Subject: Object epiphany: A new way of building widget applications
Posted by Martin Schultz on Wed, 04 Apr 2001 21:05:33 GMT
View Forum Message <> Reply to Message

Hi all,

With almost a week delay, I finally get around to release the first version of a new class of IDL objects: the MGS_GUIObject hierarchy. Don't shy away immediately! It's far easier than it sounds, and once you will have discovered how easy it now becomes to develop widget applications, you will get hooked! Ben Tupper has managed to get something running within a day.

Even though the package is only 68k (mostly documentation ;-), I prefer not to attach it to this message but distribute it via anonymous ftp (I may put it on my web site tomorrow). You can get guiobjects_1-0.zip from
ftp://ftp.dkrz.de/pub/Outgoing/martin_schultz/idl/

Before describing the object hierarchy a little, here is a quick runup over the major features of this new approach:

(1) You can use the same object program to create blocking, non-blocking, and compound widgets

(2) In general, you will not have to worry about widget event handling in great depth. Most of this is taken care of for you in the basic MGS_GUIObject already

(3) If you compose a widget out of several compound widgets, you simply add them to a built-in container, and they will be managed automatically for you. You only need to overwrite 3-4 methods (with very simple code) to get very powerful widget programs.

(4) Due to the hierarchical class structure, it is very easy to maintain the code. If you or I want to add a new basic feature to all widget programs, the change only needs to be made once (in MGS_GUIObject), and all programs will "see" the change immediately.

Most of the programs described below contain an example procedure at the end, so if you ".run mgs_...__define" and then call "example, /block" you get to see what they do.

Hierarchy:

MGS_BaseObject : this is the father (or mother) of all my objects. It contains a pretty sophisticated ErrorMessage interface including traceback information and the choice to display errors as dialogs or on the log window, and including a debug level. This object can also clone itself and do a few other things.

MGS_Container : this object enhances the IDL_Container object by allowing object access by name. Compared to the first version of about a year ago, this version now also inherits from MGS_BaseObject to take advantage of the error handling mechanism.

MGS_GUIObject : This is the "generic" widget object that lays the foundation for all widget programs to come. It is fully functional in itself, although all you will get is an empty frame with two buttons ("Accept" and "Cancel" or "Apply" and "Close" depending on the Block keyword to the GUI method). The object already handles all major event handling and the general procedure of building and displaying a widget hierarchy. Inherited objects will only need to add the elemental or compound widgets they need in a BuildGUI method; registration with the XManager, positioning of the widget, etc. is all taken care of by MGS_GUIObject. This object also allows you to set the default font and the label font of your widgets (apparently not on a Mac: Pavel, please test! - and only if you close and rebuild the GUI).

MGS_Field : This is a sibling of David's FSC_Field program but extensively modified to make it fit into this new framework and to make it even more modular. As a bonus, I added a character_mask feature that allows you to specifiy the characters that shall be allowed for string input. And, special offer for JD ;-) you can even specify a regular expression which must then be matched by the input field [needs some tuning, still]. Note, that you can use this field object right away with four statements:

```
  thefield = Obj_New('MGS_Field', value=!DPI, min_valid=0.,
max_valid=10.,  $
                   labeltext='Enter value: ')
  thefield->GUI, /block
  print, 'Current field value is ',thefield->GetValue()
  Obj_Destroy, thefield
```

... and if you want to use it as a compound widget, see:

MGS_InputMask : This widget object simply collects information from a structure and displays the structure tag values (must be scalars) in individual MGS_Field input fields. Just think of how much work this would be in classical widget programming, then take a look at this code. You'll be amazed!!

MGS_RangeInput : This object demonstrates how to use two compound

widgets (again MGS_Field) and link them together to provide a range input pair of fields. I admit they still need some more communication, but you'll get the idea ...

MGS_Drawcolor : Again, you might recognize the name similarity to David's FSC_Drawcolor program, and, indeed, that's what this does, too: selecting a color.

MGS_GUIDemoObject : This object is yet another demonstration of how to combine compound widgets. It builds a widget with three color picker objects and one text field.

Now it's time for me to call the day,

hope you enjoy this new stuff,

Martin

PS: A great big thank you to David again for providing all this great stuff on his web page - and for developing MPI_Plot which served as a basis for the development of my programs.

--
[[[[[[[[[
[[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   [[
[[[[[[[[[