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
