Subject: Re: passing parameters from base to base
Posted by JD Smith on Thu, 28 Nov 2002 00:14:19 GMT
View Forum Message <> Reply to Message

On Tue, 26 Nov 2002 11:47:37 -0700, Pavel A. Romashkin wrote:

> This sounds sensible, although I didn't see one piece of information (or
> did I miss it?) - how do ObjMsg objects know about each other? There has
> to be a place they go to when they want to "sign up". Is the awareness
> established at initialization, or in a common block? How do they become
> aware of each other?
> Cheers,
> Pavel
>

You put your finger right on the tricky bit (good eye ;). Right now, my solution is to "introduce" objects which want to communicate to each other on a higher level. E.g. I might say in my wrapper program:

oDraw=obj_new('tvDraw')
oFunk=obj_new('tvFunky',oDraw)
oNother=obj_new('tvNother',oDraw)

and oFunk could sign up for the tvDraw messages it needs. Perhaps oFunk even builds an ObjMsg object of its own and signs up with it directly. Usually this is perfectly adequate: relations and introductions can be done at compile time or Init time. This works well for, e.g., building a viewer shell with all the desired plug-ins, especially when you need to carry widget bases around to build the interface inside.

But sometimes you'd like to find new ObjMsg objects to communicate with at run time (maybe because they didn't exist yet when you were first made). For this I use one other mechanism: if you have an ObjMsg object, you can query for other ObjMsg objects on *its* message recipient list, typically by class name (sort of a "friend of a friend" thing). E.g., if I knew that my oDraw object probably had relations with an object of class 'tvColor' that I needed to talk to, I'd say:

oColor=self.oDraw->GetMsgObjs(CLASS='tvColor')

I'd probably make it robust in case there's no such object to communicate with. Objects which have an absolute dependency on each other to function should probably introduce themselves the "old fashioned way".

Any suggestions much appreciated. Another option is to publish

"capabilities" or lists of potential messages sent in a common block of some kind, but that gets complicated with many multiples of the same type of object on the global stack.

JD
