L2/02-176

Kenneth Whistler
UTC Agenda item: How to make "oo" with combining breve/macron over pair?

03/07/2002 11:

[My apologies to some on the unicore list who may not have followed this thread, since most of it was posted on the unicode list -- but it is my opinion that this *belongs* on the unicore list, and should not continue on the unicode list.

Also, I would like an agenda item for this topic at the upcoming UTC meeting in April, and an L2 document number for this email, to serve as a seed document for the UTC discussion. --Ken]

Peter Constable said:

> Ah, but then what does that imply for the metrics of the diacritic? ...
>
> Do you *really* think very many type designers want to mess with this? If
> you stick with pairs, you'll likely get something. A fixed and not overly
> large set of n-tuples would get supported, but we're talking about an
> open-ended mechanism here.
>

Which brings us back around to the basic question of what to do about plain text representation of the arguably useful instances of diacritics over or under digraphs.

The options still are:

1. Make the CGJ have a general category of Me instead of Mn (and then explain the implications of that choice), or

2. Encode some additional number of double diacritics to extend the existing set of:

   U+0360 COMBINING DOUBLE TILDE
   U+0361 COMBINING DOUBLE INVERTED BREVE
   U+0362 COMBINING DOUBLE RIGHTWARDS ARROW BELOW

   (with the initial obvious candidates being DOUBLE BREVE, DOUBLE MACRON, and DOUBLE LOW LINE)

I fully agree with Peter about all the hideous implications of trying to generalize the mechanism to apply to n base characters. That is why

I think any Unicode plain text solution should be constrained, by implementation guidelines, to *digraphs* only. Correspondingly, if we decide to extend the list of encoded double diacritics, it should only be for a few more useful *double* diacritics, and not even consider heading into the uncharted waters of *triple* diacritics (i.e., combining diacritics which would extend over two following base forms).

Anything diacritic applied to more than two base forms should be handled by markup (and by specialized rendering that deals with that markup), and should not be expected to be handled in Unicode plain text.

As Peter indicates, unless this gets buy-in from the rendering engine and font providers, this won't get implemented anyway, and would be useless.

On the other hand, the font and rendering issues presented by the two alternatives:

1. X CGJ Y CGJ COMBINING_BREVE
2. X COMBINING_DOUBLE_BREVE Y

are essentially identical -- they are just two attempts to represent the same piece of text, and the issue of what happens to the breve shape if X is an ascender and Y is not, or X is wide and Y is not, and so on, are effectively the same with either representation.

Since I consider these two options technically identical, in terms of the *rendering* issues they pose, I think the choice in options comes down to other aspects of the alternatives:

Option 1

The advantage of Option #1 is that it requires only a property change, and no newly encoded characters. It is a generalized solution, so that in principle, the discovery of new instances of double diacritic application (e.g., a combining bridge below a "th" digraph, to indicate a dentalized articulation) wouldn't require having to come back to committee to encode a new double diacritic.

The *disadvantage* of Option #1 is that it is *too* open-ended. It can, obviously, lead to all kinds of erroneous expectations about what problems it can solve, and will have to be constrained by implementation guidelines to work only with digraphs. People will also have to be trained as to what kinds of diacritics might actually be implemented (e.g. breves and macrons, but not long solidus overlays or ogoneks,

for example) this way, and there might be a considerable amount of chaos among font implementations, so that people would have trouble determining whether a particular font supported the display of a particular diacritic across a digraph in the way they wanted.

Option 2

The advantage of Option #2 is that it appropriately constrains expectations, by limiting the double application of diacritics to precisely that list of diacritics that we go to the bother of encoding distinctly. Nobody, font vendor or end user, is going to worry about what happens if one tries to apply an ogonek to a digraph, for example, because the option won't exist in the first place.

The *disadvantage* of Option #2 is that it requires encoding another clone of an existing diacritic, for application to digraphs, every time somebody digs up another example. So what is clearly another example of a generative principle in the Latin script will end up being a matter of encoding some small, but in principle open-ended list of presentation forms for what are already existing encoded diacritics. And we'll never be shut of the problem, because somebody will always turn up another one for encoding.

Frankly, I think the arguments one way or the other teeter on the knife edge for this one. It isn't at all obvious, in the long run, which direction would be best.

In fact, I haven't even mentioned the third option: do nothing. We could just continue the status quo and say that any other double diacritics need to be handled by markup. But that option has its own inconsistency problems, since we already have 3 double diacritics encoded in the standard, and since we already have an expectation problem to deal with for the CGJ.

--Ken