Subject: Native DPI
Posted by Tom1 on Thu, 21 May 2015 09:45:26 GMT
View Forum Message <> Reply to Message

Hi,

Is there any specific reason for having a fixed Size(96,96) returned for screen DPI in the following Windows SystemDraw function?:

```
Size SystemDraw::GetNativeDpi() const
{
 return Dots() ? nativeDpi : Size(96, 96);
}
```

I just got a new 32" 4K display on my desktop with native 144 DPI and metric size of objects is now wrong because of this default DPI.

May I suggest GetNativeDpi() reporting whatever Windows reports directly, like this?

```
Size SystemDraw::GetNativeDpi() const
{
 return nativeDpi;
}
```

Best regards,

Tom

Subject: Re: Native DPI
Posted by mirek on Sat, 23 May 2015 18:23:55 GMT
View Forum Message <> Reply to Message

Well, long term recommendation is not to depend on DPI for screen, its value is mostly bogus (I am speaking here about values that OS returns).

Instead, rescale everything based on current system font size - that values is telling, because that is how user feels about screen resolution. Ctrl::*Zoom functions do exactly that.

That said, I have never tested with UHD and I am happy you will be testing it. I mostly expect nasty problems with icon sizes... maybe we will need to do something like automatic 2x upscale there...

(BTW, I nearly bought UHD Lenovo notebook this week, but then opted out because I need antireflexive and that one was glossy... :(

Mirek

Subject: Re: Native DPI
Posted by Tom1 on Sun, 24 May 2015 08:29:58 GMT
View Forum Message <> Reply to Message

Hi,

In fact I'm already using GetStdFontSize for scaling GUI in general, but my map display is supposed to work with real scales, like 1:10000 etc. This can not be compensated for using GetStdFontSize. So, could you consider the GetNativeDpi change anyway? If my client purchases displays with drivers reporting bad DPI values, then it's no longer my problem. Of course one way to go is to add user calibration of display DPI, but this may get difficult in multi display environments with different screens.

And yes: GUIs are damaged, especially toolbars with icons... For example, TheIDE toolbar on the same line with menu gets partly clipped and icons are really tiny. I think it would be useful if the system scales toolbars and selects properly sized icon variants based on the StdFontSize.

I started to render my (simple) toolbar icon bitmaps run-time from vector data using Painter, so that the icons scale perfectly to the toolbar button rect derived from StdFontSize. A bit of work but I only had a few icons in this case.

If you do something about those icons in TheIDE, I can report the results here. Just let me know if there are any changes to this.

Best regards,

Tom

Subject: Re: Native DPI
Posted by mirek on Sun, 24 May 2015 19:16:26 GMT
View Forum Message <> Reply to Message

As for DPI: Perhaps the right solution here is to read DPI from Windows directly?

(Seriously, one reason why I stubbornly return 96 is that I do not want users to start relying on it ;)

As for HIDPI troubles: Could you post a screenshot please?

Mirek

Subject: Re: Native DPI
Posted by Tom1 on Mon, 25 May 2015 08:05:39 GMT
View Forum Message <> Reply to Message

Hi,

For this display, the ratio of the size of toolbar and menubar change according to "Windows Control Panel > All Control Panel Items > Display > Change the size of all items" where recommended setting for this screen appears to be closer to the "Larger" end of the dial than the other. This adjustment scales all Windows GUI objects. The results can be seen in StdFontSize in Upp.

Clearly, it would be useful to automatically scale user selectable font sizes in relation to StdFontSize too.

This screenshot shows TheIDE in its default setting:

This screenshot shows TheIDE when toolbar is in a row with the menubar shows clipping of the toolbar and lack of scaling of droplist widths:

So, icons in various places suffer from lack of scaling.

Best regards,

Tom

File Attachments
1) TheIDE2-HIDPI.png, downloaded 1310 times
2) TheIDE-HIDPI.png, downloaded 1240 times

Subject: Re: Native DPI
Posted by Tom1 on Mon, 25 May 2015 14:54:54 GMT
View Forum Message <> Reply to Message

Hi,

It seems Hi-DPI aware applications now have new requirements for per monitor DPI awareness. This is new in Windows 8.1. I think we need support for this inside upp:

https://msdn.microsoft.com/en-us/library/windows/desktop/dd4 64659%28v=vs.85%29.aspx

Best regards,

Tom

Subject: Re: Native DPI
Posted by mirek on Mon, 25 May 2015 21:13:30 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 25 May 2015 16:54Hi,

It seems Hi-DPI aware applications now have new requirements for per monitor DPI awareness. This is new in Windows 8.1. I think we need support for this inside upp:

https://msdn.microsoft.com/en-us/library/windows/desktop/dd4 64659%28v=vs.85%29.aspx

Best regards,

Tom

Well, great, GetDPIForMonitor is Win8.1 specific only...

Anyway, I agree we need to handle all this and soon. Based on screenshots, I would say that in reality, we are in fact quite close, theide needs fixing but these are not problems of library. Except that we need have a new Chameleon variable that would rescale images 2x here and there...

As for rescaling algorithm, that one is tricky. Maybe generic Lancosz3 or bilinear would be OK, but we might consider somehing like these:

http://en.wikipedia.org/wiki/Image_scaling

Mirek

Subject: Re: Native DPI
Posted by chickenk on Tue, 26 May 2015 05:35:35 GMT
View Forum Message <> Reply to Message

Have a look at Waifu2x as well: http://waifu2x.udp.jp/
A blog post by John Resig: http://ejohn.org/blog/using-waifu2x-to-upscale-japanese-prin ts/

Cheers
Lionel

Subject: Re: Native DPI
Posted by Tom1 on Tue, 26 May 2015 08:22:31 GMT
View Forum Message <> Reply to Message

Hi,

Not having GetDPIForMonitor() does not seem to be a catastropy. I tried a few things on Windows 8.1 with two displays (UHD as primary and FullHD as secondary):

It seems we can always query "Windows main display's" DPI using GetDeviceCaps(handle,LOGPIXELSX) and GetDeviceCaps(handle,LOGPIXELSY).

