

## Re: Refresh() and Invalidate()

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.drawing/2005-08/msg00100.html>

---

- *From:* SharpCoderMP <[csharp\\_mp@xxxxxxxxxxxxxxxxxxx](mailto:csharp_mp@xxxxxxxxxxxxxxxxxxx)>
  - *Date:* Thu, 11 Aug 2005 20:41:20 +0200
- 

i'll keep this in mind. thanks for your help.

Scott McChesney wrote:

Calling Refresh after an Invalidate call shouldn't cause your control to paint twice. I suppose it could happen, because you are technically at the mercy of the Windows message loop, and if you hit it just right... but I've never seen it. What I would expect to see is one Paint event (which means one re-paint), with the ClipRectangle encompassing the entire client area of your control. From MSDN's docs about Refresh:

"Forces the control to invalidate its client area and immediately redraw itself and any child controls."

So your control's entire client area would be invalidated, and since that would contain any region you passed to Invalidate, your Paint event would show the whole client area as invalid. Given what Refresh does, calling Invalidate and then immediately calling Refresh - regardless of whether you pass a region to Invalidate or not - is unnecessary; calling Refresh will get you there.

Note that Refresh also causes a redraw of all child controls, which is an important distinction from Invalidate (which only cares about the parent control.)

Whether you use the ClipRectangle or not, in my experience it's usually better to use Invalidate unless you absolutely need the immediate redraw that Refresh gives you. Both will end up in the Paint event - it's just that Refresh can (but not necessarily would) get you there faster. But if you use Invalidate, you'll potentially generate fewer Paint events. As I already mentioned, the less drawing you do, the better.

I will say that I rarely use Refresh. Usually, the only time I need something like that is if I have to re-draw the entire control on a size change, and that can be handled by setting

## Re: Refresh() and Invalidate()

the ResizeRedraw style flag.

- Scott

"SharpCoderMP" <csharp\_mp@xxxxxxxxxxxxxxxxxxxx> wrote in message [news:OfHMKvnnFHA.1416@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:OfHMKvnnFHA.1416@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

thanks for your reply. that cleared my mind a bit :)  
it looks that unless i use ClipRectangle in my drawing operations i do not need Invalidate(). what about calling Refresh() immediately after calling Invalidate(someArea)? will it force my control to refresh twice or just cause repainting without waiting for windows to take care of it \*and\* having my someArea sent with paint events? or maybe this is totally system dependend?

Scott McChesney wrote:

If you call Invalidate() without specifying a region, then yes, that is essentially equivalent to calling Refresh() - though, as you have already mentioned, Invalidate() does not cause an immediate re-paint. There are many situations where you \*don't\* want to re-draw the entire control - just a portion of it - and that's where Invalidate() comes in handy, since you can specify the region that has become invalid. That region is translated to a rectangle (if necessary), and that becomes the ClipRectangle that's provided within the Paint() event.

The reason that Invalidate() is not immediate is that you can call Invalidate() multiple times, specifying different regions, and .NET will combine all the calls into one rectangle which represents the entire invalid area in your control. That rectangle might be more than you actually specified - it depends on the regions that are invalidated - but it will contain

## Re: Refresh() and Invalidate()

everything that Invalidate() has received to date. It's that overall rectangle that's provided to the Paint() event. It's a performance enhancement, since you would receive less Paint() events than Invalidate() calls. Given that drawing is one of the most expensive operations in Windows, the less you do it, the better. I don't know the particulars of how many Invalidate() calls will be combined into one Paint() event, and I wouldn't be surprised if there are no hard-and-fast rules (i.e. if it's based on loop timing of when Windows gets around to firing off a Paint() event.)

Refresh() is conceptually the same as calling Invalidate() with no update region, and then forcing Windows to fire a Paint() event. It's designed for those situations where you have to update the UI and can't wait for Windows to get around to it.

Of course, all this is for nothing if you don't (or can't) take advantage of the ClipRectangle within your Paint() event. And by that, I mean you're not going to get the speed boost this process provides if you don't use the ClipRectangle to restrict the amount of drawing you do on your control. Of course, there are times when the particular drawing situation doesn't necessarily afford the use of the ClipRectangle - gradients and background bitmaps are a couple of situations that spring to mind. In those cases, if you're drawing is not speedy enough, you resort to double-buffering (which you probably should be doing anyway for any moderately-complex custom drawing.)

HTH

- Scott

"SharpCoderMP"

## Re: Refresh() and Invalidate()

<csharp\_mp@xxxxxxxxxxxxxxxxxxxx> wrote in  
message  
[news:O5fJlPnnFHA.3448@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:O5fJlPnnFHA.3448@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

two questions:

1. what's the difference with  
user drawn control? if i call  
Invalidate() without  
specifying region isn't that  
the same as Refresh()? what's  
the difference "inside" the  
framework? does this two  
methods are handled in a  
different way?

2. why Invalidate() is not  
immediate? i had few  
situations when i needed to  
force redrawing of a user  
drawn control and when i  
called Invalidate nothing  
happend. putting Refresh() in  
the same place instead of  
Invalidate() did the trick.  
what may be the reason?  
because of that i do not use  
Invalidate() at all, but  
lately when i came across bob  
powell site i noticed that he  
is using (probably) only  
Invalidate(). why?