Subject: Re: widget_control and group_leader
Posted by nrk5 on Sun, 24 Dec 2000 07:34:07 GMT
View Forum Message <> Reply to Message

In article <3A43E6DF.98E40A55@astro.cornell.edu>,
JD Smith <jdsmith@astro.cornell.edu> wrote:

>
> Hi Nidhi, how's the weather in Fargo? Glad to see you didn't take my
> advice and are hard at work.

Such is the life of a lowly college student. Besides, I live in Fargo.
What else can I do? :)

Thanks for explaining the project, btw.

> There's nothing to say a single motion event can't simultaneously
> display a zoomed image, update a data/coordinate status line, and
> stretch a colormap, all at once, even from within different entire
> widget trees or programs. You obviously have to be a bit careful
> throwing all these events around, but in practice it's no problem.
This
> means, you never have to use:
>
> widget_control, event_pro=foo
>

Let me paste in a bit of the code from program A. In the event handler
that I am mostly concerned with, the user sets the mode. mousemode
cases 0-3 were already there, so I added 4 for uniformity. When 4 is
selected, events on the draw_widget are sent to the foreign event
handler.

```
pro a_event, event
; Main event loop for atv top-level base, and for all the buttons.
...
widget_control, event.id, get_uvalue = uvalue
...
case uvalue of
   'mode': case event.index of
      0: widget_control, state.draw_widget_id, $
        event_pro = 'atv_draw_color_event'
      1: widget_control, state.draw_widget_id, $
         event_pro = 'atv_draw_zoom_event'
      2: widget_control, state.draw_widget_id, $
         event_pro = 'atv_draw_blink_event'
      3: widget_control, state.draw_widget_id, $
         event_pro = 'atv_draw_phot_event'
```

```
      4: widget_control, state.draw_widget_id, $
        event_pro = state.foreign_event_handler $
        + '_event'
      else: print, 'Unknown mouse mode!'
    endcase
```

> You can just process and dispatch events from within the already
> existing widget handler. This also obviates your "Foreign Event
> Handler" button, as this can all be automatic, and you can be using
> those events all over the place, whenever appropriate.

The functionality I'm going for is that the user can decide when to use external event handlers and when to let program A run 'naturally'. At the moment, I have tried to keep foreign_event as general as possible. Each B can do whatever it pleases with its own particular foreign_event handler. The two things (now) registered with A are the foregn event handler to use and a widget_ID to use. Whatever that needs to be.

> What I would recommend in this case is set up a foreign event handler
> *method*, since the foreign widget is an object. That is, have a
> routine to sign up for events from A. from within B., like this:
>
> a_signup, self, "Handle_A_Events", /Button, /TRACKING
>
> or some such. Then, each "foreign" object can sign up for whatever
> events it wants.

The object method part makes sense, but "signing up" is a bit confusing.

> All
> you'd need to add to A. is code to manage this "signup" list (add,
> delete entries -- a pointer on A.'s common block would be most
flexible
> here), and a small function which uses:
>
> call_method,method,obj, ev
>
> to dispatch the event from within A's standard event handler, based on
> the events requested (B would turn on and turn off the event spigot
when
> appropriate).

How do I register event/object pairs? Ok. So here I'm a little lost (Caution: Newbie IDL-er at work). A registers the following event handlers:

widget_control, top_menu, event_pro = 'topmenu_event'
widget_control, state.draw_widget_id, event_pro = 'draw_color_event'

widget_control, state.draw_base_id, event_pro = 'draw_base_event'
widget_control, state.keyboard_text_id, event_pro = 'keyboard_event'
widget_control, state.pan_widget_id, event_pro = 'pan_event'

And everything in these main bases is differentiated by uvalues (as you can see from the above code). So I'm a bit confused about how to go about differentiating the "events requested" and how the reigstering in "call_method,method,obj, ev" works.

If you'd like to make it quite simple (e.g. no need to
> expand it later to more than one type of foreign object widget),
> dispense with the optional events, and just send them all.

So, at the
> most basic level, it's the same as having your foreign_event_handler,
> but just as foreign_event_method instead (which necessitates storing
an
>

> One more wrinkle: What if you didn't want to modify A's code at all?
> So you could drop in new versions as they become available, for
> instance. All you allow yourself to do is change the event handler
for
> A, after it sets itself up (how you get A's TLB ID is up to you). In
> this case, a special purpose event broker (call it C.) could sit
between
> A and the rest of the world. It could interpose it's own procedure as
> the primary event handler, and feed both A., and all the B.'s. It
could
> also serve as a proxy for A. when signing up different types of
events,
> etc. (i.e., the B's sign up with C., not A.!)
>
> Whatever you do, make it 1 notch more general than you think you need,
> and you'll thank yourself later.

Thats good advice, and its prettymuch why I am being so fussy about this right now. The requirements Jim has given me really dont require much, but I would really hate to have to rewrite everything (or anything, for that matter) later.

So, ideally, here's the functionality im looking for. On "foreign" mode, all events go to foreign_event_handler. If foreign event handler wants to do something with it, wonderful. If not, the event goes back to where it would go on non-foreign mode.

The quick and dirty way is to put in a simple statement in each of the four event handlers:

if (foreign) send_event, foreign_event_handler, event (or whatever).

hmm...waitaminit. what if i register foreign_event_handler as the event handler for the top level base? what would that do? Would all events then go to foreign_event_handler and then bubble up/down?

I think I'll need to sleep on this.

Thanks much

>
> JD
>

--

Nidhi Kalra
nrk5@cornell.edu

--

Nidhi Kalra
nrk5@cornell.edu