Subject: Using Qtf forgets control font
Posted by cbpporter on Mon, 07 Jul 2008 12:04:08 GMT
View Forum Message <> Reply to Message

I have a simple test case here to illustrate the problem. I have a layout with two labels, and both have their label set to the same text, only the second one gets a "\1" prepended to enable Qtf and allow word wrapping.

The second label will forget the font that it's supposed to use, and thus, under Linux, with the current font problems, will simply not display anything since the font that the Qtf defaults to does not have the character. A workaround could be to give the font name again in the Qtf, but shouldn't this "forgetting" be fixed? I'll try to figure out what this would imply.

File Attachments
1) Test.zip, downloaded 716 times

Subject: Re: Using Qtf forgets control font
Posted by mirek on Mon, 07 Jul 2008 13:30:33 GMT
View Forum Message <> Reply to Message

If you use QTF in label, font is ignored. (-> a feature, not a bug

Mirek

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Mon, 07 Jul 2008 14:53:24 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 07 July 2008 16:30If you use QTF in label, font is ignored. (-> a feature, not a bug

Mirek
Strange feature. A little counter-productive, but I guess I can compensate with extra Qtf. I tried something like this:
lblKun.SetFont(f).SetText("\1[!" + f.GetFaceName() + "! [3 " + ToUtf8(Join(kanji_.KunReading, EntrySeparator)) + "]]");
only to find out that using [3 ...] for 12 points is different from the StdFont height. Is there a function that takes a font height and return the Qtf magical number needed to obtain that size?

And is there a Qtf tag to disable font antialliasing? I couldn't find one.

Or even better, is there another way to enable word-wrapping.

Subject: Re: Using Qtf forgets control font
Posted by mirek on Mon, 07 Jul 2008 20:51:03 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Mon, 07 July 2008 10:53luzr wrote on Mon, 07 July 2008 16:30If you use QTF in label, font is ignored. (-> a feature, not a bug

Mirek
Strange feature. A little counter-productive, but I guess I can compensate with extra Qtf. I tried something like this:
lblKun.SetFont(f).SetText("\1[!" + f.GetFaceName() + "! [3 " + ToUtf8(Join(kanji_.KunReading, EntrySeparator)) + "]]");
only to find out that using for 12 points is different from the StdFont height. Is there a function that takes a font height and return the Qtf magical number needed to obtain that size?

[/quote]

Unfortunately, things are a little bit more complicated (as usual). QTF is basically "physical unit format" (intended to be eventually printed). The unit is "dot" - 1/600 of inch. However, for screen displaying, there is zooming coeficient that is derived from Font-zoom.

Font "numbers" are tables of dots, but you can express the font height if you prepend it with '+', like "[+80 ...".

Mirek

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Tue, 08 Jul 2008 11:47:47 GMT
View Forum Message <> Reply to Message

Sorry, but Qtf is more trouble than it's worth. I'll fork label and rewrite Paint so that it uses manual word-wrapping.

Subject: Re: Using Qtf forgets control font
Posted by zsolt on Tue, 08 Jul 2008 12:38:39 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Tue, 08 July 2008 13:47Sorry, but Qtf is more trouble than it's worth. I'll fork label and rewrite Paint so that it uses manual word-wrapping.

I can't agree. QTF is one of the most useful features.

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Tue, 08 Jul 2008 13:28:02 GMT
View Forum Message <> Reply to Message

zsolt wrote on Tue, 08 July 2008 15:38cbpporter wrote on Tue, 08 July 2008 13:47Sorry, but Qtf is more trouble than it's worth. I'll fork label and rewrite Paint so that it uses manual word-wrapping.

I can't agree. QTF is one of the most useful features.
I'm sure it is really useful in printing and quick fancy formated outputs. Yet it can not handle a simple thing like "take all the formating info out of that font and use it". Maybe if we define some kind of a mechanism similar to the one that allows the usage of picture inside Qtf (which is a great feature BTW)...

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Fri, 29 Aug 2008 19:09:54 GMT
View Forum Message <> Reply to Message

I'm trying to implement something regarding DPI/Font to Qtf and I need to determine screen DPI. Something like Draw::GetPixelsPerInch seems a good place to start, but first of all I can only call it on a Draw instance, and even if I do, I get a failed assert regarding IsDrawing. Using it on ScreenInfo also seemed like a good idea, but I can't use it or find it's definition. All I can find is about 6 such declarations scattered around:
Draw& ScreenInfo();

Also, could someone point me in the direction where GUI scaling is done, so I can understand the way sizes scale with higher DPI. Is it CurDPI / 72 * intended size or something similar?

