Subject: Re: When should objects be used?
Posted by Martin Schultz on Mon, 28 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Richard G. French wrote:
>
> Nearly all of my IDL programs started as interactive bits of code, saved
> in
> a journal file and then expanded into programs. [...]

I've been thinking about using objects for a while now, but every time I am ready for it there is something that has to be done right away so I postpone again. From what I read about objects so far, I would think that (perhaps) the most important
difference between object oriented (direct) graphics and old-style sequential graphics programming is the decoupling between setting the appearance of a plot and actually creating the plot (what David called "sticky"). An old=style program might look something like this:

```
   pro myexcitingplot,data,color=color,line=line,...

   if (n_elements(color) ne 1) then color = 1
   if (n_elements(line) ne 1) then line = 0
   ; ...

   plot,data[0,*],data[1,*],color=color,line=line,...
   end
```

And you will have to store the color, line type etc externally and pass it into this routine whenever you want to redraw the plot (e.g. on a postscript printer). A little more sophisticated would be the use of a stucture containing all the style parameters you want, e.g.

```
   thisstyle = { color:1, line:0 }
   myexcitingplot,data,thisstyle
```

where myexcitingplot would now "parse" the structure for relevant information:

```
   pro myexcitingplot,data,stru

   if (ChkStru(stru,"color")) then color=stru.color
   ; ... (this uses my ChkStru routine)
   plot,....

   end
```

In OOP you would instead write

```
    myexcitingplot->SetProperty,color=1,line=0
```

and you would want to store your data within the object as well (or at least a pointer):

```
    myexcitingplot->SetData,Data
```

and the Draw method would (re)produce your plot with the new properties:

```
    myexcitingplot->Draw
```

As long as you keep your object reference somewhere (and don't destroy the object ;-) you can repeat that Draw method anywhere and anytime you like. In the old-style approach you would have to keep the structure with all the plot parameters and the data instead which is more vulnerable to unforeseen changes. I would argue that it is possible to do (almost?) anything that you can do with objects without them as well, but it will occasionally be much more complicated. On the other hand I have a feeling that OOP requires more thinking in advance and lends itself less to a more experimental programming style -- but that may be a wrong impression because I am just not that familiar with it.

.... paused to think ;-) ....

But how about batch job processing? As mentioned before, you need to detroy objects after use shall they not leak your memory. Hence, if you want to process, say 1000 data files all in the same way your code would look like

```
  for i=0,n_elements(filename)-1 do begin
    thisplot = obj_new(filename[i])
    thisplot->SetProperty,.....
    set_plot,'PS'
    device,....
    thisplot->Draw
    device,/close
    obj_destroy,thisplot
  endfor
```

So not that much different from an old-style program. But the nice thing is you can reuse that object say within a widget program where you can mess around interactively. AND (what I think hasn't been mentioned intensively enough) you can inherit your object methods without having to rewrite everything. Thus, say myexcitingplot would produce a world map in a cylindrical projection, you could write myexcitingpolarplot

which would inherit almost everything but take care of a few extra
"features" that one needs to be aware of in polar plots (i.e. all that
would have to change is your Draw method).

Hmm. That was long and maybe nothing new, but it helped me sort through
some things (writing as psychotherapy ;-)

Cheers,
Martin

|||||||||||||||\\\\\\\\\\\\\-------------------//////////////// //|||||||||||||||
Martin Schultz, DEAS, Harvard University, 29 Oxford St., Pierce 109,
Cambridge, MA 02138 phone (617) 496 8318 fax (617) 495 4551
e-mail mgs@io.harvard.edu web http://www-as/people/staff/mgs/