Fastest API for rendering text in Windows Forms?

Asked 16 years, 3 months ago Modified 4 years, 1 month ago

Viewed 10k times

14

We need to optimize the text rendering for a C# Windows Forms application displaying a large number of small strings in an irregular grid. At any time there can be well over 5000 cells visible that update 4 times per second. The font family and size is consistent across the cells, though the color may vary from cell to cell, as will bold/italic/plain.

I've seen conflicting information on the web about `TextRenderer.DrawText` vs. `Graphics.DrawString` being the fastest/best, which reduces to a GDI vs. GDI+ comparison at the Win32 level.

I've also seen radically different results on Windows XP vs. Windows Vista, but my main target is Windows XP. Articles promising great advances under WinFX and DirectX 10 aren't helpful here :-)

What's the best approach here? I'm not afraid of introducing a small C++/CLI layer and optimizing device context handling to squeeze out more performance, but I'd like some definitive advice about which direction to take.

EDIT: Thanks for the initial responses. I'll be trying a combination of background bitmap rendering and sticking with the GDI equivalent calls.

c# windows performance

Share

Improve this question

Follow

edited Jan 31, 2010 at 1:34

Peter Mortensen

31.6k ● 22 ● 109 ● 133

asked Sep 16, 2008 at 11:27

Dave Moore

4,445 ● 9 ● 28 ● 34

6 Answers

Sorted by: Highest score (default)

6

A Microsoft developer has posted a GDI vs. GDI+ Text Rendering Performance article on his blog which answers the raw speed question: on his system, GDI DrawText was about 6 times faster than GDI+ DrawString.

If you need to be a real speed demon, TextOut is faster than DrawText, but you'll have to take care of clipping and word-wrapping yourself. ExtTextOut supports clipping.

GDI rendering (TextRenderer) will be more consistent with other parts of Windows using GDI; GDI+ tries to be device-independent and so some spacing and emboldening are inconsistent. See the SQL Server 2005

Surface Area Configuration tool for an example of inconsistent rendering.

Share Improve this answer

Follow

answered Sep 16, 2008 at 12:01

Mike Dimmick

9,792 ● 2 ● 26 ● 48

The sample app in the blog link is the one I used when I saw the big difference between Vista and XP - on my Vista PC, GDI and GDI+ were equal, while on XP I see the 6x difference the author mentions... This is probably a Vista driver issue, but highlights some of the difficulties here - thanks! – Dave Moore Sep 16, 2008 at 12:18

1 Historical note: ExtTextOut used to be the quickest way to draw a solid rectangle on some cards/drivers :) – Roger Lipscombe Jan 23, 2009 at 16:37

Unfortunately the article mentioned no longer seems to exist, here is a backup: blogs.msdn.com/cjacks/archive/2006/05/19/602021.aspx – NiKiZe Jan 8, 2020 at 15:01

Not even the backup exists anymore. Microsoft are frantic about leaving about old advice as they try to hoard and steer their developer base towards their intended "replacement APIs". Their preferred tool to do so is "obsolete" previous content as if Windows isn't de-facto backwards compatible. Anyway, here's the backup of the backup by a party that specifically doesn't remove content if they can avoid it: web.archive.org/web/20100128095845/http://blogs.msdn.com/cjacks/… – Armen Michaeli Mar 15, 2023 at 18:06

5000+ text rendering is slow even with GDI, especially if you need scrolling. Create a separate rendering thread

5

and notify the UI thread every 200 ms and bitblt the current results. It gives a smooth user experience.

Share Improve this answer

Follow

edited Jan 31, 2010 at 1:36

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Sep 16, 2008 at 12:00

null

3

GDI is faster at drawing in general that GDI+. I worked on a project that had to draw thousands of lines and text strings and switching from GDI+ to GDI made a significant performance improvement. That was using Windows XP so I cannot comment on Vista. I would also recommend using double buffering for your drawing to also improve performance. Create a compatible off screen bitmap and reuse that each time you need to draw.

Share Improve this answer

Follow

answered Sep 16, 2008 at 11:47

Phil Wright

22.9k ● 16 ● 85 ● 141

I agree double buffering helps. It went from 12ms for drawing just a bunch of string to 0ms. – tigrou Dec 5, 2023 at 22:11

On my Windows 7 64 Bit system TextOut is even a bit slower than DrawString! TextRenderer.DrawText is much

3

slower than DrawString.

Share Improve this answer

Follow

answered Dec 22, 2010 at 20:32

fritz

377 ● 1 ● 5 ● 11

1 I also discovered this...but I cannot find any reason why!? – Matthew Layton Jan 26, 2015 at 16:44

2

Creating a C++/CLI interop class to do the drawing in native code will result in crazy-fast drawing. We've witnesses this and measured it.

If you're not up to doing that, we've found graphics.DrawString is just slightly faster than than TextRenderer.DrawText.

Share Improve this answer

Follow

answered Sep 16, 2008 at 14:13

Judah Gabriel Himango

60k ● 39 ● 161 ● 215

0

From recent experience, fastest text output is achieved via `ExtTextOut` with `ETO_GLYPH_INDEX` flag. This comes at a price, and it's that you aren't printing characters anymore, but font glyphs directly. This means that you need to translate your regular character strings to glyph indexes strings prior calling `ExtTextOut`, either by calling `GetCharacterPlacement` everytime, or calling this function

just once to build your own translation table, that will be valid until a new font is selected in the DC. Remember that glyph indexes are 16bit, so you can store them in a Unicode string and call `ExtTextOutW` version regardless of original string character size.

Share Improve this answer

Follow

answered Nov 3, 2020 at 2:27

Chungalin

420 ● 4 ● 5