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Fri, 29 Aug 2008 20:39:38 GMT
View Forum Message <> Reply to Message

Also, WindXP is quite helpful in giving you this handy tip:
Quote:If your screen resolution makes items too small to view comfortably, you can increase the DPI to compensate.
Is it just me, or is this statement completely backwards. Why would increasing the DPI make any text larger. That doesn't make any sense to me . Am I missing something?

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Sat, 30 Aug 2008 11:43:33 GMT
View Forum Message <> Reply to Message

I think I understand now. When you are printing on paper, the size of an inch is constant. Thus increasing DPI, makes text smaller.

But on screen, DPI is considered PPI and the pixel size for a resolution is considered constant. By altering DPI, you actually change the size of an inch. So by increasing DPI, you increase the "virtual" inch size, and thus text gets larger.

Subject: Re: Using Qtf forgets control font
Posted by mirek on Sat, 30 Aug 2008 12:05:26 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Sat, 30 August 2008 07:43I think I understand now. When you are printing on paper, the size of an inch is constant. Thus increasing DPI, makes text smaller.

But on screen, DPI is considered PPI and the pixel size for a resolution is considered constant. By altering DPI, you actually change the size of an inch. So by increasing DPI, you increase the "virtual" inch size, and thus text gets larger.

Welcome to the real world.

That is why I keep saying that points and DPI for screen are largely irrelevant. What only matters is zoom ratio.

Mirek

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Sat, 30 Aug 2008 12:10:41 GMT
View Forum Message <> Reply to Message

luzr wrote on Sat, 30 August 2008 15:05
Welcome to the real world.

Thank you !

Quote:
That is why I keep saying that points and DPI for screen are largely irrelevant. What only matters is zoom ratio.

Still, there are still two issues left:
1. Even though DPI is irrelevant for screens, and computations are wrong, I still want to use the same computations as the rest of the world. My size x font from U++ must be pixel-by-pixel identical to the same font in Notepad or a window title bar.
2. I still need to know how to use ScreenInfo and GetPixelsPerInch.

Subject: Re: Using Qtf forgets control font

Posted by mirek on Sat, 30 Aug 2008 12:53:23 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Sat, 30 August 2008 08:10
Still, there are still two issues left:
1. Even though DPI is irrelevant for screens, and computations are wrong, I still want to use the same computations as the rest of the world. My size x font from U++ must be pixel-by-pixel identical to the same font in Notepad or a window title bar.

Definitely. It is. That is what Ctrl::FontZoom is for.

Quote:
2. I still need to know how to use ScreenInfo and GetPixelsPerInch.

Sure. What is the problem?

Mirek

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Sat, 30 Aug 2008 20:08:23 GMT
View Forum Message <> Reply to Message

luzr wrote on Sat, 30 August 2008 15:53Definitely. It is. That is what Ctrl::FontZoom is for.

I can't find Ctrl::FontZoom.

And:
Quote:Something like Draw::GetPixelsPerInch seems a good place to start, but first of all I can only call it on a Draw instance, and even if I do, I get a failed assert regarding IsDrawing. Using it on ScreenInfo also seemed like a good idea, but I can't use it or find it's definition. All I can find is about 6 such declarations scattered around

---

Subject: Re: Using Qtf forgets control font
Posted by mirek on Sun, 31 Aug 2008 08:46:40 GMT
View Forum Message <> Reply to Message

Sorry, "FontZoom" is "general term". The methods are

Ctrl::

```
static const char *GetZoomText();
static void SetZoomSize(Size sz, Size bsz = Size(0, 0));
static int  HorzLayoutZoom(int cx);
```

```
static int  VertLayoutZoom(int cy);
static Size LayoutZoom(int cx, int cy);
static Size LayoutZoom(Size sz);
static void NoLayoutZoom();
static void GetZoomRatio(Size& m, Size& d);

static int  HZoom(int cx)                       { return HorzLayoutZoom(cx); }
```

The idea behind this is that there is some basic font size. All dialog layouts are designed with this basic font size in mind - and these are units shown in layout designer.

Then there is "GetZoomText" - a standard text whose size is known in basic font size (by basic font we consider here the "normal font" used for most texts in GUI).

Then, when U++ app is started, the same text is measured in host platform "normal GUI font" and this new size / "standard size" is taken as "FontZoom ratio".

