Subject: DrawWin32.cpp: PrintDraw::InitPrinter()
Posted by Tom1 on Tue, 18 Nov 2008 15:09:51 GMT
View Forum Message <> Reply to Message

Hi,

May I (?) suggest changing the code for DrawWin32.cpp: PrintDraw::InitPrinter() as follows:

Current:

```
::SetMapMode(handle, MM_ANISOTROPIC);
::SetViewportExtEx(handle, inchPixels.cx, inchPixels.cy, NULL);
::SetViewportOrgEx(handle, 0, 0, NULL);
::SetWindowExtEx(handle, 600, 600 , NULL);
::SetWindowOrgEx(handle, 0, 0, NULL);
```

Modified:

```
::SetMapMode(handle, MM_ANISOTROPIC);
::SetWindowExtEx(handle,pagePixels.cx,inchPixels.cx*pagePixels.cy/inchPixels.cy,NULL);
::SetWindowOrgEx(handle, 0, 0, NULL);
::SetViewportExtEx(handle, pagePixels.cx, pagePixels.cy, NULL);
::SetViewportOrgEx(handle, 0, 0, NULL);
pagePixels.cy=inchPixels.cx*pagePixels.cy/inchPixels.cy;
```

I ended up changing this because I faced multiple problems while printing on an Epson Stylus Photo 1290. The raster and vector coordinate spaces were different since the printer's native 360x360 or 720x720 dpi resolution did not nicely map to the default 600x600 dpi resolution. Also, when printing large images, the image was trashed for some reason. (I made a custom ::SetDIBitsToDevice() based "DrawImage" replacement to circumvent that, but that's another story.)

The modification attempts to present a device context with native pixels, if possible, and also tries to adjust non-square device pixels to square device context pixels.

Also, the Draw::GetPagePixels now returns the corrected value. (This was not the case before and it was a part of the problem.)

NOTE: If somebody has a printer (device context) with non-square pixels, it would be good to test if this change breaks anything on printing of vectors and/or rasters. (The Epson's printer pixel is not square, but the device context pixel from the printer driver is luckily square.)

If someone really relies on the fixed 600x600 dpi resolution of a upp generic printer device, they may possibly not like my suggestion.

// Tom

Subject: Re: DrawWin32.cpp: PrintDraw::InitPrinter()
Posted by mirek on Sun, 23 Nov 2008 10:01:31 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 18 November 2008 10:09
If someone really relies on the fixed 600x600 dpi resolution of a upp generic printer device, they may possibly not like my suggestion.

Well, but that is how physical dimensions in U++ are defined.

When you are doing output with physical representation (can be e.g. PDF too), it is guaranteed that 1 pixel is equivalent to 1 pixel on 600dpi printer. We call this unit the "dot". Rellying on fixed 600 dpi resolutions seems to be very convenient.

What exactly was the problem? Also, what OS is it? Win32?

I believe anything not really related to Images should print OK, with correct aspect ratio.

As for images, well, pehaps we should not relly on GDI to rescale the image and do it ourselves - that should solve the problem.

Mirek

Subject: Re: DrawWin32.cpp: PrintDraw::InitPrinter()
Posted by Tom1 on Mon, 24 Nov 2008 11:41:22 GMT
View Forum Message <> Reply to Message

I'm testing this on Windows XP. Luckily the Epson Stylus Photo 1290 driver has a preview option in XP and it does not even require the printer to exist to be able to install the driver and use its preview option.

There are two separate problems:

1) Only about 60% of width and about 60% of height of the printable area is covered by the printout.

2) The DrawImage(rect,image) produces a broken image probably due to the resolution mismatch (360x360 versus 600x600). It does not get correctly re-scaled from 600 dpi to 360 dpi in the GDI or the printer driver.

Please see the attached files broken.png and fixed.png.

Current code:

::SetMapMode(handle, MM_ANISOTROPIC);

```
::SetViewportExtEx(handle, inchPixels.cx, inchPixels.cy, NULL);
::SetViewportOrgEx(handle, 0, 0, NULL);
::SetWindowExtEx(handle, 600, 600 , NULL);
::SetWindowOrgEx(handle, 0, 0, NULL);
```

Result with current code:

My slightly enhanced solution looks like this:

```
::SetMapMode(handle, MM_ANISOTROPIC);
int dpi=min(inchPixels.cx,inchPixels.cy);
::SetWindowExtEx(handle, dpi*pagePixels.cx/inchPixels.cx, dpi*pagePixels.cy/inchPixels.cy,
NULL);
::SetWindowOrgEx(handle, 0, 0, NULL);
::SetViewportExtEx(handle, pagePixels.cx, pagePixels.cy, NULL);
::SetViewportOrgEx(handle, 0, 0, NULL);
pagePixels.cx=dpi*pagePixels.cx/inchPixels.cx;
pagePixels.cy=dpi*pagePixels.cy/inchPixels.cy;
inchPixels.cx=dpi;
inchPixels.cy=dpi;
```

The result after the above changes:

With the above change, it works with the standard DrawImage.

Just an opinion, but I think the best printing quality can be obtained using the printer's hardware resolution (and optionally dropping it with integer dividers if it is excessively high). For correct page size and scaling I always use Draw::GetPagePixels() and Draw::GetPixelsPerInch() to get the map scale right on both screen and printer.

