Subject: Re: event_pro for compound widgets -- fsc_field issues and handler field
Posted by Benjamin Hornberger on Wed, 08 Sep 2004 18:39:45 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Benjamin Hornberger writes:
>
>
>> I am quite confused now and can't figure out how to write a compound
>> widget which has an event_pro keyword and behaves like a regular basic
>> widget. Sorry for the lengthy message...
>>
>> My compound widget is supposed to control a stepping motor and will
>> consist of a non-editable field with the current position, editable text
>> fields for "move to position" and "move by distance", a slider to set a
>> step size and two buttons to move forward and backward in steps whose
>> step size is set by the slider. As text fields, I am using David
>> Fanning's fsc_field.
>>
>> This compound widget I want to include in other widgets by something like
>>
>> motor1_id = cw_move_motor(parent, ..., event_pro = 'motor1_event')
>>
>> When I change the step size of the slider, this should only affect the
>> CW itself (updating its internal "stepsize" value and a displayed
>> number"). I could manage to do that already.
>>
>> All other events (hitting enter in the moveto or moveby field, or
>> clicking the forward / backward buttons) should send an event structure
>> to motor1_event.pro (something like { CW_MOVE_MOTOR, id: id, top: top,
>> handler: handler, type: 0L, moveby: 0D, moveto: 0D, stepsize: 0D } where
>> type holds a code for whether it was a moveto or moveby request, etc.).
>>
>> Then, of course I would want to write
>>
>> PRO motor1_event, event
>> ...
>> IF event.type EQ 0 ...
>>
>>
>> Now, what do I do? In the CW definition function, I define
>> event_func='cw_move_motor_event' for my fsc_fields, buttons etc., This
>> cw_move_motor_event.pro will create and return the desired event
>> structure. But where is that returned to?
>>
>> And of course I have to define a keyword event_pro in my CW definition
>> function which will hold a string with the event handler procedure's
>> name. How do I make sure that in case of an event, my event structure is

>> created and passed to that procedure?
>
>
> You are on the right track here, in fact you have done almost
> everything right. Almost. :-)
>
> The way event handling works is that an event that gets into
> an event handler *procedure* is acted upon and the event is
> then said to be "swallowed". It doesn't go any further.
>
> But if the event handler is a *function*, there are other
> possibilities. Functions return values. If the return value
> from an event handler function is a structure, and if that
> structure has ID, TOP, and HANDLER fields, then the return
> value is treated as if it were an event structure, and the
> event "bubbles up" from that event handler in the normal
> way all events "bubble up" a widget hierarchy.
>
> So, if your compound widget event handler is a function (use
> the EVENT_FUNC keyword instead of the EVENT_PRO keyword to
> assign it), then you have the possibility of having an event
> "bubble up" to the next higher widget in the widget hierarchy,
> I.e., the parent of the compound widget. If you don't want the
> event to bubble up (that is, you wish to swallow the event),
> then just return something from your event handler function
> that is not a structure. (A zero is typically used.)
>
> *But* you want to be able to *assign* an event handler to
> the compound widget (and who wouldn't?). So you are
> probably going to have to create EVENT_PRO (and, to be
> complete) EVENT_FUNC keywords for your compound widget,
> and store the results in your info structure.
>
> So then, you get to the end of the compound widget event
> handler, and you have some choices to make:
>
>   myEvent = {ID:info.tlb, TOP:event.top, HANDLER:0L, ...}
>
>   ; Do I want to return anything? No!?
>   IF handled THEN RETURN, 0
>
>   ; Did the user provide an EVENT_PRO?
>   IF info.event_pro NE "" THEN BEGIN
>   Call_Procedure, info.event_pro, myEvent
>   RETURN, 0
>   ENDIF
>
>   ; Did the user provide an EVENT_FUNC?

```
>    IF info.event_func NE "" THEN BEGIN
>       ok = Call_Function(info.event_pro, myEvent)
>       RETURN, 0
>    ENDIF
>
>    ; Do I want to pass the event along to my parent? Yes!
>    RETURN, myEvent
>
>  That should get you going. :-)
>
>  Cheers,
>
>  David
>
```

Thanks, David, that helped a lot. Actually I was almost getting there myself after finding the right section in the IDL help. One more thing I wanted to ask:

Is there a reason that fsc_field will handle all events internally unless event_pro or event_func are specified? Which means, it's not possible to have events "bubble up" the hierarchy directly (or is it?).

I am using fsc_fields in my compound widget and want to have their events handled by the CW's base's event handler. It is important that event.handler is that base's ID because I store the CW's internal variables in the first child's user value (as recommended by the IDL help) and I access that in the event handler by widget_info(event.handler, /child). If I specify the event handler for the fsc_fields directly (using the same function as for the CW's base), event.handler will be wrong.

Currently, I am working around by specifying an event_func='fsc_field_event' for all fsc_fields, which reads

```
FUNCTION fsc_field_event, event
    return, event
END
```

which will make the fsc_fields handle their events externally while having the event bubble up the hierarchy (yes, I understood the point you were describing above!). That works, but seems a little ugly.

Ok, since I have a workaround it's not serious. I was just wondering why fsc_field behaves like that.

Thanks for your help,