Subject: Font and Image rendering slow
Posted by devilsclaw on Wed, 28 Jun 2023 15:36:11 GMT
View Forum Message <> Reply to Message

So I have been porting a app that I created in java to U++.

When I render a image with DrawImage and have 108 instances of it on the screen it is way slower then in java. So I decided to manually draw what the image is with line and rectangle since its a simple image and its performance was on par with java.

The item I am drawing also has text and with the text enable it renders much slower than java also.

So it seems that both the text and image rasterizer are highly inefficient at least compared to java.

Subject: Re: Font and Image rendering slow
Posted by jjacksonRIAB on Thu, 29 Jun 2023 02:46:29 GMT
View Forum Message <> Reply to Message

Instead of creating 108 copies of the same image did you try just drawing the same image 108 times at different positions? Also did you try caching that Image do it's not re-rendered every call to Paint?

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Thu, 29 Jun 2023 02:57:57 GMT
View Forum Message <> Reply to Message

Currently the image is an element of the the object and is in the paint member of that object, which I currently no longer use the image and just manually draw what the image looks like during the paint which has increase performance greatly. The being said I don't know how to cache an image so that would be useful in the future. so if you could explain how or give an example.

But that does not solve the Font problem since that is different for the 108 items and it has a similar slow down.

If I click and hold on the object it allows me to move it on the screen which of course causes a repaint/refresh and its performance it much lower than the java version.

Subject: Re: Font and Image rendering slow
Posted by Oblivion on Thu, 29 Jun 2023 05:43:40 GMT
View Forum Message <> Reply to Message

Hello devilsclaw

Quote:Currently the image is an element of the the object and is in the paint member of that object, which I currently no longer use the image and just manually draw what the image looks like during the paint which has increase performance greatly. The being said I don't know how to cache an image so that would be useful in the future. so if you could explain how or give an example.

But that does not solve the Font problem since that is different for the 108 items and it has a similar slow down.

If I click and hold on the object it allows me to move it on the screen which of course causes a repaint/refresh and its performance it much lower than the java version.

Image tutorials (check 6 for image caching)

If you can provide a bare minimum U++ example code (preferably in Java too) that replicates the problem we can look into it.

Also platform/OS/hardware information and specs would be very useful.

Best regards,
Oblivion.

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Thu, 29 Jun 2023 20:52:42 GMT
View Forum Message <> Reply to Message

My work computer is old now built somewhere in like 2010.

GRAPHICS CARD: NVIDIA Geforce GTX 780 3GB
MONITORS: 3x resolution of 1920x1080
RAM: 32Gigs
CPU: i7-4930K CPU @ 3.40GHz
CORE: 6
THREADS: 12

I am running Ubuntu 20.04 most of the time and to test in windows a VM with Windows 10 installed.

I have the java version done. Are you wanting the source code or just the built binary

As for the U++ I will start working on it now.

Subject: Re: Font and Image rendering slow
Posted by Oblivion on Thu, 29 Jun 2023 21:25:22 GMT
View Forum Message <> Reply to Message

Quote:Are you wanting the source code or just the built binary

Minimal source code for replicating the problem will be sufficient.

Best regards,
Oblivion

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Thu, 29 Jun 2023 21:28:42 GMT
View Forum Message <> Reply to Message

The java project is too big to upload here since it has everything except the JRE and JDK to build it.

So here is a link:
http://devilsclaws.net/downloads/javademo.zip

in the fatjar directory there is either the jar file or a window exe.

This is a netbeans project so if you want to compile it with that you can I also have script directories that let you build but you will still need to open it in netbeans at least once on your system so it can get the java pathing correct.

You can scroll with the mouse wheel and shift+scroll does the other scroll axis.

The icons are clipped to only drawing what is visible also but there is like 1089 total there. only about 108 on screen at a time on a 1920x1080 resolution.

I will be working on the U++ version tomorrow.