If the window is moved to another display, a WM_DPICHANGED message is received with wParam LOWORD and HIWORD reporting the new DPI values of the monitor, the window was moved to.

If the initial display DPI is read at window creation time with GetDeviceCaps() and always updated when a WM_DPICHANGED message is received, I think we have it all covered.

Best regards,

Tom

Subject: Re: Native DPI
Posted by mirek on Wed, 27 May 2015 08:03:12 GMT
View Forum Message <> Reply to Message

I have tried some initial fixes in theide and CtrlLib for HiDPI - basically, HiDPI should be now detected (based on font size), and as first test, toolbar icons are upscaled (in ideal worlds, we would be using svg or be downscaling big icons for non-HiDPI, but to maintain compatibility, upscaling is what we should try first imo).

Also, layout bugs in theide were fixed.

Please test and report (but be aware that theide is now in transition to new Assist++ and new features are not quite finished yet).

Mirek

(Note that this update does not deal with monitor DPI issues yet.)

Subject: Re: Native DPI
Posted by mirek on Wed, 27 May 2015 08:14:13 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 26 May 2015 10:22Hi,

Not having GetDPIForMonitor() does not seem to be a catastropy. I tried a few things on Windows 8.1 with two displays (UHD as primary and FullHD as secondary):

It seems we can always query "Windows main display's" DPI using GetDeviceCaps(handle,LOGPIXELSX) and GetDeviceCaps(handle,LOGPIXELSY).

Is default font size being changed as well?

(I sort of doubt it, this is our current code for reading font height

```
NONCLIENTMETRICS ncm;
ncm.cbSize = sizeof(ncm);
::SystemParametersInfo(SPI_GETNONCLIENTMETRICS, sizeof(ncm), &ncm, 0);
name = FromSystemCharset(ncm.lfMenuFont.lfFaceName);
height = abs((int)ncm.lfMenuFont.lfHeight);
```

- there does not seem anything that would link it with any monitor)

Quote:
If the window is moved to another display, a WM_DPICHANGED message is received with wParam LOWORD and HIWORD reporting the new DPI values of the monitor, the window was moved to.

If the initial display DPI is read at window creation time with GetDeviceCaps() and always updated when a WM_DPICHANGED message is received, I think we have it all covered.

Well, as long as we maintain single DPI for the whole application...

Mirek

Subject: Re: Native DPI
Posted by Tom1 on Wed, 27 May 2015 11:32:14 GMT
View Forum Message <> Reply to Message

Hi,

Thanks! Your icon scalings and other changes work well across the displays. (Of course, icons are not as sharp on UHD as one might wish, but this is a minor issue.) Drop lists and some other items still have rather small arrow heads compared to the button size. Also, the check marks in front of menu items are perhaps slightly small. On secondary display there is an overall fuzzyness issue, but more on that can be found below.

--

Multi display:

The current code for reading font height returns the same exact value regardless the monitor the window is dragged to. (Declaring the process PM aware or not, does not have any effect to the result.)

Unless the process is declared per-monitor hi-dpi aware, the entire window contents look fuzzy on the smaller FullHD (secondary) display because of Windows scaling for non-per-monitor-hi-dpi-compliant applications. Declaring per-monitor hi-dpi support for the application removes the fuzzyness, but everything grows in size because the fonts and everything else around retains the same size in pixels now. So, the application should be able to handle the DPI change now.

IMO: Proper per-monitor hidpi support in upp would:

- Work on per-TopWindow basis
- First detect the primary display DPI using GetDeviceCaps(handle,LOGPIXELSX) and read the system default font size using the current code.
- Then the actual StdFontSize would be calculated based on the above and the TopWindow hosting display reported DPI from received WM_DPICHANGED messages

--

I'm now aware of the stupid way Windows reports DPI values. It is simply a user setting of type: 'How large items would you like to have on your screen?'. So there is nothing there to work on towards scaling e.g. a map content on screen to a real scale. In order to implement true map scaling in my software, I would need two things: The pixel resolution of the display the window is currently hosted on and the diagonal size of the monitor involved. I'm afraid the diagonal size needs to be manually input by the user, but the pixel resolution can be read from Windows using code similar to this when WM_DPICHANGED message arrives:

```
MONITORINFO mi;
Zero(mi);
mi.cbSize=sizeof(MONITORINFO);
if(GetMonitorInfo(MonitorFromWindow(hWnd,MONITOR_DEFAULTTONEAREST),&mi)){
 Rect monitor_resolution(mi.rcMonitor);
```

Hopefully this code can be embedded in TopWindow's message loop and the resulting pixel resolution of current monitor exposed through some TopWindow function.

Best regards,

Tom

Subject: Re: Native DPI
Posted by mirek on Wed, 27 May 2015 12:13:32 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 27 May 2015 13:32Hi,

Thanks! Your icon scalings and other changes work well across the displays. (Of course, icons are not as sharp on UHD as one might wish, but this is a minor issue.) Drop lists and some other items still have rather small arrow heads compared to the button size. Also, the check marks in front of menu items are perhaps slightly small. On secondary display there is an overall fuzzyness issue, but more on that can be found below.

