Subject: Re: Interrupting widget applications
Posted by hegde on Thu, 23 Jan 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Thanks for your suggestion. Although it works for a simple for loop, this type of implementation goes out of hand for multiple loops or recursive operations. And if it is required to monitor within a time consuming loop, I have to find another way.

My need is to have an interrupt button in the main application window and it interrupt anything going on in other windows (display windows, printer gizmos etc.,).

So far the most convenient method I have found out is to monitor the event queue with WIDGET_EVENT(interrupt button) wherever the interrupt is desired and proceed accordingly. Any suggestion is appreciated.

Thanks,

-M. Hegde
hegde@neptune.gsfc.nasa.gov

William Thompson (thompson@orpheus.nascom.nasa.gov) wrote:
: What you need to do is to break your application down into discrete events, and
: then use TIMER events to run the events in sequence. For example, if your
: program uses a FOR loop, then you could instead make each run through the loop
: a discrete event, and store the loop counter between events.
:
: For example, you might define your main base with a line like
:
: MAIN_BASE = WIDGET_BASE(TITLE='CDS FITS Generation Software', $
: /FRAME, /COLUMN, UVALUE='Timer')
:
: and then in the event handler you would have code that would read
:
: WIDGET_CONTROL, EV.ID, GET_UVALUE=VALUE
: IF VALUE EQ 'Timer' THEN BEGIN
: ... do one piece of the complete operation ...
:
: It's important to restart the timer after every timed event, to keep it going.
: What I like to do is to keep a parameter called RUNNING which can be either 0
: or 1. A start button generates an event which does some initial preparation
: (e.g. setting the loop counter to 0), and sets RUNNING=1. Then at the bottom
: of the event handler routine I put a line like
:
: IF RUNNING THEN WIDGET_CONTROL, MAIN_BASE, TIMER=0.1
:
: A stop button simply sets RUNNING=0.

:
: Bill

--
test

Subject: Re: Interrupting widget applications
Posted by davidf on Thu, 23 Jan 1997 08:00:00 GMT
View Forum Message <> Reply to Message

M Hegde <hegde@news.gsfc.nasa.gov> writes:

> What is the best way to interrupt a loop in widget applications?.
> According to my IDL knowledge, IDL doesn't support concurrency; so I can't
> have a button/widget to do that. Ctrl-C is not a smart way as it exits the
> application. Any suggestions?.

Here is the most important suggestion: Don't write loops in widget event handlers! Instead, take advantage of the fact that a widget program *is itself* a loop. I have several examples of what I mean on my web page. Take a look at the programs XMOVIE and ZIMAGE for two ways to take advantage of the widget program as a loop.

Alright, so sometimes there just isn't any other way. What then?

Then, you have a button that can "cancel" the loop. I have a "show progress" widget example that you can download from my anonymous ftp site. The idea is that something that takes a lot of time is going on. (In this example, a BIG calculation.) I want to allow the user to see the progress of the calculation and I want them to be able to cancel the calculation if they don't want to wait.

Here is the relevant section of the code that is inside the big calculation loop. The variable info.cancel is the ID of the CANCEL CALCULATION button. Notice that I am using WIDGET_EVENT to look for an event directly. This is about the *only* time I manage events myself. The rest of the time I use XMANAGER.

```
; Did the "Cancel Operation" button get clicked?
; Look for an event. Don't turn off the hourglass cursor!

progressEvent = Widget_Event(info.cancel, /NoWait)

; Is this a button event?
```

```
   eventName = Tag_Names(progressEvent, /Structure_Name)
   IF eventName EQ 'WIDGET_BUTTON' THEN BEGIN

   ; If it IS a button event, destroy the show progress widget and
   ; issue an informational message to the user to alert him or her.

   Widget_Control, info.top, /Destroy
   result = Widget_Message('Operation Canceled!!', /Information)

    ; Escape from the loop. Interrupt code would go here.

   GoTo, outSideLoop
```

You can download the "show progress" example program via anonymous ftp from ftp.frii.com. It is in the directory:

```
   /pub/dfanning/outgoing/idl_training/showprog.pro
```

Compile the program and type "test" to see it work:

```
   IDL> .Compile showprog
   IDL> test
```

Let me know if you have questions.

Yours,

David

David Fanning, Ph.D.
Fanning Software Consulting
2642 Bradbury Court, Fort Collins, CO 80521
Phone: 970-221-0438 Fax: 970-221-4762
E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com

Subject: Re: Interrupting widget applications
Posted by thompson on Thu, 23 Jan 1997 08:00:00 GMT
View Forum Message <> Reply to Message

hegde@news.gsfc.nasa.gov (M Hegde) writes:

> Hi,

> What is the best way to interrupt a loop in widget applications?.
> According to my IDL knowledge, IDL doesn't support concurrency; so I can't
> have a button/widget to do that. Ctrl-C is not a smart way as it exits the
> application. Any suggestions?.

What you need to do is to break your application down into discrete events, and then use TIMER events to run the events in sequence. For example, if your program uses a FOR loop, then you could instead make each run through the loop a discrete event, and store the loop counter between events.