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Thu, 29 Jun 2023 21:35:53 GMT
View Forum Message <> Reply to Message

Also you can click and drag the icons around which is where the slow down on U++ vs Java so once I get the U++ version done it will be able to replicate the performance stuff I am talking about.

Subject: Re: Font and Image rendering slow

Posted by devilsclaw on Sat, 01 Jul 2023 18:46:24 GMT
View Forum Message <> Reply to Message

Here is the U++ example

Also The icon is being manually drawn but it can be switch to using the image instead by change the #if 0 to #if 0 in icon_t.h in function icon_paint_icon

This demo also clips and only draws what it visible on screen so making the window lager will draw more and impact the performance more. The default size does not show much difference on my system but when I maximize the window it then shows for me.

there is also a render_everything flag in the frm_main.h which if set to true will even render off screen if needed to see the effect. same with the java version

File Attachments

1) upp_demo.zip, downloaded 167 times

Subject: Re: Font and Image rendering slow
Posted by Oblivion on Sat, 01 Jul 2023 19:40:05 GMT
View Forum Message <> Reply to Message

Hi,

Thanks for the code.
Yesterday I have looked into the Java code and it seems fine and works well.

A quick tour through its U++ counterpart (I will dive deep, later), and here's some "odd" things (for U++ land, of course) I've seen so far (All understandable, as Upp can sometimes look alien even to regular C++ developers):

1) You seem to have duplicated code that is already in Upp: (Rectf, Sizef, Pointf, their operators, etc.)
2) Mixing std::string with Upp code: Not inherently bad, it is possible and sometimes necessary but Upp::String is more optimized and has a lot of optimized functionality you can utilize. Rule of thumb: Use Upp::String with Upp infrastructure whenever possible. Convert to Std only for glue code.

3) You are using BufferPainter, which is a pdf/print quality (subpixel) 2d graphics library for U++. It is great for a lot of things, but it is also very CPU-intesinve.It does not (AFAIK) utilize GPU's 2D hardware acceleration. But it can take advantage of multithreading. Try the below code, and you'll see the difference:

4) (post-post:) You also seem to use iterators a lot. Avoid them where possible. Upp's containers are designed to work with indices. In a wide range of use-cases they are lot faster & easier to use.

```
void frm_main::Paint(Draw& w) {
  Size sz = GetSize();
  ImageBuffer ib(scroll.GetViewSize());
  BufferPainter sw(ib, MODE_ANTIALIASED);

  sw.Co(); // Enable MT.
  sw.Clear(White());
  _icons_paint(sw, scroll.Get().x, scroll.Get().y, render_everything);
  sw.Finish(); // Join.

  w.DrawImage(0, 0, ib);
}
```

Now, of course the above code doesn't really fix the problem.
I am going study your code and try to come up with a diagnosis at least.

Best regards,
Oblivion

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Sat, 01 Jul 2023 19:49:35 GMT
View Forum Message <> Reply to Message

The replicated code is actually to make the code work more like java and I also have a hard time with cx and cy as width and height.

I also have more standard C++ code in the project then U++ since I am only using U++ for the UI elements.

I will look to see where the code only needs U++ and where C++ STD is needed and try to optimize it.

One of the reasons I used that Draw function is due to sub-pixel bleeding looking bad in integer based drawing. So I use sub pixels to make it look clean.

indices I think I might do that for code standardization to make the code work similar in all areas and maybe for deleting or its just me being used to C++ STD since it does not make it easy to delete element with out iterators. I will look to see what I can do to clean that up.

I will try you example today. Thanks.

Subject: Re: Font and Image rendering slow

Posted by devilsclaw on Sat, 01 Jul 2023 20:39:11 GMT
View Forum Message <> Reply to Message

The multi threading did make a difference still not on par with java but about a 2x increase from the feel of it.

