Subject: BufferPainter now MT optimized
Posted by mirek on Tue, 26 Dec 2017 10:52:30 GMT
View Forum Message <> Reply to Message

This one was hard... but Painter now has 'Co' method which actually works. Gains are not spectacular, with my 4/8 i7 CPU, the basic 'U++ Painter' example in PainterExamples is like 2.3 times faster in MT. In some rare cases (e.g. when there is a lot of raster data, like radial span), speedup can be >4, in some other rare cases (a lot of small polygons), MT can even be bit slower than ST.

Important: When Co is active, it is mandatory to invoke 'Finish' method to finalize painting before moving result from ImageBuffer. BufferPainter destructor does this, so either add a {} block or call Finish explicitly.

Subject: Re: BufferPainter now MT optimized
Posted by Tom1 on Fri, 29 Dec 2017 12:07:40 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Sorry to inform you, but there is something wrong with ::Co when drawing solid strokes of longer polylines. They come out as narrow center line and then some filled polygon in some part of the edge line.

I'm really busy at the moment, but I'll try to get some pictures of the phenomenon in the coming days, next week likely.

Best regards,

Tom

Subject: Re: BufferPainter now MT optimized
Posted by Tom1 on Fri, 29 Dec 2017 15:00:46 GMT
View Forum Message <> Reply to Message

Hi,

This is sooner than next week... :)

The attached images show a small vector map of an island rendered with ::Co(true) in MT-Painter.png and ::Co(false) in ST-Painter.png from the same location at the same zoom level. In this case the entire contour line became a full polygon instead of a stroke. In many cases, only part of a polyline becomes a polygon. The rest of the stroke is then represented by a very thin center line of the intended stroke. In those cases the closing edge of the polygon seems always to be horizontal.

Best Regards,

Tom

## File Attachments

1) MT-PainterIssue.zip, downloaded 356 times

---

Subject: Re: BufferPainter now MT optimized
Posted by mirek on Fri, 29 Dec 2017 16:39:26 GMT
View Forum Message <> Reply to Message

Any chance for extracting a testcase for this?

Or maybe just a hint how the painting code looks like... Painter Co works by storing multiple paths into buffer, if they are solid color only, then rasterizing these multiple paths in parallel, then rendering resulting lines of resulting raster in parallel. If it is nor a solid color polygon/stroke, it is rasterized ST, then lines rendered in parallel.

What could got wrong is that perhaps the storing multiple paths has some problems. That would be probably caused by some combination of Fill / Stroke that I do not account for correctly...

Mirek

---

Subject: Re: BufferPainter now MT optimized
Posted by Tom1 on Tue, 02 Jan 2018 14:34:44 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Sorry for the alarm. This is likely my own fault in one way or another:

While attempting to create a testcase, I discovered that my complex polyline optimization routines might end up skipping the stroking of polylines, e.g. if the lines are too narrow to display at specific zoom levels. This in turn leaves the entered vertices (Move()/Line()) in the Painter unused and probably cause trouble afterwards in the MT optimized code while it is fine with ST code. I was able to fix the MT rendering symptoms by simply enclosing the my polyline code within a Begin()/End() pair in the BufferPainter.

Thanks and have a nice 2018!

Best regards,

Tom

Subject: Re: BufferPainter now MT optimized
Posted by mirek on Tue, 02 Jan 2018 15:51:37 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 02 January 2018 15:34Hi Mirek,

Sorry for the alarm. This is likely my own fault in one way or another:

While attempting to create a testcase, I discovered that my complex polyline optimization routines might end up skipping the stroking of polylines, e.g. if the lines are too narrow to display at specific zoom levels. This in turn leaves the entered vertices (Move()/Line()) in the Painter unused and probably cause trouble afterwards in the MT optimized code while it is fine with ST code. I was able to fix the MT rendering symptoms by simply enclosing the my polyline code within a Begin()/End() pair in the BufferPainter.

Thanks and have a nice 2018!

Best regards,

Tom

Well, maybe worth fixing anyway. Can you try a testcase of such unused Move/Line?

Mirek

Subject: Re: BufferPainter now MT optimized
Posted by Tom1 on Wed, 03 Jan 2018 08:04:38 GMT
View Forum Message <> Reply to Message

Hi Mirek,

I found out that PreClip(true/false); is involved here. (To optimize rendering performance in my code, I enable PreClip only for rendering dashed strokes.)

Please find attached the MTPainterIssue testcase package. It is supposed to draw four sine waves, half of which are solid and the other half dashed. As it turns out, with BufferPainter::Co the solid strokes become fills instead of lines. The package is delivered here in its faulty state with painter.Co(true); please set to false to see the correct rendering.

The way to fix this problem in the MTPainterIssue testcase package is to switch the ordering of PreClip(true)/Begin() and End()/PreClip(false) pairs for dashed lines. This is also the appropriate fix for my actual code.

Best regards,

Tom

Update: Added the missing attachment...

## File Attachments

1) MTPainterIssue.7z, downloaded 334 times

---

Subject: Re: BufferPainter now MT optimized
Posted by Tom1 on Sat, 06 Jan 2018 10:33:38 GMT
View Forum Message <> Reply to Message

Hi Mirek,

I have started to think if swithing PreClip() on-the-fly is Co() compatible at all. In some rare occasions it seems to cause the issue even when properly bounded between a Begin/End pair.

I think there might be too little parallelism (content) in my testcase to ever face this issue after reordering the PreClip and Begin/End calls.