For example, you might define your main base with a line like

```
MAIN_BASE = WIDGET_BASE(TITLE='CDS FITS Generation Software', $
 /FRAME, /COLUMN, UVALUE='Timer')
```

and then in the event handler you would have code that would read

```
WIDGET_CONTROL, EV.ID, GET_UVALUE=VALUE
IF VALUE EQ 'Timer' THEN BEGIN
 ... do one piece of the complete operation ...
```

It's important to restart the timer after every timed event, to keep it going. What I like to do is to keep a parameter called RUNNING which can be either 0 or 1. A start button generates an event which does some initial preparation (e.g. setting the loop counter to 0), and sets RUNNING=1. Then at the bottom of the event handler routine I put a line like

```
IF RUNNING THEN WIDGET_CONTROL, MAIN_BASE, TIMER=0.1
```

A stop button simply sets RUNNING=0.

Bill

Subject: Re: Interrupting widget applications
Posted by haimov on Fri, 24 Jan 1997 08:00:00 GMT

View Forum Message <> Reply to Message

In article <5c8dlj$996@post.gsfc.nasa.gov>, hegde@news.gsfc.nasa.gov says...
>
> Thanks for your suggestion. Although it works for a simple for loop, this
> type of implementation goes out of hand for multiple loops or recursive
> operations. And if it is required to monitor within a time consuming loop,
> I have to find another way.
>
> My need is to have an interrupt button in the main application window and
> it interrupt anything going on in other windows (display windows, printer
> gizmos etc.,).

>
> So far the most convenient method I have found out is to monitor the event
> queue with WIDGET_EVENT(interrupt button) wherever the interrupt is desired
> and proceed accordingly. Any suggestion is appreciated.
>
> Thanks,
>
> -M. Hegde
> hegde@neptune.gsfc.nasa.gov

I have similar experience, although I have not put much efforts to come up with a good algorithm based on using TIMER.

My application uses widgets for a real-time radar display. It can show several different images on one or two windows and includes various widget controled features. The main reason I am using widget_event to handle widgets is the fact that my main loop is controled by the radar data acquisition system (non-IDL program) and I want the interrupts to be based on the DAQ behaviour rather than on time intervals by TIMER.

I also have a complex situations, which include needs for cancelations.
I control all of them through a DESTROY/CANCEL button on my main widget.

If you would like to see my code send me an email. It is relatively big set of routines (about 1200 lines, not including external routines). This is my first serious work with widgets and I am sure I am far from being an expert. Nevertheless it works nicely.

I am curious to know if I should make any efforts to avoid using widget_event even if TIMER is not the natural thing to do.

Cheers, Sam

Samuel Haimov, Ph.D.
Atmos. Sci. Dept.
University of Wyoming
tel: (307) 766-2726
email: haimov@uwyo.edu

Subject: Re: Interrupting widget applications
Posted by davidf on Sun, 26 Jan 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Sam Haimov <haimov@uwyo.edu> quotes M. Hegde with respect to interrupting loops in widget programs:

>> So far the most convenient method I have found out is to monitor the event
>> queue with WIDGET_EVENT(interrupt button) wherever the interrupt is desired

>> and proceed accordingly.

He goes on to write:

> I have similar experience.
>
> My application uses widgets for a real-time radar display. It can show several different
> images on one or two windows and includes various widget controled features. The main
> reason I am using widget_event to handle widgets is the fact that my main loop is controled
> by the radar data acquisition system (non-IDL program) and I want the interrupts to be
> based on the DAQ behaviour rather than on time intervals by TIMER.

Here, in my opinion, is just about the *only* justification for using WIDGET_EVENT to manage IDL events rather than XMANAGER. Sam is exactly right. If the locus of control is in an external program, then I think you want to use WIDGET_EVENT to get IDL widget events and process them appropriately.

But suppose the locus of control was the IDL widget program. How could you get the program to respond to real-time events that were being monitored by an external program?

One possible solution would be to use TIMER events. I typically set timers on widgets that don't ordinarily receive events. Usually I use a sub-base, whose only other purpose is to organize my widget layout. I attach an event handler to that base to handle the timer event. It looks like this:

```
subbase = WIDGET_BASE(tlb, EVENT_PRO='Timer_Event', COLUMN=2)
```

When I am ready to check my external program, I want to get into the TIMER_EVENT event handler, so I set the timer to go off immediately, like this:

```
WIDGET_EVENT, subbase, TIMER=0.0
```

Now I am inside my TIMER_EVENT event handler. Here I can do whatever it is I need to do. For example, I can check my external program to see if any new data has come in from my instruments. If it has, I can plot it.

When I am finished doing whatever it is I need to do, I am set to exit my event handler. If I want to come back periodically I need to set the next timer event before I exit. Suppose I wanted

to check for new data every 1.5 seconds, then I would probably have code like this at the bottom of my TIMER_EVENT event handler:

```
IF info.stop NE 1 THEN WIDGET_CONTROL, event.id, TIMER=1.5
```

The info.stop variable is a stop flag that is usually set by a STOP or INTERRRUPT or CANCEL button of some sort. In 1.5 seconds, I get back into this event handler for the next round of doing whatever it is I do. But meanwhile, I can be processing whatever other events are being generated by my widget program, in the order in which they are being generated. This will include EXIT buttons, READ DATA buttons, etc.

If you want to see a good example of a TIMER event in action, look at the program XMOVIE on my web page. This program shows you how to do an animation correctly in a widget program. It is possible to interrupt the animation because the STOP and START buttons just cue up the TIMER events.

Hope this gives people some ideas.

David

David Fanning, Ph.D.
Fanning Software Consulting
2642 Bradbury Court, Fort Collins, CO 80521
Phone: 970-221-0438 Fax: 970-221-4762
E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com
