

Re: CPU performance rises when I'm opening an other application/dialog

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.directx.managed/2005-05/ms>

- From: "Zman" <zman@xxxxxxxxxxxxxxxx>
 - Date: Mon, 16 May 2005 18:52:30 -0700
-

The way the sample framework does it has changed several times and is about to change again for June according to Tom's blog. The move away from DoEvents was because of the large amounts of allocation and deallocation that are caused when you call that method multiple times per second.

For those of you who don't subscribe to my RSS feed I wrote up all of the discussions here <http://www.thezbuffer.com/articles/185.aspx>

"Robert Dunlop [MS MVP]" <rdunlop@xxxxxxxx> wrote in message [news:Oddex\\$hWFHA.3540@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:Oddex$hWFHA.3540@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

>I just took a look at the sample framework in the SDK for the managed >samples, and it doesn't have this problem. A couple of notes:

- >
- > * It does track whether it's the active application, and if it is not it > performs a 20ms Sleep operation each frame to give over some CPU time to > other apps.
- > * It allows for redraw through the paint event if the application is > paused, as I mentioned in my last post.
- > * The native PeekMessage/TranslateMessage/DispatchMessage functions are > used rather than DoEvents, I find it interesting that they chose this > approach.

>
> --

> Robert Dunlop
> The X-Zone
> <http://www.directxzone.com/>
> Microsoft DirectX MVP

> -----
> The opinions expressed in this message are my own personal views and do > not reflect the official views of the Microsoft Corporation.
> The MVP program does not constitute employment or contractual obligation > with Microsoft.

>
> "Robert Dunlop [MS MVP]" <rdunlop@xxxxxxxx> wrote in message > news:uYZo02hWFHA.3828@xxxxxxxxxxxxxxxxxxxxxxxxxxxx
>> "Michael Schwab" <MichaelSchwab@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in

Re: CPU performance rises when I'm opening an other application/dialog

```
>> message news:787E3F70-321F-4565-A303-2FCB0856CDFE@xxxxxxxxxxxxxxxxxxx
>>> Goodness those other "answers" were about as half baked as you can get.
>>> I've
>>> seen the exact same problem you're experiencing and have a templated
>>> solution
>>> I use on each app now. Consider the following game loop...
>>
>> Most of my work is in unmanaged C++, and I've not quite wrapped my head
>> around finding the optimized game loop yet for managed code, but I think
>> I have a few suggestions that might be of use...
>>
>>> while (form.Created)
>>> {
>>> if (form.WindowState != FormWindowState.Minimized)
>>> {
>>> //Update our game, and render to screen
>>> form.Render();
>>> Thread.Sleep(1);
>>> }
>>> else
>>> {
>>> Thread.Sleep(300);
>>> }
>>> Application.DoEvents(); //Let the OS handle what it needs to
>>> }
>>
>> Another useful thing to check is if the form is currently activated, by
>> trapping form activation and de-activation. This can allow you to stop
>> animation when the form is not in the foreground, and only call your
>> Render function when a Paint event is received. This allows the form to
>> properly redraw without taking up CPU time with ongoing animation when
>> another application or form has the focus. Another option would be to
>> use a longer Sleep interval while the form is not activated, to throttle
>> back the frame rate and give more processing time to other applications.
>>
>>> form is my directx window. form.Render() calls a bunch of objects that
>>> know
>>> how to render themselves. The interesting part is the Thread.Sleep(1)
>>> line.
>>> This little sleep period causes the app to release system resources and
>>> cut
>>> my "background mode" cpu utilization from +80% to < 1%.
>>
>> Another interesting use of Thread.Sleep() is to pass it a value of zero.
>> This allows the thread to yield to other pending threads, or returns
>> immediately if there are none pending. This can allow an un-throttled
>> rendering loop to gracefully concede time to other threads, and help
>> prevent pre-emptive context switches that can hurt your rendering
>> performance as well.
>>
>>> In a real app. You'd likely have a variable whose value changes based
```

Re: CPU performance rises when I'm opening an other application/dialog

>>> on
>>> the actual render time of your app, this would become your fps lock if
>>> you
>>> wanted to lock your fps to 60fps or whatever.
>>
>> Usually it's preferable to lock the presentation interval to the vertical
>> refresh. To prevent time from being taken up while Device.Present waits
>> for the device, use SwapChain.Present with the Present.DoNotWait flag and
>> call Sleep(0) while this function throws a WasStillDrawingException.
>>
>>> Either way, without the
>>> sleep, you're app will run as fast as possible and utilize as much cpu
>>> as is
>>> available (why this only happens when the app is in background mode,
>>> is still a mystery to me).
>>
>> That is an interesting pattern. A DirectX application can often utilize
>> high CPU when it's in the foreground, but if well mannered its usage
>> should drop when other applications take the foreground.
>>
>> I'd be curious to take a look at what's occurring here... do the managed
>> DX samples show this behavior?
>> --
>> Robert Dunlop
>> The X-Zone
>> <http://www.directxzone.com/>
>> Microsoft DirectX MVP
>> -----
>> The opinions expressed in this message are my own personal views and do
>> not reflect the official views of the Microsoft Corporation.
>> The MVP program does not constitute employment or contractual obligation
>> with Microsoft.
>>
>
>

• **Follow-Ups:**

- ◆ **Re: CPU performance rises when I'm opening an other application/dialog**
◇ From: Robert Dunlop [MS MVP]

• **References:**

- ◆ **CPU performance rises when I'm opening an other application/dialog**
◇ From: Wilfried Hafner
- ◆ **RE: CPU performance rises when I'm opening an other application/dialog**
◇ From: Michael Schwab
- ◆ **Re: CPU performance rises when I'm opening an other application/dialog**

Re: CPU performance rises when I'm opening an other application/dialog

Re: CPU performance rises when I'm opening an other application/dialog

◇ *From:* Robert Dunlop [MS MVP]

◆ ***Re: CPU performance rises when I'm opening an other application/dialog***

◇ *From:* Robert Dunlop [MS MVP]

- Prev by Date: ***Re: Surface is deprecated***
- Next by Date: ***Re: Sprite.Draw draws differently on different computers***
- Previous by thread: ***Re: CPU performance rises when I'm opening an other application/dialog***
- Next by thread: ***Re: CPU performance rises when I'm opening an other application/dialog***
- Index(es):
 - ◆ ***Date***
 - ◆ ***Thread***