This is the best solution I was able to invent... In practice, it works pretty well.

Mirek

Subject: Re: Using Qtf forgets control font
Posted by mirek on Sun, 31 Aug 2008 08:51:45 GMT
View Forum Message <> Reply to Message

I had to deprecate ScreenInfo, as it is inherently MT unsafe...

You should be able to get the same thing using

ScreenDraw info(true);

Mirek

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Sun, 31 Aug 2008 11:39:46 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 31 August 2008 11:46Sorry, "FontZoom" is "general term". The methods are
The idea behind this is that there is some basic font size. All dialog layouts are designed with this basic font size in mind - and these are units shown in layout designer.

Then there is "GetZoomText" - a standard text whose size is known in basic font size (by basic font we consider here the "normal font" used for most texts in GUI).

Then, when U++ app is started, the same text is measured in host platform "normal GUI font" and this new size / "standard size" is taken as "FontZoom ratio".

This is the best solution I was able to invent... In practice, it works pretty well.

OK, basically you have Csize where you get the current size using StdFont of the zoom text, and Dsize, which is initialized with a hardcoded measurement of the same text and same font under what you consider zoom 1:1 zoom ration. And you use their ration to adjust positions.

Then you must know what which font StdFont() is .

I also noticed that max between Csize and Dsize. This probably explains why when I lowered my PPI and I go really small text, the layout was not shrunk, but when I increase PPI, layouts grow..

Quote:
I had to deprecate ScreenInfo, as it is inherently MT unsafe...

You should be able to get the same thing using

ScreenDraw info(true);

ScreenDraw worked fine. Thank you! Actually, I was getting the assert because I used PromptOK in a Paint method. I shouldn't have done that.

Can I use ScreenDraw to draw on the screen as with Win API GetDC(0)? If yes, then probably I should extract the info I need at application startup and then discard the ScreenDraw instance.

I think I have everything that I need to add HeightPt() and GetHeightPt() to Font.

Also, except font antialiasing, is there any other missing tag from Qtf that is present in a font description?

Subject: Re: Using Qtf forgets control font
Posted by mirek on Mon, 01 Sep 2008 11:37:44 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Sun, 31 August 2008 07:39

Can I use ScreenDraw to draw on the screen as with Win API GetDC(0)?

That is ScreenDraw with "false". "true" means "information context" - read only.

Quote:
If yes, then probably I should extract the info I need at application startup and then discard the ScreenDraw instance.

Yep.

Quote:
Also, except font antialiasing, is there any other missing tag from Qtf that is present in a font description?

IMO, no. In fact, "antialiasing" request sort of surprised me... OTOH, why not, there might be a use for it.

Mirek

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Mon, 01 Sep 2008 12:46:52 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 01 September 2008 14:37
IMO, no. In fact, "antialiasing" request sort of surprised me... OTOH, why not, there might be a use for it.

There are some CJK fonts under Linux which are completely unreadable with anti-aliasing, unless you use huge font size, where anti-aliasing doesn't really help in general. Also, since U++ kind of forces you to use Qtf for formated display, I think it makes no sense for it not to support all attributes of font.

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Thu, 04 Sep 2008 03:08:51 GMT
View Forum Message <> Reply to Message

I added HeightPt to Font. Or should it be HeightPoints now, so to not overuse abbreviations? If it will be accepted I'll add the possibility to choose the sizing unit to TheIDE's font properties dialog from the layout designer so that I can set up all my layouts to use points. If not, I'll leave it as it is because it is too much effort to maintain a TheIDE fork.

Subject: Re: Using Qtf forgets control font
Posted by mirek on Sun, 07 Sep 2008 11:35:36 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Wed, 03 September 2008 23:08I added HeightPt to Font. Or should it be HeightPoints now, so to not overuse abbreviations?

Ehm, after all these posts about points being meaningless for screen?

On what device that point is defined?

Quote:
If it will be accepted I'll add the possibility to choose the sizing unit to TheIDE's font properties dialog from the layout designer so that I can set up all my layouts to use points.

That's Pandora's box, believe me.

Mirek

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Sun, 07 Sep 2008 12:08:45 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 07 September 2008 14:35
Ehm, after all these posts about points being meaningless for screen?