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Sat, 01 Jul 2023 20:45:54 GMT
View Forum Message <> Reply to Message

In the past when I was dealing with some other graphics issues I believe I saw that that font is rendered with a bunch of little lines that could make up lets just say 100 per letter. One of the things I was thinking I could due is pre render all the letters and then hold them in an array with the character as the index. this would prevent the need to fully render each letter every time it draws.

Subject: Re: Font and Image rendering slow
Posted by Oblivion on Sat, 01 Jul 2023 21:29:39 GMT
View Forum Message <> Reply to Message

I've just made some simple modifications to the code (to give you an idea. Please check the attached code.

I used Draw, directly. and didn't bother fixing the rectangles part. But you can also draw the whole "sprite" (icon) into a image, then add text to it and then blit it to the draw, you'll likely get the same result.

Best regards,
Oblivion

File Attachments
1) upp_demo.zip, downloaded 185 times

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Mon, 03 Jul 2023 15:06:45 GMT
View Forum Message <> Reply to Message

You mentioned blitting but I am not sure how to blit in U++ there does not seem to be a specific function to blit so I was wonder how one might do it.

Thanks

EDIT: fixed weirdness in the statement

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Mon, 03 Jul 2023 18:35:29 GMT
View Forum Message <> Reply to Message

The code example you gave me definitely was a big help. I also notice that this version of Draw seems to write to the sub pixel position so that the lines always look clean which is awesome.

I had to rewrite the rectangle code to use lines and then fill. I also had to add in the ability to do dashed outline rectangles.

Now everything is nearly on par with java. mouse movement seem that same but the number of trailing icons which moving the icon around is about half as much which seems like refresh rate.

Just going to say that is good enough for what I need. I think I might add a FPS counter just do I can see if there is any diff in that relation or something else. just to see.

Subject: Re: Font and Image rendering slow
Posted by Oblivion on Mon, 03 Jul 2023 18:48:27 GMT
View Forum Message <> Reply to Message

Happy to help.

By the way, your drawing code can be easily further optimized. For one, I've noticed that you are refreshing the whole "scene".
Instead, you narrow down the refreshed areat to the dirty parts. There is an Draw::IsPainting method for that. Check that out.
Also, Draw::DrawLine function can draw dashes and dots (it accepts certain patterns. You can use it to draw the rects.

If you need more help, I'll be around.

Best regards,
Oblivion

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Mon, 03 Jul 2023 19:18:39 GMT
View Forum Message <> Reply to Message

I think what I am seeing is the mouse polling right vs java.

In java I will see up to 113 fps when dragging an icon around but in U++ is caps at 27.

But what is interesting is that if I make the window expand to my three screens it stays about the same where java drops from a max of 113 to 67.

I could be wrong about the pulling rate not sure.

So how would I use the IsPainting is there a good example somewhere ?

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Mon, 03 Jul 2023 19:35:04 GMT
View Forum Message <> Reply to Message

Nope it is actual FPS.

I am currently having in both java and U++ do a refresh/repaint at the end of the paint code so I can track the FPS and AVG FPS and java is in the 120fps range and the U++ is 25.

So I definitively should find a way to optimize this.

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Mon, 03 Jul 2023 20:31:34 GMT
View Forum Message <> Reply to Message

With a quick test of drawing nothing except the counter the max I get is 61 FPS. There seems to be a fps cap somewhere also.

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Mon, 03 Jul 2023 20:51:04 GMT
View Forum Message <> Reply to Message

Seems in linux GTK will force the FPS to max out at the monitors Hz rate. so fps is max at 60 for me right now. So if I am able to optimize it I should be able to get 2x at max fps.