Well, hopefully I will find a better scaling algorithm...

Quote:
- Then the actual StdFontSize would be calculated based on the above and the TopWindow hosting display reported DPI from received WM_DPICHANGED messages

I still miss how this is possible. The function we are using does not have any binding to monitor (there is no HDC or HWND or HMONITOR passed as parameter).

Obviously, as long as UHD is your primary display and application sets "Vista style" DPI awarness, you are getting "BIG" font size...

Mirek

Subject: Re: Native DPI
Posted by Tom1 on Thu, 28 May 2015 07:47:38 GMT
View Forum Message <> Reply to Message

Hi,

This is what I tried.

Added file manifest.xml to the ide package to enable per-monitor DPI awareness:

```
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0"
xmlns:asmv3="urn:schemas-microsoft-com:asm.v3" >
 <asmv3:application>
  <asmv3:windowsSettings xmlns="http://schemas.microsoft.com/SMI/2005/WindowsSettings">
   <dpiAware>True/PM</dpiAware>
  </asmv3:windowsSettings>
 </asmv3:application>
</assembly>
```

This also needs a Custom build step to be added for WIN32 - post-link to be effective:

mt.exe -manifest manifest.xml -outputresource:$(EXEPATH);1

Then added two Size variables to TopWindow class in TopWindow.h:

```
 Size  currentdpi;
 Size  primarydpi;
```

Finally added two new cases to TopWindow::WindowProc in TopWin32.cpp:

```
 case WM_CREATE:{
  HDC hdc=GetDC(hwnd);
  primarydpi=Size(GetDeviceCaps(hdc,LOGPIXELSX),GetDeviceCaps(hdc,LOGPIXELSY)); //
Returns primary display DPI
  ReleaseDC(hwnd,hdc);
  // RLOG(Format("WM_CREATE: primary DPI(%d,%d)",primarydpi.cx,primarydpi.cy));
  break;
 }
 case 0x02E0: // WM_DPICHANGED -- e.g. moved window to another monitor with different DPI
  // WndSetPos(Rect(*(RECT*)lParam)); // LPARAM: RECT* of suggested new window pos --
non-optimal

  currentdpi=Size(LOWORD(wParam),HIWORD(wParam)); // Get DPI setting for current display
monitor
  // RLOG(Format("WM_DPICHANGED: current DPI(%d,%d)",currentdpi.cx,currentdpi.cy));
  if(primarydpi.cy&&currentdpi.cy){
   String face;
   int height;
   GetStdFontSys(face,height);
   SetStdFont(GetStdFont().Height(height*currentdpi.cy/primarydpi.cy));
   GUI_HiDPI_Write(currentdpi.cx>150?1:0); // This does not work optimally here.
  }
  break;
```

Possible DPI values returned by Windows depending on scaling set in Control Panel / Display:

96 DPI = 100% scaling
120 DPI = 125% scaling

144 DPI = 150% scaling
192 DPI = 200% scaling

... but there may be even more available. I think I saw 250% somewhere already.(?)

The result is that Font size scales properly when moving the window between the two monitors with different DPI -- 192 and 96 in my case.

There are problems though:

- Line spacing in lists does not adjust according to the StdFontSize while the actual fonts in the lists do.
- Check boxes, radio buttons, scroll bars and spin control arrows in dialogs remain in the HI-DPI mode regardless the change in the mode.
- Manually set font sizes in TheIDE do not scale according to the changes in StdFontSize. In fact, it would be very useful to just declare a scaling percentage e.g. 25... 400 in comparison to the StdFontSize for these fonts instead of their actual sizes. This would keep them proportionally sized even in single monitor scenarios where display scaling is changed in Windows Control Panel.

Further on I think all controls should be scaled and drawn based directly on a per TopWindow defined StdFontSize. This becomes evident when opening a new dialog on a different monitor or moving it onto another monitor. Then scaling really gets out of control.

Best regards,

Tom

Subject: Re: Native DPI
Posted by mirek on Fri, 29 May 2015 08:16:25 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Thu, 28 May 2015 09:47Hi,

This is what I tried.

Added file manifest.xml to the ide package to enable per-monitor DPI awareness:

```
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0"
xmlns:asmv3="urn:schemas-microsoft-com:asm.v3" >
 <asmv3:application>
  <asmv3:windowsSettings xmlns="http://schemas.microsoft.com/SMI/2005/WindowsSettings">
   <dpiAware>True/PM</dpiAware>
  </asmv3:windowsSettings>
```

```
 </asmv3:application>
</assembly>
```

This also needs a Custom build step to be added for WIN32 - post-link to be effective:

```
mt.exe -manifest manifest.xml -outputresource:$(EXEPATH);1
```

Then added two Size variables to TopWindow class in TopWindow.h:

```
 Size  currentdpi;
 Size  primarydpi;
```

Finally added two new cases to TopWindow::WindowProc in TopWin32.cpp:

```
 case WM_CREATE:{
  HDC hdc=GetDC(hwnd);
  primarydpi=Size(GetDeviceCaps(hdc,LOGPIXELSX),GetDeviceCaps(hdc,LOGPIXELSY)); //
Returns primary display DPI
  ReleaseDC(hwnd,hdc);
  // RLOG(Format("WM_CREATE: primary DPI(%d,%d)",primarydpi.cx,primarydpi.cy));
  break;
 }
 case 0x02E0: // WM_DPICHANGED -- e.g. moved window to another monitor with different DPI
  // WndSetPos(Rect(*(RECT*)lParam)); // LPARAM: RECT* of suggested new window pos --
non-optimal

  currentdpi=Size(LOWORD(wParam),HIWORD(wParam)); // Get DPI setting for current display
monitor
  // RLOG(Format("WM_DPICHANGED: current DPI(%d,%d)",currentdpi.cx,currentdpi.cy));
  if(primarydpi.cy&&currentdpi.cy){
   String face;
   int height;
   GetStdFontSys(face,height);
   SetStdFont(GetStdFont().Height(height*currentdpi.cy/primarydpi.cy));
   GUI_HiDPI_Write(currentdpi.cx>150?1:0); // This does not work optimally here.
  }
  break;
```

