Subject: Re: Concurrent widget program.
Posted by davidf on Mon, 01 Jun 1998 07:00:00 GMT
View Forum Message <> Reply to Message

David Foster (foster@bial1.ucsd.edu) replies to Imanol Echave
by writing:

> Imanol Echave wrote:
>>
>>         I've a little question: Is it possible to run two IDL programs concurrently? My
>> problem is that I run a time expensive IDL program into a widget program, and
>> I'd like to continue managing widgets while the other program is processing. Any
>> advice?
>
> If you have IDL 5.0+ then you can use the /NO_BLOCK keyword in
> your call to XMANAGER; this prevents IDL from "blocking" (where
> processing stops at the XMANAGER call) until the widget is destroyed.
>
> Your command line will reappear after starting this program, and
> new programs may be started and their events will be handled properly.
> This was one of the best features of IDL 5.0, IMHO.
>
> Look up NO_BLOCK keyword under XMANAGER in the Online Help.

I think there is some confusion here over asynchronous event
processing, which widget programs certainly allow, and
multi-tasking, which IDL doesn't do.

You can certainly have as many widget programs as you like
all running concurrently. IDL doesn't care which widget
program generates an event. As events are generated IDL puts
them into a queue to handle one after the other. If the code
for each event (i.e., the event handler code) is fast, then it
would appear that widget programs are running "concurrently".

But that is not the case at all. Consider this example.
Suppose the event handler code went into a processing loop
and that the loop took 5 minutes to execute. Then the user
could be pushing as many buttons on as many widget programs
as he or she liked, but nothing would be happening. In
fact, nothing at all would happen until that loop finished
and then, probably, all hell would beak loose as IDL rushed
to handle all the events that had queued over the past five
minutes.

IDL does only one "thing" at a time. It is not a multi-threaded
or multi-tasking program. This is exactly why you try to
avoid writing loops in event handler code. In fact, when you

need a loop, you try to take advantage of the widget program
itself *acting* like a loop. A widget animation is a perfect
example of this. If a widget animation was really in a loop,
there would be no way to interrupt the animation with, for
example, a Quit button. You can look at the example XMOVIE on
my web page for an example of how to use the program itself
to simulate a loop. You will see that each "event" actually
displays just a single frame of the animation sequence.
Between one event and the next a Quit button event, for
example, could be queued up and processed.

Cheers,

David

-----------------------------------------------------------
David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/