Subject: Re: Font and Image rendering slow
Posted by Oblivion on Mon, 03 Jul 2023 20:55:00 GMT
View Forum Message <> Reply to Message

One cap is in the EventLoop(). By default it lets the gui sleep for 20 ms (in order to process (key, mouse, etc.) events.);
You can override the Run() method and implement your own. The other cap is in the backends. Compiling the code in NOGTK (X11) mode lets the app jumpt to 90+ fps on my machine (ryzen 5 5600g (no discrete gfx card), 16 GB ram, linux 6.3.9, GNOME 44.1)

Overriden Run example:

```
void frm_main::Run()
{
 OpenMain();
 while(IsOpen()) {
  Ctrl::ProcessEvents();
  Sleep(1); // can be dynamically set to adapt to workload...
 }
}
```

Best regards,
Oblivion

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Mon, 03 Jul 2023 20:58:38 GMT
View Forum Message <> Reply to Message

I found adding my only clipping back in pushed it up to 61 FPS even spanning across all through monitors so the clip done by Draw still takes time to do even if its not trying to render it to screen.

```
void frm_main::_icons_paint2(Draw& w, int x_offset, int y_offset, bool _render_everything)
{
  rect_f visible = rect_f(scroll.x, scroll.y, GetSize().cx, GetSize().cy);
  for(auto it = icons_zorder.rbegin(); it != icons_zorder.rend() ; it++) {
    icon_t* icon = *it;
    rect_f ricon;

    ricon = icon->get_bounds();

    if(_render_everything || ricon.intersects(visible)) {
      w.Clip(GetSize());
      icon->icon_paint2(w, x_offset, y_offset);
      w.End();
    }
  }
}
```

Subject: Re: Font and Image rendering slow
Posted by Oblivion on Mon, 03 Jul 2023 21:00:22 GMT
View Forum Message <> Reply to Message

As for the Draw::IsPainting(),

First you select the area to refresh, using the Refrest(Rect) method, not the Refresh() method. Then, in the Paint() method, you check for the rectangle, using the Draw::IsPainting() method, and if it is true, only then you paint.

Two -different- examples, using this method to boost the speed:

1) see: CtrlLib/LineEdit.cpp,ln. 468
2) see: UppHub/TerminalCtrl/Renderer.cpp, ln. 196

Best regards,
Oblivion

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Wed, 05 Jul 2023 18:40:58 GMT
View Forum Message <> Reply to Message

So here is I believe is my last problem. I was also using BufferPainter for scaling. Is there an alternative way to handle it ?

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Wed, 05 Jul 2023 18:53:33 GMT
View Forum Message <> Reply to Message

Actually I found another problem. I was also using RGBA for transparency effects. Is there a way to do that with out BufferPainter as well

---

Subject: Re: Font and Image rendering slow
Posted by Oblivion on Wed, 05 Jul 2023 21:18:21 GMT
View Forum Message <> Reply to Message

For such operations, you can directly use ImageBuffer & Image utility functions. They are cheaper.

Best regards,
Oblivion

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Thu, 06 Jul 2023 15:16:34 GMT

View Forum Message <> Reply to Message

Thanks, I do want to point out that most base operating systems drawing functions support opacity setting of dialog components. The Gtk for linux is cairo_set_source_rgba, in windows you can use alpha blending https://www.codeproject.com/Articles/286/Using-the-AlphaBlend-function. I did not check on CoCo though. I did a simple test with cairo_set_source_rgba and it does work. I think in the long run it might be worth setting up the SystemDraw functions to support alpha / opacity control, but that just my opinion.

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Thu, 06 Jul 2023 18:44:30 GMT
View Forum Message <> Reply to Message

I have been looking at what you have suggested but I don't see how it can help me. If we are drawing directly to Paint(Upp::Draw& w) which is where we get the speed boost, but then we switch back to ImageBuffer which does not have any function its self to draw lines and rectangle and other things. Then I looked at the Image utility functions and I also don't see any functions that allow drawing lines I only see rectangles, so I am not seeing anyway to do this with out some sort of Painter.

I have tried PaintingPainter, BufferPainter, ImagePainter, DrawingDraw and then used PaintImageBuffer to get that into the ImageBuffer and it is all slow, like 5 FPS.

