.edu> wrote:

>>
>> My questions are...
>>

>> (1) is it possible to properly size/place object graphics on a printer
>> page given *only* the VIEW(s) and the PRINTER object?
>
> Yes.
>
> Look in $RSI_DIR/Examples/Visual/Utility you'll find a program called
> set_view.pro which will set your view accordingly based on the destination
> device (printer or window) you provide. I haven't actually tried it (I just
> found it a day or so ago) but it looks promising.
>
>
>> (2) If it is possible to size/locate object graphics in a general way,
>> can the concept be 'enlarged' to encompass any destination device
>> (like the clipboard or buffer)?
>
> Why is it whenever I think I am the first person to stumble upon some trick
> in IDL I learn that I am years behind...
>
> While digging thru idl.dll the other day I found the methods and keywords
> for IDLgrGraphic which is the superclass of all graphic atoms. Three
> keywords for the GetProperty method were really interesting: XRANGE, YRANGE,
> ZRANGE. Since I had been mulling about just this very thing I was quite
> excited and wrote a nice little recursive function which returns the axis
> aligned bounding box of a model hierarchy. Exactly what you are looking
> for.
>
> After doing this I stumbled upon the aforementioned set_view.pro and then
> get_bounds.pro (in the same directory). Get_bounds.pro performs a similar
> function although I haven't studied it. So much for originality. I still
> like my version better though :) You are welcomed to it, although I need to
> fix one feature.
>
>
> You would still have to deal with scaling your models to fit the perspective
> of your printed page but you should have the information you need. I would
> probably create a second view specifically for printing and an additional
> model that I could add an alias of my main model tree to. That way you can
> scale that model without distorting your scene in other views.

Thanks Rick,

Just the nudge I needed. I have used these (set_view and get_bound)
before in an generic axis object. My bride tells me that I am blessed
with a short-term memory - here's evidence.

I'm tickled to see that the model scaling is *not* fiddled with in
Set_View. Whew, I dread fiddling with the transform matrix. I think
your idea of creating a second view and model (aliased) just for the

printer is just the ticket. I wonder why the View container doesn't
accept aliased items like the model container can?

Another thing to consider is views bundled in ViewGroups and views
placed inside of scenes.

Now, if I could just figure out how David's neat little fsc_plotwindow
works!

Cheers,
Ben

Subject: Re: Object Graphics Printing
Posted by David Fanning on Fri, 25 Oct 2002 13:57:40 GMT
View Forum Message <> Reply to Message

Ben Tupper (btupper@bigelow.org) writes:

> Now, if I could just figure out how David's neat little fsc_plotwindow
> works!

Yeah, me too. :-(

Have you had a look at the FSC_SURFACE_PRINT module
in FSC_SURFACE? I want the printed version of the
window to have the same aspect ratio (I.e., look like)
the version on the display, so I muck around with the
printer aspect ratio, then set the view coordinates.

I'm not sure this is what you want, but it might
give you some ideas for your new BT_Config program. :-)

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Object Graphics Printing
Posted by btupper on Tue, 29 Oct 2002 02:14:56 GMT

On Fri, 25 Oct 2002 07:57:40 -0600, David Fanning <david@dfanning.com>
wrote:

> Ben Tupper (btupper@bigelow.org) writes:
>
>> Now, if I could just figure out how David's neat little fsc_plotwindow
>> works!
>
> Yeah, me too. :-(
>
It really is a whiz-banger. Soo many details.

> Have you had a look at the FSC_SURFACE_PRINT module
> in FSC_SURFACE? I want the printed version of the
> window to have the same aspect ratio (I.e., look like)
> the version on the display, so I muck around with the
> printer aspect ratio, then set the view coordinates.
>

Good idea. I have modified your handy ASPECT program slightly
(appended below) so that the arguments are the VIEW and the
DESTINATION objects. Margins can be specified in each direction as a
two element vector (perhaps that should be 4 elements... left, bottom,
right, top?)

The tricky part with this is dealing with the default VIEW dimensions.
In this case, the programmer has allowed the VIEW dimensions to fit
the destination device (usually a window). A call to the VIEW's
getproperty method returns [0,0] for the dimensions of the view.
Oops, what kind of aspect is that? There's nothing to be done in
that case, except match the VIEW's dimensions to the DESTINATION (in
my case the printer - which doesn't have the same aspect ratio as the
VIEW.)

This makes it tough to come up with a general solution to using just a
VIEW to properly scale to the printer, buffer, clipboard,... unless
there's another way around it that is right under my nose!?!