Well my problem is simple. I have Font selection dialog. User chooses size 16 for example. In U++ size is in pixels. User chooses size 16 in different program. Size is in points. User complains that I have bug with font sizes . After a simple investigation, it seems that this is what is needed to get same size:

```
Font& Font::HeightPt(int pt)
{
    height = pt * sScreenPPI / 72;
    return *this;
}
```

Not extensively tested, but seems to generate pixel perfect matches.

Quote:
That's Pandora's box, believe me.

I'll assume that you don't want it opened. Anyway, after all other task are done, the property browser could also be improved and (dare I say it) made more like the Delphi/VS one.

PS: something is messed up with Assist/Navigate.. in SVN. All items appear about 15 times.

Subject: Re: Using Qtf forgets control font
Posted by mirek on Sun, 07 Sep 2008 18:47:41 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Sun, 07 September 2008 08:08luzr wrote on Sun, 07 September 2008 14:35
Ehm, after all these posts about points being meaningless for screen?

Well my problem is simple. I have Font selection dialog. User chooses size 16 for example. In U++ size is in pixels. User chooses size 16 in different program. Size is in points. User complains that I have bug with font sizes . After a simple investigation, it seems that this is what is needed to get same size:

```
Font& Font::HeightPt(int pt)
{
    height = pt * sScreenPPI / 72;
    return *this;
}
```

Not extensively tested, but seems to generate pixel perfect matches.

That is certainly reasonable requirement, but makes me really wonder whether you need a method for this.

IMO, simple function (PointsToPixels) would be enough.

Mirek

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Tue, 09 Sep 2008 00:15:17 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 07 September 2008 21:47
IMO, simple function (PointsToPixels) would be enough.

I guess a simple function will do too. I used a method to ease the transition from pixels to points.

```
int PointsToPixels(int points)
{
 static int screenPPI = ScreenDraw(false).GetPixelsPerInch().cy;
 return points * screenPPI / 72;
}
```

Normally this would mean a lot of extra typing to replace the sizes of all GUI elements. I guess I'll have to use styles.

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Tue, 09 Sep 2008 01:14:33 GMT
View Forum Message <> Reply to Message

I'm afraid that double return values are needed for such conversions. Otherwise I get rounding errors when I'm trying to build Qtf as in the example.

Edit: or maybe a conversion table.

File Attachments
1) FontSize.rar, downloaded 372 times

Subject: Re: Using Qtf forgets control font
Posted by mirek on Tue, 09 Sep 2008 06:53:17 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Mon, 08 September 2008 21:14I'm afraid that double return values are needed for such conversions. Otherwise I get rounding errors when I'm trying to build Qtf as in the example.

Edit: or maybe a conversion table.

Well, to the API it goes always as integer (in Win32 and X11). Means that you can do rounding in the function.

Mirek

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Wed, 10 Sep 2008 04:07:10 GMT
View Forum Message <> Reply to Message

I tested under Win and rounding seems OK but must be done in all conversion functions. Must test under Linux too.

I noticed that for testing purposes is is a fair assessment to consider StdFont as Arial. Does this apply to X11 also? And if for some strange reason the only font that I have is Wingdings, will StdFont default to Wingdings?

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Wed, 10 Sep 2008 06:08:44 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Wed, 10 September 2008 07:07Must test under Linux too.

It appears that ScreenDraw is not defined under Linux. Now do you understand why I think that a cross platform framework like U++ should do everything in it's power to stop users from using system dependent functionality.

We need a cross platform way of getting screen PPI and hide ScreenDraw better.

Subject: Re: Using Qtf forgets control font
Posted by mirek on Wed, 10 Sep 2008 18:46:04 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Wed, 10 September 2008 00:07I tested under Win and rounding seems OK but must be done in all conversion functions. Must test under Linux too.

I noticed that for testing purposes is is a fair assessment to consider StdFont as Arial.

Nope, I Win32 it is taken from the system, in XP it is Tahoma (in older Win, it is MS Sans Serif).

Quote:
Does this apply to X11 also?

It is also taken in X11. Anyway, in X11, "system" font usually is the same font chosen for "Arial".

Quote:
And if for some strange reason the only font that I have is Wingdings, will StdFont default to Wingdings?

If Wingdings are set as system font, then yes:)

However, you would not be able to start U++ app without "three basic fonts" - Arial, Roman, Courier. On Win32 they are required. (and on X11, equivalents are always defined in fontconfig).

Mirek

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Wed, 10 Sep 2008 19:27:08 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 10 September 2008 21:46
Nope, I Win32 it is taken from the system, in XP it is Tahoma (in older Win, it is MS Sans Serif).