Possible DPI values returned by Windows depending on scaling set in Control Panel / Display:

96 DPI = 100% scaling
120 DPI = 125% scaling
144 DPI = 150% scaling
192 DPI = 200% scaling

... but there may be even more available. I think I saw 250% somewhere already.(?)

The result is that Font size scales properly when moving the window between the two monitors with different DPI -- 192 and 96 in my case.

There are problems though:

- Line spacing in lists does not adjust according to the StdFontSize while the actual fonts in the lists do.
- Check boxes, radio buttons, scroll bars and spin control arrows in dialogs remain in the HI-DPI mode regardless the change in the mode.
- Manually set font sizes in TheIDE do not scale according to the changes in StdFontSize. In fact, it would be very useful to just declare a scaling percentage e.g. 25... 400 in comparison to the StdFontSize for these fonts instead of their actual sizes. This would keep them proportionally sized even in single monitor scenarios where display scaling is changed in Windows Control Panel.

Further on I think all controls should be scaled and drawn based directly on a per TopWindow defined StdFontSize. This becomes evident when opening a new dialog on a different monitor or moving it onto another monitor. Then scaling really gets out of control.

Best regards,

Tom

I am afraid that per-window scaling is a massive change. We will do it eventually, but for the next release, I would concentrate on 'Vista style' DPI awareness (means primary UHD display scaled by us, secondary non-UHD display scaled down by windows).

(Note that HiDPI is not the primary focus of next release; new Assist++ parser, better support for C++11 are main topics).

Mirek

Subject: Re: Native DPI
Posted by mirek on Fri, 29 May 2015 11:27:04 GMT
View Forum Message <> Reply to Message

I am playing a bit with icon upscaling.

Please see the enclosed package. It displays a set of icons upscaled with various (U++ build-in) algorithms, then does unsharp-mask postprocessing - you can set amount and number of passes.

My favorite method is 7 (that is lanczos 3 rescale filter) with parameters 20/3 or 50/2.

I would like to ask how it looks on UHD display....

Mirek

File Attachments
1) AAA.7z, downloaded 356 times

Subject: Re: Native DPI
Posted by Tom1 on Fri, 29 May 2015 15:23:29 GMT
View Forum Message <> Reply to Message

Hi,

In my opinion method 4 without any sharpening gives the best overall result. The details appear relatively sharp while jagging on 45 degree angles is negligible.

(The circle is not good in any case. Either everything becomes too fuzzy or the circle remains jagged on the edge -- or both at the same time. I think the circle is not going to be nice no matter what.)

I can see your point with the sharper 7 + 50/2 or 20/3. However, they seem to leave some artifact behind -- like faint shadows around the objects -- but from a distance, those are barely visible.

So, just follow your own preference on this one.

Best regards,

Tom

Subject: Re: Native DPI
Posted by mirek on Sat, 06 Jun 2015 05:37:52 GMT
View Forum Message <> Reply to Message

I am now working on creating reasonable framework for HiDPI.

Initially, I was thinking that I will create a couple of functions like "Rescale image 2x if in HiDPI mode" etc... and put these functions everywhere needed.

After a couple of hours of "putting where needed", I am not so sure anymore... :)

I am now rather thinking about Draw HiDPI which would do all resizes automagically, recycling "BeginNative/EndNative" for rendering in HiDPI coordinates...

What do you think?

Subject: Re: Native DPI
Posted by Tom1 on Mon, 08 Jun 2015 08:31:51 GMT
View Forum Message <> Reply to Message

Hi,

I'm afraid I can't imagine all the implications of this approach.

Are you saying that coordinates in Draw would no longer be mapped 1:1 to pixels on screen, unless BeginNative was called? If so, I'm afraid many programs with Draw based graphics would break, or at least the quality of the graphic content would suffer. (Then graphics based on Draw in many applications should be changed to use BeginNative to restore rendering quality.)

Did I get this right?

Generally, I think that the coordinate space should be native in pixels and only drawing of controls should be scaled to match the desired physical size on screen -- which for standard controls should probably be determined from the Std font size.

Best regards,

Tom

Subject: Re: Native DPI
Posted by mirek on Sat, 13 Jun 2015 15:04:07 GMT
View Forum Message <> Reply to Message

Well, I am afraid we have run here into the same problem that made M$ create that HiDPI-aware flag in the first place..

In theory, all U++ apps should use LayoutZoom logic everywhere, then (with a bit of icon rescaling or replacing) everything would work just fine.

In practice, even in CtrlLib there is a lot of "fixed number of pixels" issues. Fixing them all will take time - and that would not solve all U++ user applications out there having the same issues.

Well, I am now not thinking about "BeginNative" anymore, but rather about per Ctrl HiDPI-aware flag. If not set (which unfortunately has to be default), Draw would scale up all things

automatically.

Unlike Win32, which scales resulting bitmap graphics, Draw would scale before rendering, which would result in crisp texts - I would say, this is still quite and advantage.

