Subject: Re: How Object-oriented?
Posted by davidf on Fri, 23 May 1997 07:00:00 GMT
View Forum Message <> Reply to Message

Stein Vidar Hagfors Haugan <s.v.h.haugan@astro.uio.no> writes
in a thoughtful article:

> Unlike David Fanning, however, I'm not completely happy with the
> pointer implementation - as far as I have understood it, you
> have no "address of" operator. I.e., you cannot *mix*
> normal and pointer variables in the sense that you cannot
> make a pointer variable point to a normal variable like you
> can in e.g., C, and thus change (or read) the contents of that
> variable by using the pointer.

A little experimenting convinces me that Stein is right
about this, but I am nevertheless still pleased with the
implementation. What if RSI *had* made it possible to
point to normal variables? I can imagine all hell breaking
loose within IDL as variables were willy-nilly redefined
and pointers were now pointing everywhere or nowhere.
I presume the cost of the software, as high as it is, would
skyrocket, just to pay for the additional technical
support engineers RSI would have to hire to sort it out.

I think they absolutely did the best, sensible thing to
make pointers that point at heap variables, which are
global in scope and with can be reached with easily copied
and passed-around variables.

> This increases the amount of work that needs to be done to
> make existing programs benefit from pointers in conjunction
> with *new* programs. Let's say you would like to make a huge
> dataset used in an existing program available to a new
> routine (or preferably, an object). It would be nice to be
> able to pass the object a *pointer* only to this data, to
> avoid copying the data, and allowing the object to keep the
> pointer for future reference. This seems to be impossible
> (though I may have misunderstood the documentation..).

Well, I'm no great fan of the documenation, but in this case
I think you read it correctly.

> Instead, you'd have to rewrite (parts of) the existing program
> to use pointers instead of normal variables - though I
> presume routines being passed a variable by reference wouldn't
> care if the call was e.g.,
>

>     my_routine,*var_ptr    instead of    my_routine,var

I think in some instances you will probably have to re-write
your programs to use pointer dereferences, but you are right
about the syntax above. It doesn't matter.

>  I guess that the reason for the lack of an address operator
>  is that "normal" variables are allocated on the stack,
>  whereas pointer variables are allocated from "heap" memory...
>  I.e., some nontrivial part of the information on a "normal"
>  variable is kept on the stack - not just the addres of where
>  that information is.. This would of course make it dangerous
>  to make a pointer to a stack variable, since e.g., the pointer
>  could be stored in a common block and then accessed after the
>  stack variable had been deallocated (and the space possibly
>  allocated to something completely else!). Or maybe it's just
>  the lack of reference counting that does it - local variables
>  are automatically deallocated (irrespective of where the actual
>  variable is stored) on returns....

I think you hit the nail on the head. It would be exceedingly
dangerous to give real address operators out. IDL would have
to be completely redesigned to accommodate it. (And you
and I will be retired before that version is released anyway.)

>  Actually, one may write object oriented programs in almost
>  any language - certainly you can write OO programs in IDL 4,
>  compound widgets being the obvious example of object orientation,
>  though with handles etc other possibilities are clearly present.
>  Over the last 2-3 years (much to my surprise and amusement) I've
>  been "rediscovering" OO programming in IDL after ceasing to use
>  Simula several years earlier.

I wa surprised to find out--once I looked into what
object-oriented programming was all about--that I had been
teaching many of the fundamental concepts in my widget
programming courses for several years. "Whoa", I thought
to myself, "This is easy enough that even *I* can figure
it out." :-)

Alas, I'm still bogging down on surfaces with axes, but
now I have a real version to play with. :-)

>  David's (temporary, I'm sure!) frustration about the
>  object-oriented graphics comes as no surprise at all - just
>  think about the time we've all spent getting "up to speed"
>  with all of the "direct" graphics stuf, and writing our
>  own favourite procedures to do this and that exactly the way

> we want. Without having any hands-on experience of OO graphics
> in IDL 5.0, I imagine it's almost like throwing away most of that
> experience and all those neat solutions all at once, and having
> to get up to speed once more in an unfamiliar, if not hostile
> terrain!

"Hostile terrain". I like that. ;-)

I appreciate the vote of confidence, Stein. Yes, I think I will
figure it out sooner or later. They say people pay you for your
experience. No better way to get a lot of experience, I guess,
than to be willing to make a lot of stupid mistakes. I'll
struggle on. But if you run across any information about axes,
I would love to hear it. :-)

Cheers!

David

---------------------------------------------------------
David Fanning, Ph.D.
Fanning Software Consulting
Customizable IDL Programming Courses
Phone: 970-221-0438  E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com