(This is irrelevant for the current issue at hand, but since it popped up: The SetDIBitsToDevice() uses hardware resolution and does not respect these Viewport/Window mappings except maybe for the starting point. StretchDIBits(), however, does work with the mappings and is also pretty economic way to put low resolution images to high resolution printers, as it does not necessarily pre-rescale the image to the printer resolution before sending it to the printer.)

// Tom

File Attachments

1) broken.png, downloaded 1213 times
2) fixed.png, downloaded 1194 times

Subject: Re: DrawWin32.cpp: PrintDraw::InitPrinter()
Posted by mirek on Tue, 25 Nov 2008 22:47:32 GMT

Interesting and confusing. Frankly, looks like driver bug.

That said, U++ does some weird things to reduce the required bandwidth and memory footprint. Namely, it splits large images into horizontal bands and scans those bands for uniform color areas. Those areas, if any, are then painted using DrawRect.

I believe that what we see on screenshot is that some of that optimization somehow collapsed or got recomputed wrongly (band part started too much in the left). I think that would point to driver bug.

Well, I plan to buy some cheap Epson ink-jet to check this out. (Unfortunately, it has to be Epson - no other is doing 'odd' DPI printers anymore

Quote:
Just an opinion, but I think the best printing quality can be obtained using the printer's hardware resolution (and optionally dropping it with integer dividers if it is excessively high). For correct page size and scaling I always use Draw::GetPagePixels() and Draw::GetPixelsPerInch() to get the map scale right on both screen and printer.

IMO, as long as we abstract from the particular image rescaling problem, the difference should not be next to nonexistent, for drawing text and polygons. After all, some coordinate rescaling is involved in either case...

I think we should aim for fixing the DrawImageOp routine - it should not be hard to switch to "physical mode" for the time the Image is printed, then switch back to normal "dots" mode, rescaling coordinates and target image size in the process...

Mirek

Subject: Re: DrawWin32.cpp: PrintDraw::InitPrinter()
Posted by Tom1 on Wed, 26 Nov 2008 10:31:39 GMT

Mirek,

You do not need to waste your money on an Epson ink-jet. Just install the latest driver on XP and check the preview option to verify the results. (I don't have such printer either, but my client unfortunately does.)

--

Here's a minimal fix for ImageWin32/Image::Data::Paint() that gets things working with the 600 dpi default resolution:

if(GetKind() == IMAGE_OPAQUE && paintcount == 0 && sr == Rect(sz) && !w.IsMetaFile() && IsWinNT()) {//TODO !IsWinNT

Becomes:

if(GetKind() == IMAGE_OPAQUE && paintcount == 0 && sr == Rect(sz) && !w.IsMetaFile() && IsWinNT() && !w.IsPrinter()) {//TODO !IsWinNT

For printer jobs, this effectively passes the image rendering work from ::SetDIBitsToDevice based SetSurface implementation to another existing image rendering mechanism that works OK.

Additionally, to keep Draw::GetPagePixels() and Draw::GetPixelsPerInch working on the remapped printing context, the following needs to be added into the end of PrintDraw::InitPrinter():

```
pagePixels.cx=600*pagePixels.cx/inchPixels.cx;
pagePixels.cy=600*pagePixels.cy/inchPixels.cy;
inchPixels.cx=600;
inchPixels.cy=600;
```

--

However, I would still strongly suggest on supporting the printer's native resolution for best printing quality and future flexibility -- even if it's an optional flag for PrintDraw::PrintDraw. The code for PrintDraw::InitPrinter then becomes:

```
int dpi=flagNativeDPI?min(inchPixels.cx,inchPixels.cy):600;
::SetMapMode(handle, MM_ANISOTROPIC);
::SetWindowExtEx(handle, dpi*pagePixels.cx/inchPixels.cx, dpi*pagePixels.cy/inchPixels.cy, NULL);
::SetWindowOrgEx(handle, 0, 0, NULL);
::SetViewportExtEx(handle, pagePixels.cx, pagePixels.cy, NULL);
::SetViewportOrgEx(handle, 0, 0, NULL);
pagePixels.cx=dpi*pagePixels.cx/inchPixels.cx;
pagePixels.cy=dpi*pagePixels.cy/inchPixels.cy;
inchPixels.cx=dpi;
inchPixels.cy=dpi;
```

// Tom

Subject: Re: DrawWin32.cpp: PrintDraw::InitPrinter()
Posted by rylek on Fri, 28 Nov 2008 12:31:29 GMT
View Forum Message <> Reply to Message

Hello Tom!

I think I agree with you. The 600 DPI "dot" is a handy unit for most ordinary printing stuff but there are cases where a truly low-cost development environment like U++ should allow switching to native device coordinates; at the very least today there are devices with much larger resolution than 600 DPI and it's a shame we can't legally draw on such devices with the highest possible resolution. On the other hand, for those unlucky ones of us who still have to support W98 from time to time, sometimes there's need to keep the DPI low because of the old GDI 16-bit coordinate limit. We could perhaps use the same approach as in DrawingDraw, letting the PrintDraw switch between dot-based and native device pixel units. Perhaps there would be not much harm in even using the existing Pixels() and Dots() methods to distinguish those two modes.

Regards

Tomas