
Subject: Urgent object question

Posted by [Ted Cary](#) on Wed, 23 Jan 2002 03:48:47 GMT

[View Forum Message](#) <> [Reply to Message](#)

This is a long one, so you can skip to the last paragraph if you like.

I have several object widgets that all subclass the same object widget. Each of these "satellite" object widgets is realized in its own top level base, with its own user interface, and performs a specific type of analysis on the subclassed "planet" object widget. The analysis performed depends on the state of the subclassed widget and can in turn change that state, behavior that is easily achieved using object widgets.

The problem is that I would like to subclass all of these widgets at once, making one big multi-function object widget utility with a user interface to switch between each type of analysis. Several of the analysis windows could be open at once, and the inherited object widgets could share data if needed, all changing simultaneously in response to any calls to the common superclass's methods. That was in fact my plan when I wrote everything, when I had no experience in IDL. Now that I have a little experience, it does not seem so easy.

To extend the metaphor, all of my satellites crash into one another. Of course my multi-function utility object cannot inherit all of these objects at once, since they all subclass the same object and the structure fields would conflict. I could get the structure definition to be right using Struct_Assign, but then the methods of all the objects would not be inherited. I can't easily just use object references in the place of inherited self fields because the communication between these satellite object widgets and their planet object widget has to be two-way--as soon as the user changes the state of the common superclass widget, satellite object widget fields are updated, and vice versa. With object references available to them, the satellite widgets could call methods on the planet, but they would not automatically respond to methods called on the planet by another satellite or by the user. Pointers into the objects would violate encapsulation principles and confuse me. I could just cut and past and rename all the methods and make my big object structure using Struct_Assign, creating one big object in one big file, but this is ugly and would make the program hard to maintain and read, since I'll be wanting to add more analyses in the future.

What I want is something like an Obj_Assign procedure, some way to inherit all subclassed object methods while creating a self with only one unique copy of every field, even if two subclassed objects share the same field. I suppose the problem with this is that then the Init methods of all the inherited objects might each try to initialize the same self fields, but in

my case none of the self fields would ever be Init-ialized by more than one object Init method anyway. I was hoping a more experienced programmer could help me on this one.

Does anyone have experience with a problem like this? Or any suggestions? .Fanning?
