Subject: Re: passing parameters from base to base
Posted by Pavel A. Romashkin on Tue, 26 Nov 2002 18:47:37 GMT
View Forum Message <> Reply to Message

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

JD Smith wrote:
>
> This is similar to a system I've developed over the years (and which
> hopefully will be available in the form of a suite of viewer tools
> "real soon now"). Instead of sending events, I abstract widget events
> and other forms of intercommunications among objects as "messages",
> and provide a framework for sending messages, signing up to receive
> those messages, etc. I call it all "Object Messaging", and "ObjMsg"
> is the parent class. Any ObjMsg object can send and receive messages
> from any other ObjMsg object. Widget programs can talk to non-widget
> programs, and all communication is treated the same way.
>
> Want to get messages from somebody about new pudding flavors?
>
> somebody->MsgSignup,self,/PUDDING_FLAVORS
>
> Tired of hearing about pudding flavors?
>
> somebody->MsgSignup,self,/NONE
>
> etc. The flow of messages is not static; indeed it sometimes changes
> quite often during run time. At its essence, there is, of course, no
> fundamental difference between this mechanism and just carefully
> keeping track of which methods to call on which objects when, but the
> beauty is, it relieves the program from having to remember all this,
> and tends to promote smaller, more modular code.
>
> For instance, my "tvDraw" object contains a draw widget, and sends all
> kinds of message, including simple motion events. At any given time,
> anywhere from 0 to ~10 different objects are interested in motion
> events. But tvDraw simply doesn't care about all that:
>
> self->MsgSend,/TVDRAW_MOTION
>
> is sufficient to get all messages sent to all the relevant parties
> with no further effort. What this means is that, if later on you

> write another ObjMsg object which would like to hear about motion
> events, no new code is required.
>
> The up-the-widget-heirarchy event passing paradigm is simple and
> useful, but for complex programs in encourages you to put everything
> in one place. Object Messages essentially breaks free from this
> paradigm: events and messages can be sent anywhere, at anytime.
>
> Once freed from the shackles of object (and especially widget)
> intercommunication, you can better resist the urge to include
> everything but the kitchen sink in single burgeoning piles of code,
> and write small, focused objects, designed to solve one task well.
>
> Of course, it's all smoke and mirrors until I show some code...
>
> JD
