Subject: Re: Object Widgets
Posted by Struan Gray on Wed, 17 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

J.D. Smith, jdsmith@astro.cornell.edu writes:

> http://www.astro.cornell.edu/staff/jdsmith/objmsg/objmsg.htm l
>
> Anyway, comments are welcome.

Nice stuff. Thanks for the sneak peek.

I have something similar (though I got lazy and subclassed from IDL_Container rather than write my own list manager), and am still muddling over whether to make the actual messages objects as well.

At some point one has to start deciding how the message should be handled by the receiver. It looks to me as if you do that by modifying the ObjMsg::Message method, which implies that all the entities participating in the message passing have to be objects and have to have a Messsage method.

For older widgets which I can't be bothered to objectise (fewer and fewer with time) I created a Gossip object which exists solely to turn a call like

```
sendlist[i]->Message, msg
```

in ObjMsg::MsgSend into a suitable widget event and put it in the event queue. The old-style widget then processes it with a conventional old-style event loop.

This would obviously work in your scheme too, but I extended (or, more accurately, am in the process of extending) the gossip object idea so that even objects and object widgets generate gossip objects to handle messaging (they don't *have* to of course). This means an object can effectively have different Message methods for communicating with different objects. I would need to see how you create and hendle your msg structures, but I get the feeling I seperate the behaviour of the list manager and list items more explicitly than you do.

As an example: I have a generic IDLgrModel viewer which generates sub-widgets in their own TLBs to control settings for the trackball, the viewing position, colours, etc. Instead of a single Message method which looks at fields (or properties) of the message to decide where it's coming from and what to do, I have several gossip objects which already know which sub-widget they are dealing with.

As is often the case in OOP, all I'm really doing is avoiding a lot of IF or CASE statements in a single, big routine (in this case, a general message handler). It's a style issue, but I find this easier to get my head round, a little easier to adapt to specific cases, and a tiny bit faster as it cuts down on the need for Message methods to run detailed checks on every possible event type.

Struan
