Subject: Re: Scaling atoms & axes in object graphics
Posted by Martin Schultz on Wed, 16 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Mark,

IMHO, you should stick to approach A for the reasons that you indicate. I had
once started to lay out a similar class structure and came to the following hierarchy:

0. (top) plot_collection ("album")
1. plot page (1 page equivalent to one window or one page of printed output)
2. panel (1 graphical entity that shows related things)
3. graph (1 plot or image which may have several curves or axes, etc)
4. molecules ;-) (axes, curves, contours, vector fields - each related to one data variable)
5. atoms (plot symbols, text labels, etc.)

Within a graph (level 3), everything should be on a fix scale, i,e. using something similar to
Conv_Coord. However, "normal" coordinates for a graph would not be IDL's normal coordinates, but
0. and 1. would always correspond to the graph edges. The same holds for the panel, so that in order
to position a graph on a panel you use the panel's "normal" coordinates. And once more for the
page, only that here you may want to take care of the printable area or user defined page margins,
so that 0. to 1. refers to the portion of the page that will receive some ink or toner when
printed. For a screen window, on the other hand, you ignore the page margins, of course.

Now, since these are many levels in a hierarchy, you probably want to be able to skip some of them,
similar to the IDLgr hierarchy of models and scenes. The minimum you would need to display data is
a graph, and if displayed or printed, it would fill the window or page. Then, to combine several
graphs in one figure for a publication (the typical (a), (b), (c), ...), you put the graphs into
a panel, and add some labels and perhaps even a figure caption. Now you want to produce a poster
or an overhead with two "figures" (or more, of course), then you combine

several panels onto a
page. If each panel would only contain one graph, you could also store
the graphs directly on
the page - all that matters is that both object types have a show method
(and add, remove, ...).
At the very last, you can combine several pages in a "photo-album" to
produce for example a
postscript file.

Then, if you really want to get fancy, you can play games: while a graph
will generally be parallel
to the panel edges, you could freely rotate panels on the page (perhaps
a solution to the postscript
upside down problem discussed a few days ago?). And by changing the
panel or page size you have an
easy way of scaling the whole thing.

Doesn't this sound nice? The only drawback is that I will probably never
have time to implement it.
I had started and got into details which led eventually to "variable"
objects which are still at a
rather young development stage, but I am already using them for "file"
objects which are still at a
rather young development, but I am using them for a "model evaluation"
object which is still ...

But I promise to take a look at your objects before I start doing
anything in this respect myself
(will be rather soon).

Cheers,
Martin

Mark Hadfield wrote:
>
> In the last few days I have been reconsidering my approach to building up
> scientific graphs in object graphics. By "scientific graphs" I mean a wide
> class of graphs in which data are represented geometrically in association
> with axes in 1, 2 or 3 dimensions. (This doesn't rule out the possibility
> that some aspects of the data are represented non-geometrically, e.g. by
> colour.) I am weighing the pros and cons of two different ways of handling
> scaling. Perhaps newsgroup readers would like to comment.
>
> Approach A: Use COORD_CONV
>

>
>
>
> Approach B: IDLgrModel::Scale & Translate, aka "COORD_CONV is evil"
>
>
>
> There is a major advantage to approach B:
>
> Once a set of atoms and axes has been put into a model, any further changes
> to the scaling of that model will not disturb the geometric relationship
> between the axes and atoms. With approach A, if you want to fiddle with the
> scaling of a graph after it has been set up, you have to EITHER: a) keep a
> list of all the atoms & axes at the top level of the graphics application
> and apply changes to the COORD_CONV properties of each one; OR b)
> have each axis keep track of all the atoms that are scaled to it and pass on
> any changes--then the graphics application just needs to keep track of the
> axes. I have been experimenting with the latter method via a class I call
> MGHgrMSaxis ("MS" = master-slave) and it's not as complicated as it sounds,
> but it's still a level of complication I'd like to avoid.
>
>
>
> Summary:
>
> The sole advantage I have cited for approach B is a biggy.
>
> What do others think? Is there an approach C I haven't thought of?
>
> Thanks to Randall Frank for setting off this train of thought.
>
> ---
> Mark Hadfield
> m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield/
> National Institute for Water and Atmospheric Research
> PO Box 14-901, Wellington, New Zealand

--
[[[[[[[[[
[[Dr. Martin Schultz   Max-Planck-Institut fuer Meteorologie   [[
[[Bundesstr. 55, 20146 Hamburg   [[
[[phone: +49 40 41173-308   [[
[[fax: +49 40 41173-298   [[
[[martin.schultz@dkrz.de   [[
[[[[[[[[[[