So the only disadvantage would be less precise "placement" (IMO hardly detectable) and images. The issue of images is in fact the same for HiDPI aware mode...

Mirek

Subject: Re: Native DPI
Posted by Tom1 on Sun, 14 Jun 2015 22:00:12 GMT
View Forum Message <> Reply to Message

Hi,

From a wider point of view, this sounds like the right way to go. Applications with standard controls scale up beautifully in general without any additional effort, and that is important.

However, from my narrow point of view, I additionally need to be sure that I can still have the low level access to "pixel for pixel" graphics. The important questions are:

1. What mechanisms are used for determining the current (per monitor) DPI (or scaling percentage) and how is this information exposed through upp?
2. How do I create a Ctrl that works at the native "pixel-for-pixel" resolution regardless the current DPI?

I agree, this is far better than the microsoft way of scaling the already rendered bitmaps.

As I understand it, this automatic scaling is an attempt to remove the concept of pixel as a measure of distance and replace it with a roughly one quarter of a millimeter unit (more precisely one pixel at 96 dpi).

I bet I will use Painter extensively to render correctly scaled graphics within my controls with graphical content. The Painter rendered content will obviously need to be rendered at 1:1 pixel ratio on screen.

--

As a general note, I think the solution should be engineered having potential future platforms (such as tablets and phones) in mind, so that their scaling requirements do not require redesign of the solution.

Best regards,

Tom

Subject: Re: Native DPI
Posted by mirek on Mon, 15 Jun 2015 12:01:58 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 15 June 2015 00:00Hi,

From a wider point of view, this sounds like the right way to go. Applications with standard controls scale up beautifully in general without any additional effort, and that is important.

However, from my narrow point of view, I additionally need to be sure that I can still have the low level access to "pixel for pixel" graphics. The important questions are:

1. What mechanisms are used for determining the current (per monitor) DPI (or scaling percentage) and how is this information exposed through upp?
2. How do I create a Ctrl that works at the native "pixel-for-pixel" resolution regardless the current DPI?

We we still speaking theoretically, but the answers would be:

1. Not yet 100% sure... :)
2. You call something like "HiDPIAware();" in constructor.

Quote:
As I understand it, this automatic scaling is an attempt to remove the concept of pixel as a measure of distance and replace it with a roughly one quarter of a millimeter unit (more precisely one pixel at 96 dpi).

I would not mix any physical resolution into it (I never considered the "dpi" concept viable for display). For non-aware Ctrl, one "logical" pixel would equal to two "physical" pixels, that is all.

(OK, I know that this might burn us again if "quadruple HiDPI" ever happens... :)

Quote:
I bet I will use Painter extensively to render correctly scaled graphics within my controls with graphical content. The Painter rendered content will obviously need to be rendered at 1:1 pixel ratio on screen.

Yes.

Mirek

Subject: Re: Native DPI
Posted by Tom1 on Tue, 16 Jun 2015 20:13:26 GMT
View Forum Message <> Reply to Message

Yes, I know: When we talk about Windows, we should avoid physical units -- meters or inches. And when we ask Windows for display DPI, we in fact ask for display scaling percentage. The result 96 "DPI" represents 100 % GUI scaling.

Instead of having just a hidpi flag, how about having a display zoom percentage instead, i.e. 100% for default DPI level and 125, 150, 200 for greater zoom levels, based on what DPI Windows reports. This would allow better control of GUI scaling whenever it is possible, like it is when text is rendered. (And we could drop the term DPI while it is misleading with its false reference to real inches.)

AND: What exactly should be used as the size reference for GUI scaling anyway? Is it the StdFontSize derived from Windows or is it the above zoom percentage derived from monitor DPI, or some strange combination thereof?

Any which way it is, I would prefer having the framework supporting any scaling levels with fine granularity. While icons may cleanly only be scaled to 1x, 2x, 3x, etc., many other aspects of GUI can benefit from greater granularity.

Best regards,

Tom

Subject: Re: Native DPI
Posted by mirek on Tue, 16 Jun 2015 21:37:04 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 16 June 2015 22:13Yes, I know: When we talk about Windows, we should avoid physical units -- meters or inches. And when we ask Windows for display DPI, we in fact ask for display scaling percentage. The result 96 "DPI" represents 100 % GUI scaling.

Instead of having just a hidpi flag, how about having a display zoom percentage instead, i.e. 100% for default DPI level and 125, 150, 200 for greater zoom levels, based on what DPI Windows reports. This would allow better control of GUI scaling whenever it is possible, like it is when text is rendered. (And we could drop the term DPI while it is misleading with its false reference to real inches.)

It is "hidpi aware" flag...

Quote:
AND: What exactly should be used as the size reference for GUI scaling anyway? Is it the StdFontSize derived from Windows or is it the above zoom percentage derived from monitor DPI, or some strange combination thereof?

Now this is a good question (one I am thinking about since the last exchange). Somehow

StdFontSize must be different based on HiDPI-awareness...

Quote:
Any which way it is, I would prefer having the framework supporting any scaling levels with fine granularity. While icons may cleanly only be scaled to 1x, 2x, 3x, etc., many other aspects of GUI can benefit from greater granularity.

I am not quite sure what you are speaking about here... Non-HIDPI-aware widgets does not mean all is fixed.

