Subject: Re: Basic character set analyzer
Posted by cbpporter on Fri, 30 Jan 2009 08:59:05 GMT
View Forum Message <> Reply to Message

luzr wrote on Fri, 30 January 2009 10:28
Good, getting somewhere.

I believe the problem with the above font is that it does not have diacritics characters defined.

What is your solution to the problem?

That's exactly the problem. I'm using two way substitution, replacing both base character and diacritic when necessary. Diacritics look similar enough across fonts so no problems here with substitution. For base char, I'm using StdFont. This way at least characters substituted across different fonts will at least look the same, even though the might differ visually from current font. But this is not a big problem for Latin, because most font either have all characters from a range or have none. There are some who have an arbitrary subset of those characters, but I'm guessing that the subset is biased toward an existing language and since people will probably use that font to render that given language, we shouldn't have many cases where substitution result in ugly text.

Quote:
Hm, maybe it is just terminology issue - by composition I mean creating a new glyph by composing two other glyphs (basic latin character + diacritics glyph) from the same font.

That's for Latin composition (which I'm implementing right now). For non-Latin composition, I create a new glyph based on a non-Latin little drawing and combine it with another one (which could be considered a diacritic).

The reasons why I'm not altering your code and witting a new one are:
1. I want to have full composition with arbitrary bases and diacritics to handle all Unicode composition characters. I can place a '-' on something, but I can also place any printable character.
2. I am implementing these methods as functions that take a Draw object as their first parameter so that I can use the code without having to integrate it into Draw. While some minor additions are necessary to Draw (a method to determine if font has char and one to determine the exact bounding box of a character), getting these accepted is going to be easier than a full blown Unicode composition engine with heavy bias toward Latin and CJK. And BTW, I'm still completely against Utf32, so it's safer this way .