Subject: Re: Xmanager R US
Posted by John-David T. Smith on Tue, 25 Sep 2001 23:02:09 GMT
View Forum Message <> Reply to Message

"Pavel A. Romashkin" wrote:

```
>
> I know they told us not to do this, but...
> Anyone who looked into Xmanager saw that it uses event directing loops.
> It tracks managed widgets using common blocks. However, WIDGET_CONTROL
> must have an internal reference to WIDGET_EVENT, because the following
> snippet works in a fresh IDL session, with no Xmanager common blocks established:
>
> ;*******************
> pro tlb_event, ev
> help, ev, /str
> end
> ;*******************
> function b_func, ev
> help, ev, /str
> ev = create_struct(ev, 'my_field', 'My_value')
> return, ev
> end
> ;*******************
> pro junk, tlb
> tlb = widget_base(event_pro='tlb_event')
> but = widget_button(tlb, value='press', event_func='b_func')
> widget_control, tlb, /realize, /managed
> end
> ;*******************
>
> Anybody knows why and how does the above work?
> This makes me wonder, why do we need Xmanager with its loops?
> Besides, the online help says that an Event_pro or Event_func are called
> by Widget_event using widget's ID as an argument. As far as I could see,
> the widget's event structure is passed as an argument instead. Oh well.
> A typo, I guess.
> Cheers,
> Pavel
```

This most surely relates to non-blocking mode, which, when introduced,
moved much of the event handling code out of XManager, and into the
internal IDL code itself. Here's a telling comment from xmanager.pro:

```
  ; This is the standard XMANAGER event loop. It works by dispatching
  ; events for all managed widgets until there are none left that
require
  ; blocking. In the best case, the command line is able to dispatch
events
```

```
  ; and there are no clients that require blocking (specified via the
  ; NO_BLOCK keyword to XMANAGER) and we are able to return immediately.
```

So, really, when you use NO_BLOCK, you're effectively telling XManager "Don't worry about really managing this widget, just return immediately... it's being taken care of." And the thing doing the care taking is inside IDL's command-line reading loop itself, which internally calls widget_event(). So obviously, not calling XManager at all is almost equivalent, except you won't be able to use XRegistered() and others that rely on the common blocks correctly.

What I didn't understand is how your trick works without the secret call to

```
 WIDGET_CONTROL, /XMANAGER_ACTIVE_COMMAND, id
```

you'll find in xmanager. I had presumed that this marks a widget as being "listened to" by the command line code, and beyond XManager's purview, and only then will the command line process events for it.

I think the answer to this conundrum reflects RSI's intentions upon introducing non-blocking mode: that all widgets should run without blocking, making XManager an unecessary relic. Then during v5 beta testing, they found people who relied on blocking, and they changed the default behavior at the last minute. You've stumbled on a side-effect of this earlier decision with the MANAGED keyword there.

My explanation: by default the command line is running widget_event() on all widgets it knows about (including those submitted by widget_control). Only when you actually run XManager will it set about blocking (not returning, but calling widget_event() itself in a loop) for those widgets for which you set NO_BLOCK=0. The internal event loop code itself does not discriminate between "blocking" and "non-blocking" widgets. I verified this with the following change:

```
pro junk, tlb,XAC=xac
tlb = widget_base(event_pro='tlb_event')
but = widget_button(tlb, value='press', event_func='b_func')
widget_control, tlb, /realize,/managed, $
            XMANAGER_ACTIVE_COMMAND=keyword_set(xac)
print,'Blocking: ',widget_info(/XMANAGER_BLOCK)
end
```

And now try:

```
IDL> junk
Blocking:        1
** Structure WIDGET_BUTTON, 4 tags, length=16:
  ID              LONG                 6
  TOP             LONG                 5
  HANDLER         LONG                 6
  SELECT          LONG                 1
** Structure <825c72c>, 5 tags, length=24, refs=1:
  ID              LONG                 6
  TOP             LONG                 5
  HANDLER         LONG                 5
  SELECT          LONG                 1
  MY_FIELD        STRING    'My_value'

IDL> junk,XAC=1
Blocking:            0
** Structure WIDGET_BUTTON, 4 tags, length=16:
  ID              LONG                12
  TOP             LONG                11
  HANDLER         LONG                12
  SELECT          LONG                 1
** Structure <81b6554>, 5 tags, length=24, refs=1:
  ID              LONG                12
  TOP             LONG                11
  HANDLER         LONG                11
  SELECT          LONG                 1
  MY_FIELD        STRING    'My_value'
```

Events are processed in both cases, but in one, it's supposedly
"blocking".

So you see, without the XMANAGER_ACTIVE_COMMAND keyword set, things are
set up for XManager to block, only they never get a chance to, since
XManager isn't ever called! It's the same behavior if blocking is
"disabled" with the XMANAGER_ACTIVE_COMMAND keyword. Since XManager
never enters the equation, there can be no blocking, and the events keep
on flowing.

JD