Subject: cw_fslider
Posted by Glenn Newnham on Thu, 15 Mar 2001 09:25:34 GMT
View Forum Message <> Reply to Message

I've just tried using cw_fslider for the first time. In the past I have handled widget_slider events in their own procedure, named using event_pro. In these procedures one of the variables in the base widget uvalue structure is changed to the slider value.

Unfortunately the event_pro keyword is apparently not allowed in the call to cw_fslider. Can anyone suggest the best way to handle a cw_fslider event.

Cheers,
Glenn Newnham
Curtin University
Perth, Western Australia

Subject: Re: cw_fslider
Posted by davidf on Thu, 15 Mar 2001 15:00:15 GMT
View Forum Message <> Reply to Message

Glenn Newnham (gnewnham@ses.curtin.edu.au) writes:

> I've just tried using cw_fslider for the first time. In the past I have
> handled widget_slider events in their own procedure, named using
> event_pro. In these procedures one of the variables in the base widget
> uvalue structure is changed to the slider value.
>
> Unfortunately the event_pro keyword is apparently not allowed in the
> call to cw_fslider. Can anyone suggest the best way to handle a
> cw_fslider event.

I've never really understood why the folks at RSI write compound widgets like this. It is easy enough to add an Event_Pro or Event_Func keyword, and it seems as essential to me as defining a user value for a compound widget.

But you have several things you could do. The simplest is probably to put the floating slider widget in its own base widget, and attach the event handler for the slider to the base widget:

```
eventbaseID = Widget_Base(tlb, Event_Pro='FSlider_Events')
fsliderID = CW_FSlider(eventbaseID, Value = 54.6)
```

Or, you could easily modify the CW_FSlider code itself.

```
  FUNCTION cw_fslider, parent, $
      DRAG = drag, $
      EDIT = edit, $
      FRAME = frame, $
      MAXIMUM = max, $
      MINIMUM = min, $
      SCROLL = scroll, $
      SUPPRESS_VALUE = sup, $
      TITLE = title, $
      UVALUE = uval, $
      VALUE = val, $
      VERTICAL = vert, $
      XSIZE = xsize, $
      YSIZE = ysize, $
      FORMAT=format, $
      UNAME=uname, $
      EVENT_PRO=event_pro
  IF N_Elements(event_pro) EQ 0 THEN event_pro = ""
```

Store the name of the event procedure in the state structure:

```
   state = {slideid:0L, labelid:0L, top:max, bot:min, format:format, $
     event_pro:event_pro }
```

Then, find these two lines at the bottom of the event handler:

```
 WIDGET_CONTROL, stash, SET_UVALUE=state, /NO_COPY
 RETURN, { ID:parent, TOP:ev.top, HANDLER:0L, VALUE:value, DRAG:drag }
```

Modify these two line to these:

```
  thisEvent = { ID:parent, TOP:ev.top, HANDLER:0L, VALUE:value, DRAG:drag }
  IF state.event_pro NE "" THEN BEGIN
    Call_Procedure, state.event_pro, thisEvent
    thisEvent = 0
  ENDIF
  RETURN, thisEvent
```

IF you are being complete, the exact same thing can be done
for an EVENT_FUNC keyword, except that CALL_FUNCTION is used
instead of CALL_PROCEDURE.

Cheers,

David
--

David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155