Best regards,

Tom

---

Subject: Re: BufferPainter now MT optimized
Posted by mirek on Sat, 06 Jan 2018 12:14:39 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sat, 06 January 2018 11:33Hi Mirek,

I have started to think if swithing PreClip() on-the-fly is Co() compatible at all. In some rare occasions it seems to cause the issue even when properly bounded between a Begin/End pair.

I think there might be too little parallelism (content) in my testcase to ever face this issue after reordering the PreClip and Begin/End calls.

Best regards,

Tom

Actually, PreClip originally was not meant to be switched on-the-fly at all... :)

It is not attribute, just the setting of whole BufferPainter, sort of like Co or image dimensions.

Mirek

---

Subject: Re: BufferPainter now MT optimized
Posted by Tom1 on Sat, 06 Jan 2018 14:45:44 GMT
View Forum Message <> Reply to Message

Quote:
Actually, PreClip originally was not meant to be switched on-the-fly at all... Smile

Well, it's just that using PreClip when rendering an average chart, adds a cost of about 5-15%. The case when worse comes to worst (a long dashed line with small fragment only visible), it makes a world of difference shrinking seconds of rendering time to milliseconds instead. This is why I like to turn PreClip on for rendering dashed strokes only and disable it in every other case.

Do you think if it would be possible to condition the PreClip functionality to dashed strokes only internally in BufferPainter to optimize performance for both dashed and solid strokes?

BTW; I have found that on my development system (Core i7, 4 Cores plus hyper threading) Co(true) improves chart rendering performance by about 15-20% on the average. That's nice already, but raster images render at about 3x speed! Not bad :)

Best regards,

Tom

---

Subject: Re: BufferPainter now MT optimized
Posted by mirek on Sat, 06 Jan 2018 19:35:39 GMT
View Forum Message <> Reply to Message

I have just found out that Preclip had bug that affected performance hugely. Fixed in trunk.

The idea of applying preclip only if dashed lines are active is good one!

Mirek

---

Subject: Re: BufferPainter now MT optimized
Posted by mirek on Sat, 06 Jan 2018 20:16:50 GMT
View Forum Message <> Reply to Message

BufferPainter::PreClipDashed

Please try...

Also: Currently, "End" limits polygons that can be rasterized in parallel. This is mostly "just to be sure" thing - there are some cases where this is necessarry and I was to scared to try to identify them precisely.

Current model is:

Polygons (or lines) filled with single color can be rasterized in parallel, then filled in parallel. End limits the group of such polygons.

More complex fills are rasterized in single thread, then filled in parallel.

(Rasterized means more or less "converted from lines to pixels").

If you think it is helpful for you code to remove End boundary, I can try...

---

Subject: Re: BufferPainter now MT optimized
Posted by Tom1 on Sat, 06 Jan 2018 23:06:13 GMT
View Forum Message <> Reply to Message

Hi,

Thanks Mirek! This sounds excellent. I will try this out as soon as I can, on Monday I hope.

I'm not sure I fully understand the current logic for using Begin/End. I have previously used it like PUSH/POP for the context of rotations or transformations in general. Additionally, I have similarly enveloped the polylines which had dashing enabled. If I get it right, I do not need to use it for polylines, polygons or dashing anymore. Right? (All my vector objects are solid single color, no gradient or image fills there.)

What exactly are you referring to by the possibility to try to remove End boundary? Does it mean you could possibly make it even more parallel than described in your message?

Anyway, I will test this and see how well it works and how much faster it is. I will also try if I can do the dashed polylines/polygons without Begin/End pairs.

Thanks and best regards,

Tom

---

Subject: Re: BufferPainter now MT optimized
Posted by mirek on Sat, 06 Jan 2018 23:24:40 GMT

Tom1 wrote on Sun, 07 January 2018 00:06What exactly are you referring to by the possibility to try to remove End boundary? Does it mean you could possibly make it even more parallel than described in your message?

The End boundary is there mostly because of possible clipping changes - it is hard for me to run parallel rasterizers when there are more clippings active.

If I could detect that before and after End the same clipping is active, I could run both sides (of End) in parallel.

Subject: Re: BufferPainter now MT optimized
Posted by Tom1 on Mon, 08 Jan 2018 09:37:13 GMT


Hi Mirek,

OK, now I've tested with a very long dashed line only partially visible in the view. The performance improvement with PreClip() and PreClipDashed() is (obviously) similar: 2.5x speed compared to running without any PreClip. Using PreClipDashed(), instead of on-the-fly switching of PreClip(), removes any rendering issues thereof. So, I suggest obsoleting the old PreClip() altogether. PreClipDashed() does it all.

Does this even need to be selectable anymore? There does not seem to be any performance penalty for ordinary (solid) strokes.

I still had to use Begin()/End() to enclose dashed strokes. Otherwise, some solid strokes came out as dashed. Then I noticed that the Begin()/End() pair was (an expensive) way to put the stroking back to solid mode after drawing dashed strokes. Calling Dash(""); before every solid stroke removed the need for such usage of Begin()/End(). Is Dash(""); the cheapest way to return to solid stroking?

Please note: Getting rid of extra Begin()/End() pairs nicely improved overall rendering performance by some 5-10 %. :)

Thanks and best regards,

Tom

Update: After further consideration, I decided to do all dashed strokes by the sequence of "painter.Dash(x); painter.Stroke(); painter.Dash("");". This keeps the default line style solid. It also lowers the number of calls to Dash(); because most of the strokes are solid anyway.