Subject: Re: widget objects
Posted by JD Smith on Wed, 31 Mar 2004 20:48:04 GMT
View Forum Message <> Reply to Message

On Tue, 30 Mar 2004 15:56:25 -0500, Michael A. Miller wrote:

> I've got some questions regarding objects and widgets and am
> hoping that someone here can help point me toward a robust
> design. Here's the situation...
>
> I have a collection of IDL objects that include GUI components.
> I use these in two ways: by themselves in their own top level
> windows and swallowed in widget_bases of other GUIS, like a
> compound widget. This works pretty well for me, but has some
> limitations. For example, I have a color map object that allows
> me to select a color map and adjust the limits. It inherits from
> a publisher class so other objects that might use it can
> subscribe. When the color map is changed, it tells its
> subscribers. Typically a subscriber will redraw something using
> the color map when it is notified of a change.
>
> Now I've decided that in some cases, it's is ok to have a color
> map GUI on the screen (as part of a main window or in its own
> window) all the time. In other cases, I'd really like to put the
> color map in its own window which can be drawn and withdrawn at
> will, like with tk's withdrawn window state. I don't believe
> that this is possible with IDL, so instead I thought I'd separate
> the color map object from the gui so that an instance can create
> and activate the gui and destroy it at will (at the programmers
> will, that is) without destroying the entire object.
>
> This sounds fairly straight forward, but I'm confident that I can
> make it complex enough to be cumbersome. I wonder if any of yous
> idlers have given this sort of thing some thought and have ideas
> that you'd be willing to share or just discuss in general. The
> part in particular that I'm wondering about is the capability of
> putting a gui component in it's own top level base or in another
> window. I've implemented this with a widget object class which
> is appended below. Using it just seems a bit cumbersome to me
> and I can't quiet put my finger on why.

I do this without any difficulty. I usually have a separate method call which actually contains the widget definition code and (optionally) XManager call. I call it when necessary, and provide a means to test if the GUI is running (XRegistered is convenient, as is widget_info(/VALID_ID)). To have it function either standalone or as a compound widget, pass an optional parent widget ID. If the parent ID is passed and is valid, build all widgets beneath it, and don't

call XManager (since the parent application will do that). If it's not passed, make your own TLB (perhaps MODAL) and proceed the same way, calling XManager this time to manage your own events. In either case, the events will pass through your own event handler (which, since you can't rely on XManager to set for you, you should do using WIDGET_CONTROL on the necessary top-level widget). The widget object doesn't know whether it's running *inside* another widget or as a stand-alone. Nor does it care whether it actually has any widgets realized and showing. The only potential difficulty appears if you want to pass events "up" from your widget-object-as-compound-widget to be handled directly in the parent app's event handler. You can do away with this difficulty is you simply stick to your publish/subscribe based communication in all cases, and swallow all events (keeping them internal to your object's widget hierarchy). This methodology will pay off in the long run, since then you can reuse the same widget object in a variety of ways, and its communication pathways remain flexible.

To have, e.g., a colormap widget set appear and disappear within another base, you can just wrap the widget building code in widget_control, self.parent,UPDATE=0, widget_control, self.parent,UPDATE=1 and destroy it when it's not popped up. Since the object lives through this process, there is no problem maintaining state: it simply appears that the widget disappears and reappears on command. Except for the heaviest of widgets, this is acceptably fast.

JD

---