Mirek

Subject: Re: Native DPI
Posted by Tom1 on Wed, 17 Jun 2015 08:02:22 GMT
View Forum Message <> Reply to Message

Hi,

Sorry, my explanation on "hidpi aware" flag vs. "display zoom percentage" was misleading even to my self after six hours of sleep... Let's try again:

- When "Hidpi aware" flag is set, the coordinates on the Ctrl are equal to physical pixel coordinates. When not set, the Ctrl exposes logical coordinates, which are automatically mapped to physical pixel coordinates (and back) internally in upp according to "display zoom percentage". This will enable compatibility for existing (non-hidpi-aware) Ctrls.

- The "hidpi aware" Ctrls will need additional support from upp to scale their content correctly themselves. This additional support should include a function call to retrieve current display zoom percentage and a function call to retrieve current StdFontSize. Both are needed to get clean results.

In Windows the default font size can be adjusted separately. Please take a look at Control Panel > All Control Panel Items > Display: There you can see "Change the size of all items". The Smaller/Larger -slider changes the Windows reported "DPI". Additionally, on the same page you can see "Change the text size only". A pair of drop lists is used to adjust font size for different text items. It seems TheIDE uses the font size derived from "Menus" font size.

--

When I wrote: "Any which way it is, I would prefer having the framework supporting any scaling levels with fine granularity. While icons may cleanly only be scaled to 1x, 2x, 3x, etc., many other aspects of GUI can benefit from greater granularity.", I meant: "Hidpi aware" Ctrls should have access to the current "display zoom percentage" value in order to be able to scale their contents equally cleanly for any display zoom percentage (100/125/150/200%). Scaling to just 1x, 2x, 3x... is not enough.

Best regards,

Tom

Subject: Re: Native DPI
Posted by mirek on Fri, 19 Jun 2015 06:26:29 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 17 June 2015 10:02
When I wrote: "Any which way it is, I would prefer having the framework supporting any scaling levels with fine granularity. While icons may cleanly only be scaled to 1x, 2x, 3x, etc., many other aspects of GUI can benefit from greater granularity.", I meant: "Hidpi aware" Ctrls should have access to the current "display zoom percentage" value in order to be able to scale their contents equally cleanly for any display zoom percentage (100/125/150/200%). Scaling to just 1x, 2x, 3x... is not enough.

I guess there is nothing that could prevent hda (hidpi-aware) widgets to use everything.... Anyway, I believe that most of time, scaling to correct standard font height will be enough. We can certainly add something to read display zoom percentage too.

That said, GetStdFont function will become tricky. I have even ugly but perhaps bulletproof plan to have thread local variable that would switch what GetStdFont returns as height... (so that I can setup it before starting Draw method).

Subject: Re: Native DPI
Posted by Tom1 on Mon, 22 Jun 2015 09:52:26 GMT
View Forum Message <> Reply to Message

Hi,

OK, it sounds like you have it all figured out already. Please let me hear of any progress on the subject.

Thanks,

Tom

Subject: Re: Native DPI
Posted by Tom1 on Fri, 26 Jun 2015 07:13:29 GMT
View Forum Message <> Reply to Message

Hi,

I just took a look at Windows 10 Pro Insider Preview. It is worth noting that the display scaling range there is 100.. 350% from the default settings and 100.. 500% when accessed through advanced "Custom scaling level" dialog. So it seems Microsoft is already preparing for beyond UHD era already.

Best regards,

Tom

Subject: Re: Native DPI
Posted by mirek on Mon, 03 Aug 2015 18:11:54 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 22 June 2015 11:52Hi,

OK, it sounds like you have it all figured out already. Please let me hear of any progress on the subject.

Thanks,

Tom

Well, I have now started working hard(er) to this. In the end, I have abandoned my "automatic scaling" idea - it in fact was not that hard to 'fix' CtrlLib and TheIDE for HiDPI, so I guess any application can do that as well. So in the end, we will probably follow Microsoft mode, setting "DPIAware" by application (or its manifest), otherwise use windows scaling.

Please check how TheIDE now looks in HiDPI. Image rescaling is still primitive, I hope to employ some smart algorithm for this soon. However, except the visual quality of images, all other glitches should be now reported, filed in RM and fixed.

As you are perhaps the only one with UHD display, I really hope to hear from you... :) (I was developing using 200% text size on 1080p display...)

Thanks,

Mirek

Subject: Re: Native DPI
Posted by Tom1 on Wed, 05 Aug 2015 11:03:06 GMT
View Forum Message <> Reply to Message

Hi,

Thanks for the update on this! Here are my observations starting from the installation of build 8788 on Windows 8.1 Professional x64 using 200 % display scaling:

- The 'fine print' on the installer License agreement dialog is truly fine print. It should be about twice as large for 200 % scaling factor. The same is true for the Installation guide dialog.

- The down pointing triangle-arrow on the droplist button seems very small. (The right pointing arrows on similar controls are the right size.)

- The code editor font is very small. I think it should be scaled accordingly regardless of the actual point size in user configurable settings. The same is true for the help content.

- The Abbreviations code editor font is too small.

- The font in Custom build steps dialog's 'Commands:' and 'Output file(s):' panels is too small.

- The 'SvnSynchronize SVN repositories' dialog's List's first line with 'Working directory' text is too flat. The 'Working directory' text is clipped at the lower edge.

- The 'Find in files' dialog's entry fields / drop lists have down and right pointing triangular arrows. They are all too small.

That's all I could find at this time.

Thanks and best regards,

Tom

PS. Also the Console font (below the code editor in ide) is very small and so are the icons on the right edge of the console too.

Subject: Re: Native DPI
Posted by mirek on Wed, 05 Aug 2015 12:56:54 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 05 August 2015 13:03
- The code editor font is very small. I think it should be scaled accordingly regardless of the actual point size in user configurable settings. The same is true for the help content.