So I am not sure how to do everything I need. The main Application Has Rectangles, Rectangles with Dashes, Lines, Lines with Dashes, Images, an overlay with transparency and scaling when zoomed in and out with ctrl + scroll wheel.

I figured out the dashes and rectangles and lines but the scaling and overlay with Draw is not possible and the only option after that is to render everything into a ImageBuffer and all the methods I have seen are really slow.

Oblivion wrote on Wed, 05 July 2023 14:18For such operations, you can directly use ImageBuffer & Image utility functions. They are cheaper.

Best regards,
Oblivion

Subject: Re: Font and Image rendering slow
Posted by Oblivion on Thu, 06 Jul 2023 23:54:58 GMT
View Forum Message <> Reply to Message

But you can use the painter efficiently. What you need to consider is not using it in a hot path such as Paint, if possible (if you need high FPS, I mean). A basic but very effective strategy: draw the image elsewhere, and only paint it to system draw in Paint. Also, an image cache will help here

significantly.

I have attached a simple code that uses an image cache to draw 100x100 icons.
The example code draws the icon to a ImagePainter object and can manipulate (rotate) it on-the-fly, without a significant performance hit. Only 8 slots of the cache will be filled in this code (rotation is in 4 directions * left and right, so 8 images total). Images have alpha channel enabled.

The result of consecutive 305 rotations (window maximized, GTK, 1920x1080), calls to Paint (RELEASE MODE), no explicit clip:

TIMING Paint : 834.63 ms - 2.74 ms (835.00 ms / 305), min: 2.00 ms, max: 4.00 ms, nesting: 0 - 305

Edit: Code updated.

Best regards,
Oblivion

File Attachments

1) ImageTest.zip, downloaded 178 times

Subject: Re: Font and Image rendering slow
Posted by Oblivion on Fri, 07 Jul 2023 10:55:51 GMT
View Forum Message <> Reply to Message

By the way, I have tested the modified demo (with Draw) on Windows 10, and got ~150 FPS in release mode, maximized window. (same machine).

Even the vanilla demo you've provided runs around ~120 fps there, but lags when starting up and resizing (for obviouse reasons: Huge canvas, no clipping).

Best regards,
Oblivion

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Fri, 07 Jul 2023 20:21:45 GMT
View Forum Message <> Reply to Message

I don't only run on windows. I primarily run on linux and its building against the Gtk libs and not X11. It seems to behave differently than windows.

Also my work machine is old. I am sure my home computer that has a AMD 5950X, with 128Gigs

of ram and a 3080TI 12Gig would run just find with out any problems.

But since the software I am making will go into an industrial setting I can't expect them to be running a high end machine to run something as simple as this.

The test I am running is. I add a Refresh directly to the end of the Paint code, so that once its done painting it will draw it again. Then I measure the number of frames it has drawn in a second. The reason for this is that when I Drag and Icon across the screen this is essentially what is happening which is when I see the performance hit.

The example you just gave me gets 51 FPS on my work machine under linux with a 1920x1080 full screen, and the fact that it does not even do any form of clipping and still manages 51 FPS on my machine is better then what I have seen so far. I will get this implemented on my setup.

I believe I can even do scaling in the cache section of the code since you were able to do a rotate.

Now the final problem I am not seeing how to fix though since everything is direct to the Upp::Draw in the main paint section is how to do a full screen transparency effect.

Here is a complete demo:
Zoom In and Out: Ctrl+Scroll
Select Icon: Left Mouse
Toggle Select Multiple: Ctrl+Left Mouse
Group Select: Shift+Left Mouse Drag on blank space and no selected icon
Create Link: Click and drag between icon from there small box handles
Delete Link: Click a link and then press the delete key
Show Link Details: F1 when links are present
Mouse over Summary: Move the cursor over icon and half a second later a summary should pop up
Show Overlay: F2
Show FPS: F3
Scroll Left/Right: Shift+Scroll Wheel
Scroll Up/Down: ScollWheel

This cover all of the Drawing elements that my program currently used.