Thanks for the tip,

Ben

```
;+
; NAME:
; FITASPECT
;
; PURPOSE:
```

```
; This function returns the new dimensions
;   required to fit a IDLgrView into an
;    IDLgrDestination device. The aspect ratio
;   of the orginal view is preserved if the VIEW
; are explicitly set.
;
; CATEGORY:
; Object Graphics
;
; CALLING SEQUENCE:
; returned = FITASPECT(view, $
; dest, [margin = margin], [location = location])
;
; ARGUMENTS:
; VIEW Set this to a valid IDLgrVIEW or an
;     object that is superclassed by IDLgrVIEW.
;     If the dimensions of the VIEW are not
;   explicitly set, these are returned by
;   IDL as [0.0, 0.0] which is the flag indicating
;     'fit to destination'.  In this case
;   the VIEW ASPECT will be figured to
;   fit the destination (accounting for margins.)
; DEST Set this to a valid IDLGRSRCDESTINATION
;  object, such as a Printer, Clipboard, Window, etc.
;
; KEYWORDS:
; MARGIN  Set this equal to two element
;     array of the x and y margins
;     in normalized units.  If set to a scalar
;     this margin is used in both the x and y directions.
;   The default is [0.0, 0.0].
; LOCATION  Set this equal to named
;     varibale to retrieve the appropriate
;     location for the view with in the destination.
;    This is a two element array of [xloc, yloc]
; UNITS Set this equal to a named variable
;     to retrieve the units of the
; destination device.
;
; EXAMPLE (kind of...)
; Suppose you had an object graphics view set
;   just the way you like it in and IDLgrWINDOW...
;   how do you place it on the printer and have it come
; with the same apsect ration and fitting on the page?
;
;  ;preserve you orginal settings
;  view->GetProperty, Dim = oldDim, Loc = oldLoc, units = oldUnits
;   ;get the new dimensions and location
```

```
;  newdim = fit_aspect(view, printer, margin = [0.1, 0.1], loc =
NewLoc, Units = newUnits)
;   ;update the view
; view->Setproperty, loc = newloc, dim = newdim, units = newunits
;   ;print
; printer->Draw, view
; printer, newdocument
;  ;restore the views orginal settings
;  view->SetProperty, Dim = oldDim, Loc = oldLoc, units = oldUnits
;
;
; MODIFICATION HISTORY:
;  Adapted from David Fanning's FSC_SURFACE and
; ASPECT programs.     28 OCT 2002, BT
;-


FUNCTION  Fit_Aspect, view, dest, $
 margin = margin, Location = Location, $
 Units = Units

If n_elements(View) EQ 0 Then $
 Message, 'View is required'

If Obj_ISA(View, 'IDLGRVIEW') NE 1 Then $
 Message, 'View must be IDLGRVIEW or '+ $
   'superclassed by IDLGRVIEW'

If n_elements(Dest) EQ 0 Then $
 Message, 'Destination is required'

If Obj_ISA(Dest,  'IDLGRSRCDEST') NE 1 Then $
 Message, 'View must be a destination class object ' + $
  'or superclassed by a destination class object'

 ;get requested margin in normalized units
Case n_elements(Margin) of
 0: Marg = [0.0, 0.0]
 1: Marg = [Margin, Margin]
 Else: Marg = Margin[0:1]
EndCase

 ;force the destination into device units
Dest->GetProperty, units = Units
Dest->SetProperty, units = 0

 ;get the destination properties
Dest->GetProperty, Dimension = dDim
```

```
dAspect = Float(dDim[1])/dDim[0]
Dest->SetProperty, units = Units

 ;now get the units again (in the original setting)
Dest->GetProperty, Dimension = dDim


 ;get the view properties
View->GetProperty, Dimension = vDim
If TOTAL(vDim) EQ 0 Then Begin
 Message, 'Dimensions of VIEW not explicitly set... '+ $
  'setting aspect to destination', /Info
 vAspect = dAspect
EndIf Else vAspect = Float(vDim[1])/vDim[0]

 ;compare the two
Case 1 Of

vAspect LE dAspect:  Begin

  ;taller than wide
  x0 = Marg[0]
  y0 = 0.5 - (0.5 - marg[1] ) * (vAspect/dAspect)
  x1 = 1.0 - marg[0]
  y1 = 0.5 + (0.5 -  marg[1] ) * (vAspect/dAspect)

  NewDim = [x1-x0, y1-y0] * dDim

 End

vAspect GT  dAspect : Begin

  ;wider than tall

 x0 = 0.5 - (0.5 - marg[0]) * (dAspect/vAspect)
 y0 = marg[1]
 x1 = 0.5 + (0.5 - marg[0]) * (dAspect/vAspect)
 y1 = 1.0 - marg[1]

 NewDim = [x1 - x0, y1 - y0] * dDim

 End

Else: Begin


 newDim = min(Region)
 NewDim = [NewDim, NewDim]
```

```
 End

EndCase

Location = (dDim - NewDim)/2.0

Return, NewDim
END ;
```