Subject: Re: background processing and widget_event
Posted by J.D. Smith on Wed, 29 Jul 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Ray wrote:
>
> I'd like to put some low-priority, background processing into a widget
> application.
> I'd like the background processing to be interruptable - it bails if the
> user generates
> any events while the background processing is occurring. (At a later
> point background
> processing could be started again but it would have to start at the
> beginning - at this
> point I am not concenred about using partial results obtained when it
> was interrupted.) Finally, I'd like this to be workable without
> straying too far from using xmanager to manage the widgets.
>
> It seems to me that I should be able to sprinkle widget_event calls
> throught the
> background processing procedure and to have the procedure bail if any
> user events
> have been generated. Of course, the process of checking for user events
> using
> widget_event should not result in any loss or change in the order of
> queued events.
>
> Below is my current attempt. (.run test & test). I am puzzled. I
> click on "the long event"
> and processing commences as expected. However, when I click on "the
> short event"
> while the long event is processing, I see that the short event gets
> processed and then
> the long event processing continues where it left off. It seems the me
> that
> the widget_event(/nowait) call initiates the short event processing.
> Based on
> my interpretation of the IDL docs, I expected widget_event would simply
> return the next
> event and not actually process it. What have I missed?
>
> Ray Muzic
> -
>
> 8>< on the dotted line
> ----------test.pro----------
> ; example to test method for simulating multi-threading
> ; 1. create base widget, start event processing

```
> ; 2.  while "the long event" processing, occasionally check for pending
> events.
> ;     if one is found, bail out of long event routine and process the
> event.
> ;     this should require re-queueing events to not alter the order of
> event
> ;     processing.
>
> function LongEvent, event
>   common test, count, base
>   ; function needed to enable event re-queueing
>   l=0L
>   print,'Long Event'
>   for i=0,9 do begin
>     print,'Inside LongEvent, i=',i
>     ; check for queued events
>     we=widget_event(base, /nowait)
>     print,'we=',we.id,we.top,we.handler
>     if (we.id ne 0L) then begin
>       ; bail if there is a queued event
>       print,'Bailing'
>       return, we
>     endif
>     for j=0L,500000L do begin
>       l=l+j
>     endfor ; j
>   endfor  ; i
>   return,0L  ; swallow event - LongEvent has completed
>   ;
> end
> pro ShortEvent, event
>   common test, count, base
>   ; print sequence of numbers.  if re-queueing is done properly, no
>   ; events should be lost
>   print,'Short Event'
>   print, 'Count=',count
>   count=count+1
> end
> pro DoneEvent, event
>   widget_control, event.top, /DESTROY
> end
> pro test
>   common test, count, base
>   count=0L
>   ; create simple base
>   base=widget_base(/column)
>   button0=widget_button(base, value='the long event',
> event_func='LongEvent')
```

```
>    button1=widget_button(base, value='the short event',
>  event_pro='ShortEvent')
>    button2=widget_button(base, value='Done', event_pro='DoneEvent')
>    widget_control, base, /REALIZE
>    xmanager, 'test', base
>  end
```

The reason is that, since you're letting XManager do the work for you, and since you've assigned event handlers to the specific widgets, the widget event is being handled automatically for you. To put it more plainly, widget_event() returns *only* those events which are not swallowed along the way by predefined event handlers (your ShortEvent, in this example). ShortEvent is swallowing the event (by virtue of being a procedure), so even though widget_event triggers the event's processing, only 0's are returned.

To have events both handled automatically and *not* swallowed, use an EVENT_FUNC, return the event as is from the function, and don't use XManager. The reason you can't use XManager is that it *assumes* a top level event handling procedure, defaulting here to 'test_event', and so doesn't really let you get away with any unswallowed events. This behaviour of this solution will possibly depend on the implementation of the operating system on which IDL is running (since it depends on IDL sharing time with the window manager, to register the event in the meantime). This is obviously cumbersome, and I think you could accomplish what you're after with a TIMER event, which would be cross-platform.

See the man pages for info on using TIMER to run a background routine every n seconds.

JD

--
J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-6263
304 Space Sciences Bldg. |*| FAX: (607) 255-5875
Ithaca, NY 14853 |*|