File Attachments

1) upp_demo.zip, downloaded 182 times

Subject: Re: Font and Image rendering slow
Posted by Oblivion on Sat, 08 Jul 2023 09:44:14 GMT
View Forum Message <> Reply to Message

Quote:Now the final problem I am not seeing how to fix though since everything is direct to the Upp::Draw in the main paint section is how to do a full screen transparency effect.

I might be getting you wrong here, but do you mean this? (If you mean window background transparency effect, there is no direct method for that in Upp, not that I know of, at least)

```
void frm_main::Paint(Draw& w) {
  Size sz = GetSize();
  w.DrawRect(sz, Yellow()); // System draw, background = yellow.
  ImageBuffer ib(sz);
  BufferPainter bp(ib, MODE_ANTIALIASED);
  bp.Clear(RGBAZero());     // Buffer now has transparent background.

  _icons_paint(bp, scroll.Get().x, scroll.Get().y, allow_scaling, render_everything);

  if(show_overlay) {
    _paint_overlay(bp);
  }

  SetSurface(w, 0, 0, sz.cx, sz.cy, ~ib); // You need to use SetSurface to directly paint buffer content here (this is optimized for systemdraw (possibly faster than usual image drawing) ...
}
```

Also, IMO, the optimized way of painting a large painter object is, drawing it in another thread, in parallel -if possible- and only update the system draw -using SetSurface- as needed. (If there is no strict requirement on doing all the work in single-threaded environment, I mean.)

Further note: You can use also use ImagePainter class instead of using a ImageBuffer + BufferPainter. ImagePainter is basically a wrapper for BufferPainter + ImageBuffer.

Best regards,
Oblivion

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Sun, 09 Jul 2023 18:14:58 GMT
View Forum Message <> Reply to Message

I will look into it, but the transparency was in the overlay which is enabled with F2 in the full demo. You will see a grey transparency where you can see the icons under it and then a blue box with progress bars and a cancel button.

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Fri, 14 Jul 2023 15:18:09 GMT
View Forum Message <> Reply to Message

So here is a new full demo.

I added a new hot key which is F4 which will draw lines from the first icon to the rest. with that on and with F1 on the FPS for me drops to 3 FPS

I have implemented some of the things suggest but I decided to not do the image cache since once you have a bunch of images cached it actually hurts performance trying to get the correct image.

I did implement a simple cache which is if there is no change the keep current rendered image if there is a change then draw a new one.

I also faked the transparency effect which helped the speed a a lot.

I also switched away from iterators in the draw code.

I also changed how the scaling works.

The performance is much better then it used to be. If you happen to have anymore suggestions on how to get more speed that would be awesome.

File Attachments

1) upp_demo.zip, downloaded 169 times

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Fri, 14 Jul 2023 15:47:09 GMT
View Forum Message <> Reply to Message

Well the new demo does not work correctly in Windows. It looks fine in linux but not windows. It look like the ImageBuffer which is used for drawing the lines and is 100% transparent every where except the lines and the summary icon is drawing as black. also the randint which is in experimental random does not exist in the windows version it seems.

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Fri, 14 Jul 2023 16:14:34 GMT
View Forum Message <> Reply to Message

So I switched away from SetSurface back to w.DrawImage due to it working differently in windows and linux, which then allowed for the transparency to work again.

Now for some reason on window the font for the icons is not working but it works on linux. I did a test and manually put text at the pos 10, 10 and that text showed up so it seems windows is not handling that the same either.

Subject: Re: Font and Image rendering slow
Posted by devilsclaw on Fri, 14 Jul 2023 16:49:00 GMT
View Forum Message <> Reply to Message

I switched away from SetSurface since it does not act the same in windows as it does in linux.

I also switch away from the experimental randint

I also moved the bp.scale back before any rendering is done to the BufferPainter. It does matter where its at.

The new demo works the same in windows as it does in linux.

File Attachments

1) upp_demo.zip, downloaded 175 times