I am 'relatively unsure' about this... I was sort of aware about the problem; my initial solution was to only about default value.... as values displayed in "Environment" dialog are supposed to be in pixels....

But maybe you are right.

Subject: Re: Native DPI
Posted by Tom1 on Wed, 05 Aug 2015 13:50:32 GMT
View Forum Message <> Reply to Message

BTW, the Ctrl + mouse wheel (present in TheIDE code editor and also in many web browsers these days) is an excellent way around wrong font size in the UHD environment. However, I would still prefer having it the right size immediately from the installation -- just like it is with most of the Ctrls already.

UPDATE: Maybe the environment dialog could show the font size in percentage compared to the StdFontSize, which is used for e.g. menus. This way it will scale correctly but still offer full configurability.

Tom

Subject: Re: Native DPI
Posted by mirek on Sun, 09 Aug 2015 11:22:37 GMT
View Forum Message <> Reply to Message

All listed issues should be now fixed.

W.R.T. ide editor font size, I have decided to use "default font" approach - after install, height is defaulted based on zooming (-> OK), but it does not change when system DPI changes.

Notes about "UHD" support, current model:

- Image now has "resolution" attribute, which can be either Standadard, UHD, None

- When Image is created in code, this defaults to None

- IML images have default "standard"

- IML images, if not NONE, are automatically rescaled if mode does not match

- There are functions DPI that can be used to scale int, size or Image based on UHD mode

- Ctrl::SetHiDPIEnabled (maybe I will rename to SetUHDEnabled...) has to be used to activate Windows HiDPI mode.

- Ctrl::[Horz|Vert]LayoutZoom have now global function synonyms Zx and Zy (simply beacause they get called a lot...:) (Also, there is Zsz returning Size

Looking forward for more bugs to fix... :)

Subject: Re: Native DPI

Posted by Tom1 on Mon, 10 Aug 2015 09:12:59 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Thanks for the fixes. Here are some more things to look at :)

- One thing I missed on the first test run when installing is the Directory selector dialog for "Select the directory for MyApps". The listings for places and also files/directories overlaps the below 'status line' showing file properties. How much, depends on resizing the dialog vertically.

- The default font size in the e.g. code editor is now much better. However, the place where the font sizes are set up (Format setup), there is font size defined only for three of the six fonts. Others are blank:

- The Abbreviations code editor font was too small. Now it seems a bit too large -- not that it really matters, but could it follow the 'Normal' font specified in Format setup?

- The font in Custom build steps dialog's 'Commands:' and 'Output file(s):' panels was too small. Now it seems a bit too large -- not that it really matters, but could it follow the 'Normal' font specified in Format setup?

Next I will take a look at what effect all this has on my applications when compiled using 8808. This may take a while...

Thanks for all your work on this.

Best regards,

Tom

File Attachments
1) upp8808.png, downloaded 771 times

Subject: Re: Native DPI
Posted by Tom1 on Thu, 13 Aug 2015 10:22:55 GMT
View Forum Message <> Reply to Message

Hi Mirek,

The last minor thing that does not look good is the 'Select folder' dialog. As you can see the places and folder content panels overlap the 'status line' containing information about the content:

Otherwise, in my opinion, it works beautifully with UHD now.

--

My secondary Windows-scaled 24" FHD monitor on the other hand looks pretty bad with just about anything on it: There are not many programs in fact that can handle the situation correctly. I know per-monitor HiDPI support is not your top priority right now, but when the time comes, please let me know.

Thanks and best regards,

Tom

File Attachments
1) upp8813.png, downloaded 816 times

Subject: Re: Native DPI
Posted by mirek on Tue, 18 Aug 2015 18:10:28 GMT
View Forum Message <> Reply to Message

Should be now fixed. I have also replaced scaling algorithm with "smart" one (based on xbr), hopefully it will help a little...

Mirek

Subject: Re: Native DPI
Posted by Tom1 on Wed, 19 Aug 2015 11:07:55 GMT
View Forum Message <> Reply to Message

Hi,

Thanks! The directory selector works as expected now.

Additionally, up-scaled icons surely look sharper too.

I wonder if it would be possible to use 256x256 icons for all purposes now? I tried to set icon for my TopWindow (using Icon()), but the downscaling does not work as expected. It feels like there is no antialias filtering present before down sampling. If the downscaling algorithm is under U++ control, could it be changed to use some antialias/low-pass filtering algorithm to give better results?

One slightly unrelated question: Why has the flag parameter FD_ZERO for FormatDoubleFix()

changed to FD_ZEROS some time since the last release?

Best regards,

Tom

Subject: Re: Native DPI
Posted by mirek on Wed, 19 Aug 2015 20:35:38 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 19 August 2015 13:07Hi,

Thanks! The directory selector works as expected now.

Additionally, up-scaled icons surely look sharper too.

I wonder if it would be possible to use 256x256 icons for all purposes now?

Unlikely...

Quote:
I tried to set icon for my TopWindow (using Icon()), but the downscaling does not work as expected. It feels like there is no antialias filtering present before down sampling. If the downscaling algorithm is under U++ control, could it be changed to use some antialias/low-pass filtering algorithm to give better results?

Depends on what you have really done there... Have to say, situation there is really fuzzy for me (because it also depends on what Windows think about icons).

Quote:
One slightly unrelated question: Why has the flag parameter FD_ZERO for FormatDoubleFix() changed to FD_ZEROS some time since the last release?

Working on new C++ parser, I have found a nameclash with FD_ZERO posix macro... (see 'select').

