

Re: 2D graphics & speed quest

Source:

<http://www.tech-archive.net/Archive/VB/microsoft.public.vb.general.discussion/2006-04/msg02026.html>

- *From:* "Mike Williams" <Mike@xxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Sat, 22 Apr 2006 15:38:15 +0100
-

"Mike D Sutton" <EDais@xxxxxxxx> wrote in message
[news:%23jPg1\\$eZGHA.428@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:%23jPg1$eZGHA.428@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

The display is being invalidated and refreshed at just the same rate

I was talking about being able to see the "currently drawn lines", which of course doesn't happen if the drawing is going into an offscreen buffer, such as was the case when I used Autoredraw. Besides, I really didn't expect to see any individual updates for PolyLine to draw just 1000 lines, because I expected it to be completed too quickly to visibly identify the different stages of the drawing, but I could definitely see them plain as day, whereas I couldn't see any at all for the LineTo drawing. Obviously I was seeing them simply because PolyLine was taking an unexpected amount of time, about 14,000 milliseconds on my machine for the 100 iterations of the NumDraw loop, or about 140 milliseconds for the first loop (which of course is the only one I would see being "built up"). That's about 10 frames on my 75 Hz display, so of course I could see that happening. In contrast, the LineTo loop took only 6,000 milliseconds, or 60 milliseconds for the first loop. That of course is about 4 frames, which I wasn't quick enough to see (and of course I couldn't see those at all unless I ran the LineTo loop first, or on its own. Incidentally, those figures (14,000 and 6,000) are quite abysmal compared to your own (3,200 and 1,700), even though I'm running a fairly fast processor and a fairly powerful Radeon 9800 graphics card. Mind you, the processor in an AMD (3000+) which really only runs at about 2.2 Ghz and the 9800 is the slightly crippled SE version :-)

if anything PolyLine() should have the upper hand since it could wait until it's finished drawing until it invalidates the display but it doesn't appear to be the case.

It certainly didn't have the upper hand on my machine, and apparently also on yours. However, I just happened to be running your code with a Command Button on the Form, and it turns out that the "hole in the clipping region" was slowing things down by a massive amount, certainly much more than I would have expected. Simply getting rid of the Command Button caused the same code to run at an amazingly different speed, and then PolyLine most definitely did have the upper hand. These are the timings I am currently getting for the full 100 iterations of the "1000 drawn lines":

Standard non-Autoredraw Form with
a Command Button in the middle:

Re: 2D graphics & speed quest

PolyLine: 17300 milliseconds
LineTo: 7060 milliseconds

Exactly the same code on exactly the same
Form but with the Command Button removed:
PolyLine: 2 milliseconds !
LineTo: 206 milliseconds

Both PolyLine and LineTo gain a lot of speed by simply getting rid of the Command Button (obviously because there is no longer a "hole in the clipping region"), but the difference in speed of PolyLine is dramatic! Removing the Command Button from the Form causes the PolyLine code run runs over eight thousand times faster!

These are the timings I'm getting with AutoRedraw set to True:

AutoRedraw Form with
a Command Button in the middle:
PolyLine: 2 milliseconds
LineTo: 220 milliseconds

Exactly the same code on exactly the same AutoRedraw
Form but with the Command Button removed:
PolyLine: 2 milliseconds
LineTo: 218 milliseconds

As might be expected from the first set of results, AutoRedraw gave approximately the same timings as the "non autoredraw Form" with the Command button removed, and the timings were the same with or without the Command Button. This if course was because the autoredraw DC didn't have "a hole in the clipping region" whether the Command Button was present on the Form or not.

Further tests proved just as interesting. Drawing to only about 90 per cent of the Form and with the Command Button still in the middle made no difference of course in the "non autoredraw test with the Command Button" and PolyLine still took approximately the same 17000 milliseconds, with LineTo taking about the same 7000 milliseconds. However, progressively moving the Command button nearer and nearer the edge of the "drawn area" and retesting caused PolyLine to remain fixed at about 17000 milliseconds, but for LineTo to gradually get faster and faster, going down to about 4000, 3000, 2000, 1000 and finally 210 milliseconds as the Command Button was moved "nearer the edge". PolyLine however remained solid at 17000 milliseconds no matter where the Command Button was, even if just a tiny corner of the Command Button was covering just a single one of the "thousand points". However, as soon as the Command Button was moved just a couple of pixels further, so that it was covering none of the thousand points, the PolyLine timing immediately shot right down from 17000 milliseconds to just 2 milliseconds. Obviously, each of the individual LineTo statements (in the LineTo code) was checking whether that specific line was partially underneath the Command Button, and those that weren't were drawn quickly while those that were under it were drawn slowly. In the case of PolyLine though, if just a single one of the thousand points was under the Command Button then every line in the set was drawn slowly. It makes sense I suppose when you think about it, but it was totally unexpected nonetheless.

Oddly enough, enabling AutoRedraw only seems to slow down
LineTo() here while providing no performance increase to PolyLine():
PolyLine(): 3182.70

Re: 2D graphics & speed quest

LineTo(): 2320.58

In view of the timings I'm getting that seems very odd indeed. Methinks this needs a bit more investigation. I'll try it on my laptop later.

Mike