Subject: Re: Top 10 IDL Requests
Posted by Michael Plonski on Tue, 01 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

1) Complement of Where - so the same call returns where, and a named variable returns where-not

2) Object graphics improvements for 2D - it seems like the only way to get any performance on 2D object graphics is to use the IDL emulation of Open-GL. The hardware based Open-GL seems to make you pay all the 3D overhead even if you are only working in a 2D perspective.

3) Better Mapping - specifically a map object that works with object graphics (could be one of those 2D objects from above) and User Defined Mapping. PV-Wave long ago had a user defined map feature though it was not documented to the point that I could ever get it to work. What I would like is to be able to set up a map projection (mostly for raw satellite data) where I pass a function that converts lat/lon to X/Y and vice versa and then all the other map features are available to me. Also Map_set never quite did what you expected. If I recall it used to add something like .01 to the boundary values you gave it. Screwy things happened if you thought you defined a rectangular region in a particular project, but map set didn't (you might get a little 1 or 2 pixel wide triangular wedge along a side that did not have a valid coord_transformation). I found, I always had to read the 4 corner points after using map_set to see what it used since it rarely seemed to use what you asked for. These problems only show up when you are trying to resolve precise coordinates down to the pixel level. A good description of how mapping works and what all those system map variables actually contain also would be nice.

4) Multiple inheritance from objects with a common data field name. I like that you can explicitly reference an inherited objects methods, why can't they make it so that you can also reference an inherited objects data structure. IDL could follow the same multiple inheritance rules that it applies to methods. I actually don't even care about being able to specifically access all the fields in the inherited objects as unique fields. My primary beef is that the inheritance fails if you have conflicting field names in two different objects. I would be content if they allowed a flag on the inheritance that would just take the field name and data type from first inherited object with that field name and used it. This is what happens when you multiply inherit methods with the same name, except that IDL also allow you to access any of the unique methods if you fully qualify the name with the inherited class. I understand that this would probably be a big performance hit to add this capability to object structures, but it would add some flexibility. I assume that multiple object inheritance just uses relaxed structure concatenation which is probably the cause of this

limitation. While we are at it, why not fix this problem on structure
concatenation so that you can concatenate structure with conflicting
field names, where the first structure with that name get to define the
field data type.

5) public, private operations for object inheritance

6) Function Autodefinition files for structures: I hate that you can't
set values for structures in autodefinition files. Objects fix this
problem somewhat, but then the object structure is private data so you
can't access it easily (yes there are workarounds). If RSI does item 5,
that would take care of this item. The other alternative is to allow
functions for structure autodefinition. If a procedure does the
autodefinition you get back a null structure. If a function does the
autodefinition you get the named structure with the default values set
as returned by the function. It just make the semantics easier than
having to call a function to autodefine and set the values for a
structure.

7) Integrated support for overlay bit planes. It would be nice to have
an easy way to overlay a few bit planes of different colors on top of an
image. This is easy to do if you want to give up some of the color
values and embed it in the image, but I want a way to toggle - on /off
near instantly (like when you load a new 256 bit color map). The current
approach used is to embed the overlay into copies of the data into
various pixmaps and then load the respective pixmap (ala the old flick
routine approach). The problem is I have dozens of bit planes and large
images (10s of Megapixels) so you waste a lot of time and memory making
pixmaps that may never be displayed.   I think the only solution may be
the color map for the alpha channel in object graphics, but it would be
nice if there was some efficient way to do this in direct graphics.
Maybe someone can explain what a 32-bit true color direct graphics
window is, since the channel variable on TV only lets you get to 24 bits
(RGB). If we can have 32 bit object graphics, why can we have 32 bit
direct graphics with an alpha channel and let IDL deal with how the
alpha channel is implemented in an efficient manner.   Part of my
problem may be that this application is currently using direct graphics
with scroll bars for large images, which only a small portion (say 1
Megapixel) is displayed at a time. If I wasn't so lazy I could keep
track of what portion is currently displayed and then only update the
bit plane overlay for the displayed portion of the data to increase the
flicker rate. But then, I would have to keep track of every time the
user scrolled and update manually.

8) A class browser for the IDE. It would be nice to be able to see
class structures in the IDE. It would be really nice to be able to

generate a UML diagram from a project file. If IDL was to publish the
description of the project file, it might be able to write some script
to extract the classes
from the files.

Mike Plonski
