Subject: Suggestion for IDL6 -- variable declarations
Posted by Mirko Vukovic on Sat, 18 Sep 1999 07:00:00 GMT
View Forum Message <> Reply to Message

D. Fanning mentioned compiler options on another thread.

What sometimes bytes a programmer is when a variable changes format
while one is not paying attention.

I've been wondering how usefull would it be to declare variables
to be of certain type? By that I don't mean only integers, reals,
vectors, etc, but something like Pascal and C have, but more closely
related to IDL. Some examples of such variable types

window index
color table index
color triplet
Axis range
widget index
flag
output device
Annotation
etc ...
(and vector and array versions of those)

What is nice about these is automatic data checking. Trying Flag
=10? That is an error! It would allow one to write much neater code.

One could not do an (Window index)^2. But if you really needed
something like that you could do:
Var=Window Index Variable
Value of New Window Index = Var^2
New Window Index Variable = Value of New Window Index

I've been implementing those myself via objects, and only recently
realized some
of the usefullness of this approach. All of a sudden, I needed to
figure out when a variable was ``not assigned'', or how to de-assign it
although it existed in the program. With objects as these, it is easy
to decide what value means that the variable is not assigned (-1 for a
window index), and make it act accordingly.

Another example is the Annotation type object. As someone cried out for
TeX-like notation instead of the !- control sequences, this is
accomplished easily within such an object. You pass it the sequence in
TeX-like notation, and when IDL needs it, it spews out the
!-like notation that IDL understainds. What is even better, you can
specify Annotation to use certain fonts by default (like roman complex)

A much more usefull feature that I see coming up is for building a GUI
to these variables. Once I have a consistant set of variable
declarations like these, each has a ``natural'' (that has to be defined)
GUI. But once I define these, to build a GUI, all I need to do is pass
a command CreateGUI with a base widget ID, and I get it.

What appeals to me in this approach is that there is no need for those
multi-line widget_xyz commands, that make reading of programs very hard
for me. Instead of that, a sequence of single line CreateGUI commands
is issued.

Admittedly, with this approach, one looses all the flexibility of
``designing'' the GUI interface (how in detail each GUI component
looks, etc). I haven't though this one out in detail yet, but I
can see a separate routine or file with option specs. It would have
lines like (Ylog is a flag):

```
*Ylog Options
Layout - horizontal  ; (can be vertical)
Text: "Log" "Lin"
```

Now you have separated the design of the of the GUI from the
program execution. Very desirable in my opinion.

For those that know TeX and LaTeX, IDL's widget commands correspond
to TeX -- all the raw power is available to you. The above described
approach to GUI creation is more like LaTeX's -- you are limited,
but the resulting code is more readable, and deals in more human-like
issues. You can customized it and suit it to your needs, but that
recquires some additional IDL programming.

I haven't implemented the GUI stuff yet, that awaits the latter part
of October or even November.

Mirko