Subject: automatic placement of legend in 2-D graph
Posted by Jean-Paul Davis on Mon, 21 Jan 2008 20:26:11 GMT
View Forum Message <> Reply to Message

I use a legend object in a separate model object to show a legend on a 2-D graph consisting of up to six plot objects. Currently I set the position of this legend once and leave it there. Of course, changes to the data or zoom scale can cause one or more of the plot object curves to run right through the legend.

Does anyone have ideas for an easy way to automatically determine the "best" placement of a legend? Using information on the axes ranges, legend object boundaries, and plot data themselves, I might be able to do some fancy calculations involving areas in a plane, probably limiting myself to a set number of standard positions and picking the least offensive. But this could be easier if there were a way to determine the intersections of various objects in a particular view. I'm interested in ideas related to either approach.

Jean-Paul

Subject: Re: automatic placement of legend in 2-D graph
Posted by Jean-Paul Davis on Thu, 24 Jan 2008 19:30:15 GMT
View Forum Message <> Reply to Message

Okay, so I came up with an idea on my own that should work, but it appears impossible to implement using the IDLgrLegend object. Yeah, I know the simplest answer is to write my own Legend object (or use someone else's), but that's probably not the simplest course of action for me.

Anyway, here's the problem. I cannot get the IDLgrLegend object to recompute its dimensions with changes in the containing view object's viewplane rectangle. The heirarchy is rooted at this view object, which contains two model objects, one for the plot and axes objects, the other for the legend (so I can translate the legend around separately from the plot and axes objects). The dimensions returned by IDLgrLegend::ComputeDimensions or IDLgrLegend::GetProperty(xrange, yrange) are correct for the viewplane rectangle set when the IDLgrView object is initialized, but never change after that, regardless of the setting of IDLgrLegend's Recompute property and regardless of how many times I redraw the window after subsequent changes to the viewplane rectangle. Here is an example to show this:

```
ofont = obj_new('IDLgrFont', 'Helvetica', size=16.0)
oxtitle = obj_new('IDLgrText', string='X-AXIS TITLE', recompute=2,
font=ofont)
```

```
oxaxis = obj_new('IDLgrAxis', direction=0, location=[9999,0,0.01],
title=oxtitle)
oxaxis -> getproperty, ticktext=oxtlab
oxtlab -> setproperty, , recompute=2, font=ofont
omodel = obj_new('IDLgrModel')
omodel -> add, oxaxis
olegend = obj_new('IDLgrLegend', item_linestyle=[0], item_name=['Test
Legend'], font=ofont, glyph_width=3.0, gap=0.3)
olegend -> setproperty, recompute=1 ; according to documentation, this
cannot be set by init
olegmod = obj_new('IDLgrModel')
olegmod -> add, olegend
oview = obj_new('IDLgrView', viewplane_rect=[0.0,0.0,1.0,1.0])
oview -> add, omodel
oview -> add, olegmod
owin = obj_new('IDLgrWindow', graphics_tree=oview)
owin -> draw
lsize = olegend -> computedimensions(owin)
print, lsize
oview -> setproperty, viewplane_rect=[-0.2,-0.2,1.4,1.4]
owin -> draw
lsize2 = olegend -> computedimensions(owin)
print, lsize2
```

If IDLgrLegend were recomputing it's dimensions, then I'd expect lsize2 to be different from lsize (the legend is "wider" in the normalized coordinates indicated by oxaxis object after increasing the viewplane rectangle). Has anyone seen this or have a work-around?

Jean-Paul

Subject: Re: automatic placement of legend in 2-D graph
Posted by Jean-Paul Davis on Fri, 25 Jan 2008 00:31:59 GMT
View Forum Message <> Reply to Message

Okay, one last contribution to a discussion with myself. Hey, if it saves someone else grief, then it's worthwhile.

I studied the IDL code for IDLgrLegend and found that the "recompute" property gets reset to zero at the end of each pass through the ComputeProperties method. It never seemed to work because recompute was always set back to zero by the time I tried to obtain the dimensions. The easy workaround is just to make sure you first set recompute=1 immediately prior to each use of the ComputeProperties method. Hence,

olegend -> setproperty, /recompute

lsize = olegend -> computedimensions(owin)

gives different values in lsize before and after changing the viewplane rectangle.

I also figured out that I do NOT need the extra model object in my previous post just to translate the legend separately... IDLgrLegend inherits its own copy of the Translate method from IDLgrModel, so I can translate just the legend by itself even if it's in the same model with other objects.

I also think I have an algorithm that will work for my original question, but I'm not sharing that unless someone is really, really interested (and unless I prove it to actually work).

Jean-Paul

Subject: Re: automatic placement of legend in 2-D graph
Posted by Jean-Paul Davis on Wed, 30 Jan 2008 22:44:05 GMT
View Forum Message <> Reply to Message

I thought I would share this in case someone finds it useful. After spending a shameful amount of time barking up wrong trees, I finally hit on a very simple solution to (semi-) automatic legend placement:

1. determine the legend's boundaries in device coordinates (i.e., pixels in the window)
2. hide the legend and redraw
3. get the window's image_data property
4. loop through possible legend positions (in device coordinates) to find one that has the fewest (preferably zero) non-background-color pixels in the region of the image defined by the legend boundaries
5. translate the legend accordingly, unhide it, and redraw

Jean-Paul

Subject: Re: automatic placement of legend in 2-D graph
Posted by David Fanning on Wed, 30 Jan 2008 22:49:52 GMT
View Forum Message <> Reply to Message

Jean-Paul Davis writes:

> I thought I would share this in case someone finds it useful. After
> spending a shameful amount of time barking up wrong trees, I finally
> hit on a very simple solution to (semi-) automatic legend placement:

>
> 1. determine the legend's boundaries in device coordinates (i.e.,
> pixels in the window)
> 2. hide the legend and redraw
> 3. get the window's image_data property
> 4. loop through possible legend positions (in device coordinates) to
> find one that has the fewest (preferably zero) non-background-color
> pixels in the region of the image defined by the legend boundaries
> 5. translate the legend accordingly, unhide it, and redraw

Now *that* is what I call a good algorithm. Simple enough even
I can understand it! :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming (www.dfanning.com)
Sepore ma de ni thui. ("Perhaps thou speakest truth.")
