Subject: Re: IDL Runtime
Posted by Kristine Hensel on Thu, 15 Nov 2001 01:20:01 GMT
View Forum Message <> Reply to Message

Andre Kyme wrote:
> The event-handler is just a couple of programs that follow one after the other
> - is that complicated!! Some of them call other programs - But execution is
> just moving sequentially through the event handler routine. Pretty standard I
> think.
> The actual programs in the event handler involve the user drawing a region of
> interest (I don't use draw widgets, just a standard graphics window and a
> routine involving the cursor command) and involve moving it around, and also
> confirming they are happy with its location.
> Now suppose they get fed up with the whole business (like I'm feeling right
> now) and want to go and have a coffee - so when they are asked to confirm the
> region (I use dialogue_message with the /cancel keyword), they click "CANCEL" -
> Why should it be so complicated to just return to your menu options when I
> detect the cancellation? This would have to be a pretty standard widget
> application surely - menu, option, back to menu, option, back to menu...? What
> is the conventional philosophy for doing 'this'?

Are you saying that you have *one* event handler that gets the user to
draw a ROI, move it around, confirm that they're happy with it? If so,
I would consider breaking these up into separate events. If you put all
the code that handles button events in one routine, it could look
something like this (I've assumed that you use a draw widget, even
though you don't):

```
 case which_button of
  'draw': begin
    ; make the draw widget start returning events:
      widget_control, info.draw_widget, /motion_events,$
   /button_events
    ; (an event handler for the draw widget could
    ; store how the region of interest has been      ; defined in the
info structure)
          widget_control, info.confirm_button, sensitive=1
      widget_control, info.cancel_button, sensitive=1
  end ; draw
  'confirm': begin
    ; don't pay attention to any more cursor
    ; events:
      widget_control, info.draw_widget, motion_events=0, $
   button_events=0
      widget_control, info.confirm_button, sensitive=0
      widget_control, info.cancel_button, sensitive=0
    ; pass the region of interest, stored in the
    ; info structure, to your analysis routine:
```

```
      start_analysis, info
    ; make the Save and Print buttons available,
    ; now that some analysis has been done:
      widget_control, info.save_button, sensitive=1
      widget_control, info.print_button, sensitive=1
  end ; confirm
  'cancel': begin
    ; don't pay attention to any more cursor
    ; events:
      widget_control, info.draw_widget, motion_events=0, $
   button_events=0
      widget_control, info.confirm_button, sensitive=0
      widget_control, info.cancel_button, sensitive=0
  end ;  cancel
```

> So again, the design involves having options initially (menu). Some options
> aren't relevant initially such as "Save" and "Print", so if the user clicks on
> these I issue messages for the user telling them there's no point clicking
> these ones yet.

A more straightforward way is to just make sure the user can't press
irrelevant buttons in the first place, by making them insensitive when
they can't do anything useful.

> The most likely thing they will want to do initially is
> "Analysis". "Analysis" is the button with the event handler I described above:
> programs that just simply get executed one after the other. During execution of
> the event-handler programs, the menu buttons can't be clicked, mainly due to
> the fact that the region drawing/moving uses the CURSOR command - so the
> computer is waiting for mouse clicks inside the graphics window.

The way I've written it above, the draw widget keeps on accepting events
until the user presses the confirm or cancel button.

If you're using CURSOR (which I've never used, so forgive me), I presume
you use a certain mouse click (e.g. right mouse button) to signify that
the user has finished defining the region? The event-handling code that
calls the CURSOR function should just store the ROI once the user has
said that they're done defining. XMANAGER can then wait until the
confirm or cancel button is pressed.

This is all to do than to explain, of course, so I apologize if I've
confused your further. I suggest you buy David's book. :)

Kristine
--
Kristine Hensel   e-mail: kristine@esands.com
Environmental Systems & Services phone: +61-(0)3-9835-7901

20 Council St., Level 3 fax: +61-(0)3-9835-7900
Hawthorn East, VIC, Australia 3124