Mirek

Subject: Re: Native DPI
Posted by Tom1 on Sat, 22 Aug 2015 07:56:14 GMT
View Forum Message <> Reply to Message

Ok, I see.

Thanks and best regards,

Tom

Subject: Re: Native DPI
Posted by mirek on Sun, 23 Aug 2015 10:23:46 GMT
View Forum Message <> Reply to Message

UHD is now supported in Gtk backend too...

Mirek

Subject: Re: Native DPI
Posted by Tom1 on Sat, 29 Aug 2015 11:54:22 GMT
View Forum Message <> Reply to Message

Hi Mirek,

While still on Windows UHD: I just installed 8862 (after using 8817 for a while) and found that the menu items on the main menu are now better vertically centered than before. However, the vertical centering of 'Output mode' text (e.g. MSC9 Debug) is too much on the low side now: The letter 'g' of word Debug gets clipped on the lower edge and the texts appear to be a bit too low compared to horizontal center.

BTW: Where can I find up to date information comparable to that in "https://code.google.com/p/upp-mirror/source/list", which I have used until now?

Best regards,

Tom

Subject: Re: Native DPI
Posted by mirek on Sat, 29 Aug 2015 16:26:30 GMT
View Forum Message <> Reply to Message

Could post a screenshot please? (Perhaps crop it a little so that it does not widen the forum...)

As for svn, you can use the main svn now (svn://www.ultimatepp.org/upp), then svn log...

(I will reestablish mirror soon).

Mirek

Subject: Re: Native DPI
Posted by Tom1 on Mon, 31 Aug 2015 08:24:22 GMT
View Forum Message <> Reply to Message

Sure,

Thanks!

Here's the image:

Best regards,

Tom

File Attachments
1) 8862-Debuq.png, downloaded 688 times

Subject: Re: Native DPI
Posted by Tom1 on Mon, 31 Aug 2015 12:38:50 GMT
View Forum Message <> Reply to Message

I just took a look at a fresh TheIDE on Linux Mint Cinnamon: It appears that the radio buttons and check boxes on dialogs do not scale to UHD.

Best regards,

Tom

Subject: Re: Native DPI
Posted by mirek on Wed, 02 Sep 2015 14:11:55 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 31 August 2015 10:24Sure,

Thanks!

Here's the image:

Best regards,

Tom

I am having trouble to reproduce this... I am testing by increasing font size to 200%, both in win8 and win10 - all looks ok.

What are your settings?

Subject: Re: Native DPI
Posted by Tom1 on Thu, 03 Sep 2015 07:07:23 GMT
View Forum Message <> Reply to Message

Mirek,

This is strange... I'm running on Win 8.1 pro 64 and whatever is the standard system font on Windows, is still active. I have not changed it ever.

I had 200% selected (while the recommended value was 250%). Now I tried with various values and here's the result:

- 100% and 125% work OK
- 150%, 200%, 250% and 300% all result in clipped text.

One thing I noticed is that the drop down lists on TheIDE toolbar are not as high as normal drop down lists in many dialogs. Could this have any contribution to the problem?

Best regards,

Tom

Subject: Re: Native DPI
Posted by mirek on Fri, 04 Sep 2015 06:51:31 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Thu, 03 September 2015 09:07Mirek,

This is strange... I'm running on Win 8.1 pro 64 and whatever is the standard system font on Windows, is still active. I have not changed it ever.

I had 200% selected (while the recommended value was 250%). Now I tried with various values and here's the result:

- 100% and 125% work OK
- 150%, 200%, 250% and 300% all result in clipped text.

One thing I noticed is that the drop down lists on TheIDE toolbar are not as high as normal drop

down lists in many dialogs. Could this have any contribution to the problem?

Best regards,

Tom

Thanks, that nailed it... :) It was visible only in "menu and toolbar in a row" (which I usually have ON, but accidentally had OFF during testing).

Should be now fixed.

I still have to investigate GTK issues, however, scaling of host platform widgets is left on host platform. Can you check please how other GTK applications look? Do they have it scaled?

Mirek

Subject: Re: Native DPI
Posted by Tom1 on Fri, 04 Sep 2015 08:10:51 GMT
View Forum Message <> Reply to Message

Hi,

Yes, I confirm: The drop down lists on the menu bar are fixed now. Thanks!

On the GTK issue: I opened Gimp and found out that scaling of GTK widgets is really bad or totally absent. So, it is a problem in GTK -- not U++. Changing 'Gui theme' to something else than 'Host platform' gives properly scaled widgets in TheIDE.

Still on UHD: Would it be possible to make PromptOK() -- and similar dialogs -- larger in UHD context, in order to fit the larger text better inside the popup dialog? Currently lines that would have previously fitted on one line are split to two lines making the dialog look a bit weird. Slightly longer messages extend to multiple lines and invoke a scroll bar to allow reading the full content. This would not be necessary in many cases if the dialog was scaled according to the font size.

Thanks and best regards,

Tom

Subject: Re: Native DPI

Posted by mirek on Sun, 06 Sep 2015 05:35:27 GMT
View Forum Message <> Reply to Message

Prompt issue fixed.

Mirek

Subject: Re: Native DPI
Posted by Tom1 on Mon, 07 Sep 2015 08:55:02 GMT
View Forum Message <> Reply to Message

Much better now!

Thanks and best regards,

Tom

Subject: Re: Native DPI
Posted by mirek on Sun, 04 Oct 2015 06:07:50 GMT
View Forum Message <> Reply to Message

One last small change: Ctrl::SetHiDPIEnabled -> CtrlUHDEnabled (to make naming consistent). Also, some documentation now added.