I assumed that because I had a qtf with arial and one without any font specified and they were the same. It seems than not specifying font in qtf makes it use Arial, not StdFont.

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Wed, 10 Sep 2008 20:02:47 GMT
View Forum Message <> Reply to Message

I no longer understand what's going on.

I have the test up similar to what I uploaded and everything works fine. I choose the size in points, covert to pixels, and everything has correct proportions. I am using a layout.

Then I copy paste the code to my real application, and the text is suddenly too large. My already zoomed text gets zoomed again. The only difference is that the layout that contains the Label is added as s frame with a splitter. If I disable font zooming I get correct results. Why is that frame zoomed, when other layouts are not?

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Thu, 11 Sep 2008 04:56:19 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 10 September 2008 21:46
Nope, I Win32 it is taken from the system, in XP it is Tahoma (in older Win, it is MS Sans Serif).

That is not true on my XP.
See screenshot.
First line is Tahoma from Notepad, second is Stdfont, third is Tahoma from U++ app.

File Attachments
1) Tahoma.PNG, downloaded 1115 times

Subject: Re: Using Qtf forgets control font
Posted by mirek on Fri, 12 Sep 2008 06:25:32 GMT
View Forum Message <> Reply to Message

Well, strange.

The critical question however is: Is the U++ font the same as in other apps? (Since I have fixed the issue couple of years ago, I never seen incorrect font on any windows version I have ever seen U++ app running).

It is possible that your "normal" font is even different. Maybe it is CJK version of Win32?

Anyway, the critical piece of code starts at line 173 of Draw/DrawTextWin32.cpp. Perhaps you can add

DUMP(ncm.lfMenuFont.lfFaceName)

before SetStdFont to see what is going on.

Mirek

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Fri, 12 Sep 2008 09:15:32 GMT
View Forum Message <> Reply to Message

Well the font is "Tahoma". I had problems determining what is wrong until I noticed that it depends on the application you're compiling. My test application works fine in regard to fonts.

But my real application has two problems:
1. Even though StdFont is Tahoma, I have a SplitterFrame with a layout in it where StdFont is not Tahoma somehow.
2. That layout uses bigger font zooming. Normally I have consistent font zooming, but something is wrong here.

And these two problems are IMO closely related. Somehow StdFont gets wacky and causes font zooming to increase, since the font that is used instead of Tahoma is larger.

But I think this problem is specific to me and I somehow introduced a bug somewhere while messing around with fonts. Another hint is that from some point one, all my logs start to have a progressive indentation, and some extra rtiming have started to appear. I really need to debug this.

PS: I came to the conclusion that OpenOffice is a poor model to test fonts against. The fact that it is not so specific too Windows shows, and there are often subtle differences when compared to native font rendering. so my new model is Notepad .

PS:
Quote:
Anyway, the critical piece of code starts at line 173 of Draw/DrawTextWin32.cpp

You know, you could have pointed me here all those times that I asked how do you get the name of StdFont . By reviewing that code I now finally understand how font stuff is determined. Better late than never!

Subject: Re: Using Qtf forgets control font
Posted by mirek on Fri, 12 Sep 2008 15:00:06 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Fri, 12 September 2008 05:15Well the font is "Tahoma". I had problems determining what is wrong until I noticed that it depends on the application you're compiling. My test application works fine in regard to fonts.

But my real application has two problems:
1. Even though StdFont is Tahoma, I have a SplitterFrame with a layout in it where StdFont is not Tahoma somehow.
2. That layout uses bigger font zooming. Normally I have consistent font zooming, but something is wrong here.

I would like to check some testcase to find out what is going wrong...

Mirek

Subject: Re: Using Qtf forgets control font
Posted by cbpporter on Sat, 13 Sep 2008 17:19:07 GMT
View Forum Message <> Reply to Message

I finally found the bug. It is really stupid actually. My fault.

I spent almost two hours removing classes and files one by one so that I could create a small test package. The bug just wouldn't go away, even when all I had were two classes with 5 lines each, just doing some simple layouting.

Then I found it! My bottom frame class was in it's own file and somewhere in that .cpp between function definitions was a:
ScreenDraw info(true);
That little line than must have been left over form earlier testing caused all the exaggerated font zooming. It is really funny actually, if you don't consider the time went into investigating this.

Anyway, false alarm. Need to be more careful with such dangerous thing like ScreenDraw. Now I can get back to solving my real font issues, like finding a replacement for ScreenDraw for starters.
