

# The story: Painting bitmap to screen is slow

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.drawing/2008-05/msg00038.html>

---

- *From:* "TomazK" <tomaz@xxxxxxxxxxx>
  - *Date:* Tue, 20 May 2008 11:10:25 +0200
- 

A little story about GDI+ performance.

I've done a lot graphics programming by now and the thing that bothers me with GDI+ in .NET framework the most is slow drawing of bitmaps to screen. I usually do this on the control's OnPaint event or drawing it directly to its graphics interface (using CreateGraphics).

I've heard a lot about GDI+ slow performance and mostly everybody say it's because it's not hardware accelerated. Now I don't want to talk about drawing shapes onto Graphics class but I will mainly talk about drawing bitmaps to screen which is the most essential part of any graphics library. What puzzles me is why is drawing bitmaps to screen so slow? Let me give you some numbers for example. All tests were performed using 32bit full-screen bitmaps and 32 bit desktop color depth on a WindowsXP. By saying full-screen, I mean drawing on the maximized window where bitmap size is exactly the same as client area of this window. I usually worked with resolution 1152x864 which is by these days low resolution.

In Win32, there were two types of bitmaps, compatible and memory bitmap (Memory bitmap can be a GDI DIB Section or a plain simple memory buffer which has nothing to do with GDI). Drawing a compatible bitmap to screen was of course super fast as bitmap was already located inside graphics card memory. This easily gave 1000+ fps and was indeed hardware accelerated. With memory bitmap located in system RAM, drawing to screen involved copying bitmap's data from RAM to video card which was a lot slower but still very fast though it was very much dependent not so much of the graphics card, but of the interface speed (AGP 2x, 4x, 8x, PCIe), motherboard and probably the processor. PCIe gives a real speed boost here. This gave me i.e. 100fps on AGP8x, and after I switched to a much better motherboard (all other components are the same), it gave me 300fps. This was not hardware accelerated as HW acceleration can only be done inside graphics card. This fps is more than enough for any desktop graphics performance.

Now let's come to .NET framework and GDI+ behind it. The bitmap for testing was constructed using a main

## The story: Painting bitmap to screen is slow

form's graphics class to make bitmap as compatible (regarding color depth) with screen as possible. Drawing such bitmap was much slower. It was some three times slower than drawing memory bitmap in using GDI. I usually got some 100fps (with previously mentioned new motherboard) which is still not bad. Now the best part. When I switched to Vista (with or without SP1, it doesn't really matter cause it's the same), the drawing was even worse. I don't know the exact number but it was even up to 50% slower. That is all on low resolutions.

The conclusion. How can a person make a descent graphics library for desktop apps, when the most basic operation of any such library is very slow, that is drawing all that beautiful contents onto screen for people to see. Again, "experts" will tell you that's because GDI+ is not hardware accelerated. But I don't believe this because for comparison, how can GDI draw memory bitmaps so fast. Why can't GDI+ do the same because it probably uses the same bitmaps or memory buffers at least. Is GDI+ doing something extra while drawing bitmaps? What and how to make it stop doing that. It just doesn't make any sense. Switching to other GDI+ bitmap's pixel formats makes drawing even slower (PARGB included).

At the end, let me show you the performance on Vista (SP1) and on a huge 24" monitor. Machine configuration: Intel Core2Duo 2.13GHz, 2GB RAM, 32bit desktop of size 1920x1200, NVIDIA Quadro NVS 290 Graphics card. We all agree that my machine is not a slow one and it's probably more (or at least equal) than the machine average user has these days. The drawing speed is astonishingly 12fps :). Now imagine how slow it will become when I start drawing something onto this bitmap...

best regards

TomazK

.