Subject: Re: Can Timer interrupt widget callbacks?
Posted by dg86 on Tue, 31 Mar 2015 02:32:01 GMT
View Forum Message <> Reply to Message

On Monday, March 30, 2015 at 6:13:55 AM UTC-4, David Grier wrote:
> On Monday, March 30, 2015 at 5:00:39 AM UTC-4, Helder wrote:
>> On Sunday, March 29, 2015 at 9:50:33 PM UTC+2, David Grier wrote:
>>> On Sunday, March 29, 2015 at 2:58:36 PM UTC-4, Helder wrote:
>>>> On Sunday, March 29, 2015 at 2:45:33 PM UTC+2, David Grier wrote:
>>>> > Dear Folks,
>>>> >
>>>> > A change in the behavior of Timer callbacks from IDL 8.3 to IDL 8.4 has broken one of my
>>>> > applications, and I could use some help in fixing it.
>>>> >
>>>> > Under IDL 8.3, the firing of an asynchronous Timer preempts widget callbacks.
>>>> > This behavior appears to have been reversed in IDL 8.4, so that callbacks now take precedence.
>>>> >
>>>> > Here's the issue ...
>>>> >
>>>> > My application has a widget_draw object that is supposed to update at regular intervals while
>>>> > the user drags graphics objects across the screen. I'm using Timer events to trigger the updates.
>>>> > Under IDL 8.3, the widget_draw's animation is smooth. Under IDL 8.4, the animation stops
>>>> > updating during drag events, which defeats the purpose of the animation.
>>>> >
>>>> > Is there any way to restore the old behavior so that firing a Timer interrupts a widget
>>>> > callback, perhaps as an option to Timer::Set()?
>>>> >
>>>> > If there's no way to make the Timer "dominant", does anyone have suggestions for modifying
>>>> > my widget callbacks so that they can check for pending timer events and handle them?
>>>> >
>>>> > All the best,
>>>> >
>>>> > David
>>>>
>>>> Hi David,
>>>> not sure if this is what you are looking for... could it be: !DEBUG_PROCESS_EVENTS = 0
>>>>
>>>> From the IDL help (http://www.exelisvis.com/docs/Whats_New_8_3.html):
>>>> Event handling while debugging
>>>> In the past, IDL would not sent widget events when you were stopped within a routine. Now, by default, IDL sends widget events even when stopped within a routine. This allows you to use graphics and widget applications while debugging.
>>>> There is a new system variable, !DEBUG_PROCESS_EVENTS, that can be set to 0 to

disable this behavior, or to 1 to enable this behavior. The default value is 1.
>>>>
>>>> I hope it helps.
>>>>
>>>> Cheers,
>>>> Helder
>>>
>>> Hi Helder,
>>>
>>> This is a great idea, but not apparently what's going on in my program. I tried setting
>>> !DEBUG_PROCESS_EVENTS = 0
>>> but did not see any change in performance. Under IDL 8.3 the program displays the animation
>>> smoothly, and under IDL 8.4, the animation pauses during drag events. This makes sense
>>> because !DEBUG_PROCESS_EVENTS was introduced in IDL 8.3 (where my program works)
>>> and does not seem to have changed in IDL 8.4 (where it doesn't).
>>>
>>> I'm guessing that the difference I'm seeing is due to a change in the behavior of asynchronous
>>> timer events because there's a notation to that effect in the Version History of the documentation
>>> for Timer.
>>>
>>> Even so, knowing about !DEBUG_PROCESS_EVENTS will be very useful for building widget programs
>>> in the future -- thanks for the pointer!
>>>
>>> All the best,
>>>
>>> David
>>
>> Hi David,
>> I was trying a quick answer to you question... I now realize, that !debug_process_events has nothing to do with your problem.
>>
>> However, I have a maybe the answer to "why" and a question:
>> Answer: timers have been modified "under the hood" according to this post:
https://groups.google.com/d/msg/comp.lang.idl-pvwave/NLy0qn0 JVVs/1smQwF_YPu0J
>> cite:
>> [* They no longer fire in the middle of system callbacks. For example,
>> they won't interrupt widget event handlers and object cleanup methods.
>> This is better since there are fewer nasty surprises.]
>> If you read further down, Doug explains in what conditions in 8.4 async timers will not interrupt the pro code in the widget event handling:
>> "async timer will not fire until the event handler finishes. As long as execution is in the "scope" of the handler, timers will not be processed."
>>

>> My guess: It seems that you're going to have to modify your code. Maybe you could pass on a variable to eventually fire the timer within the widget handling (as suggested by Doug).
>>
>> Question: I never used async timers, but I work a lot with draw widgets and mouse dragging operations. I normaly use widget events. *Why* don't you use those to update your video interface? Sorry if it's a stupid question, but I would have gone (out of ignorance) for the widget event handling and some widget timers.
>> Mondays are always good days to learn something new...
>>
>> Cheers and good luck with those timers...
>> Helder
>
> Thanks for finding the original discussion. I'd searched for something similar, but missed this
> thread. I still propose that the new Timer behavior ought to be controlled by keywords so that
> folks can decide whether or not timers should have precedence or other events.
>
> The animation in my code is driven by video cameras. Some cameras have APIs that raise interrupts
> when a frame is ready to be displayed. Others (including OpenCV) have to be polled. In IDL 8.3, I started using asynchronous timers to simulate interrupt-driven behavior for polled cameras. That way,
> I can use a single object class for all different types of video cameras, registering a callback that operates when an interrupt fires. The callback requests the application to redraw the screen.
> This still works fine for cameras with hardware interrupts, but not for those camera models whose interrupts are simulated with Timer events.
>
> Before IDL 8.3, my code used widget timers and polled the attached cameras from the "top down".
> The interrupt-driven code handles redraws from the "bottom up" and so adapts more naturally
> to the characteristics of different types of cameras. Swapping cameras on the fly automatically
> adjusts the frame rate, for instance.
>
> I might try writing a light-weight heartbeat DLM based on unix timers that would run in its own thread and provide the functionality I'm looking for -- sort of like Timer used to be!
>
> All the best,
>
> David

Dear Helder,

I took your advice and tried firing the timer during the time-consuming mouse-move event handler. It works in the sense that the animation appears to play smoothly, even if the timing is not really uniform. The price is that my camera objects have to provide a timer_id property, even if they don't use
timers. My interrupt-driven cameras (which don't use timers) provide a fake id, and the call to timer.fire(fake_id)
within the mouse-move event handler fails silently.

This is a nasty hack, but saves me having to sweat over a real solution, at least for the time being. The folks using my code can upgrade to IDL 8.4, and all is well with the world.

Thanks again,

David
