Subject: Re: Will UPP support full UNICODE (21bits long codepoint)?
Posted by mirek on Sat, 15 Aug 2020 12:07:08 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Sat, 15 August 2020 13:15Quote: But I am not at the moment sure whether combining characters are the only source of multi-codepoint graphemes.

Yeah, there are at least surrogate pairs (Since U++ use 16-bit wchar), ligatures and IIRC some hangul graphemes. Anything else that I miss?

Correct me when I am wrong:

I do not think surrogate pairs are the thing - those are just a way to encode codepoints.

Also ligatures are probably not a concern too, these can be considered characters of its own.

But hangul graphems is what worries me... Not that I have searched too much, but so far I have failed to find simple algo how to combine hangul characters...

Anyway, to explain my way of thinking: So far, in GUI, we have wchar == grapheme equivalency. That e.g. means that inside e.g. EditString, when user enters a character, it is simply inserted at cursor position in the text.

For full unicode, we will need to change 16bit wchars to graphemes. That means that "length" of text will now be number of graphemes, position of text the position of grapheme etc...

Of course, in later phase, we will need to deal with graphics too, but I think that might be relatively easy to changing all text edit routines. (To deal with graphics, Font::operator[](int chr) should probably change to something like Font::operator[](const char *grapheme)...)

Mirek