Subject: Re: Basic character set analyzer
Posted by cbpporter on Sun, 08 Feb 2009 11:45:34 GMT
View Forum Message <> Reply to Message

luzr wrote on Sat, 07 February 2009 15:53
I apologize for delay; HasChar is now officially in Draw, implemented both for Win32 and X11.

Great! I'll check out the win version too, but right now all my efforts focus on X.

Quote:
Well, thinking about the issue - do you have any idea how to determine which fonts search for replacements first?

I mean, for missing char in Arial, I would check some sans-serif font first rather than trying e.g. Times New Roman..

Mirek
I've given this issues a lot of thought, but I don't think there is a good way to do this.

This is why I chose composition rather than substitution.

So basically we have the first 3 Unicode sections which handle Latin. All characters are letters, punctuation and composites. I have basically two cases when a character is missing:
1. Font still has basic Latin characters. In this case I apply composition. I managed to fine tune diacritic placement and results are generally better than when replacing whole character from a different font., but for some size, it will look uglly.
2. Font doesn't have basic Latin. These font are quite rare, and in such cases I use substitution. But these fonts won't really have any Latin support, so I end up using substitution for every character, so font still looks consistent.

But for further Unicode sections the situation changes. These characters start to look less and less Latin, so the problem of substituting with a similar font starts to diminish. I mean, using Arial to draw an 'a' next to a strange fork like character might be a better choice than using Times, but it wont really make a difference. So here I will also use a single font for substitution.

So basically I divide the Unicode range in sub ranges, each containing roughly a single script. And I'll have a vector of vectors of fonts, with first vector indexed by the script number.
Algorithm is in pseudo code:

```
get scriptindex;
if (sciptindex favors composition)
  compose characters
else
  traverse font list until character found; draw box if not found
```

This approach works right now, but since it only handles a small number of scripts, I can't really say yet if it will be enough for real needs of rendering real internationalized texts. Only time will tell.

I attached a screenshot showing the results when using one of the later Latin ranges. Cyan characters are substituted, and also give a nice visual of how average fonts support such characters. Also illustrates problems with using fonts that look ugly one near the other.

Also notice the two lines in Arabic on the right. This is very early support for RTL languages. I give the text in normal left to right order, containing "abc", some Arabic characters, and "def". The text output method detects that the Arabic should be right to left, and renders it as such. First line does plain rendering, second uses specific Arabic rendering rules. While the two texts contain the same characters, the first is wrong and only the second is a correct way to render the characters from the first string.

PS: I can't read Arabic, so I have no idea what I wrote there. I hope I didn't manage to find randomly a swear word or anything like that . I we have users who can read Arabic, I would appreciate some help in this area.

File Attachments

1) snapshot7.png, downloaded 462 times