L2/10-383
2010-10-05

Title: Quick response to Irish NB comments N3931
Source: Van Anderson
Re: Duployan proposal N3895r
Action: For consideration by JTC1/SC2/WG2 and UTC

This will be a quick response to each of the sections of document N3931, submitted by Michael Everson of the Irish NB. All section numbers refer to document N3931.

Section 2 is a reiteration and confirmation of the minor changes to the collation specification that were non-controversial and incorporated into N3895(revised).

Section 3.1 characterizes the collation specification as being for a "Unified Duployan", and while this is correct, it passes over the fact that the collation algorithm will rarely be used to order items containing characters from the entire allocation. On the contrary, most items to be collated will appear in one of the six shorthand/script systems, and the "unified" Duployan collation behavior is meant to organize each of these separate shorthands. By focusing on collation of the entire Duployan scripts taken together, rather than each individually, the complexity of a binary sort vs. a collated list is vastly overstated.

In regards to **Section 3.2,** the entirety is predicated on the fundamental identity of a character being based on its general shape category. As the Duployan scripts are used, it is actually the opposite. The categories of use to the end user are primarily shorthand identity, which roughly corresponds to language of use, and naturally groups characters by how they vary from the basic Duployan set. If Mr. Everson had instead highlighted the characters necessary to write in a particular shorthand, the organization of the code chart would have still been evident, no matter the size of the columns and rows, while the collation order would be speckled with characters, no matter which shorthand were chosen.

As to **section 4.1**, my only comment is that if WG2 considers D.2.1 no longer

in force, the current allocation remains superior in regards to end-users and novice implementers.

In regards **section 4.2**, the Chinook script was used to write in at least a half-dozen native languages, each with their own unique phonological inventory, and there will certainly be users more comfortable with Duployan layouts based on QWERTY, AZERY, QWERTZ, Cyrillic layouts, etc. By my count, that makes 20+ basic keyboard layouts times four implementations (MSKLC, Apple, Keyman, Linux), not counting any users who want truly individualized input methods. In addition, input by character pickers will also be greatly simplified if the characters necessary to write in a given shorthand are found together, given that users make texts in only one shorthand at a time.

As for **section 4.3**, a simple point of comparison lies in the Latin blocks, where source language is, albeit for historical reasons, a defining characteristic of allocation. Basic Latin contains English characters, Latin 1 contains Western European additions, Latin Extended Additional has a Vietnamese sub-block, African language specific characters are found in Latin Ext-B, etc. etc. Likewise, the Duployan organizes with each of the shorthand systems – roughly, but not exactly corresponding to language – having its own sub-block.

My first comment on **section 4.4** is that allocating Duployan by character shape is like including all of the 'A'+diacritic letters in the Latin block before you get to 'B' – sure, it makes a binary sort come out nice, but it seriously complicates trying to implement something as simple as a Canadian multilingual keyboard.

The second issue with **section 4.4** is that it assumes that users will have inconsistent collation behavior, *and that this inconsistency will render the allocation less usable*. While the first part is true, I think it represents an extremely minor concern. Specifically, if we assume that the collation specification is a somewhat more intuitive ordering than the allocation order, confusion on the end-user's part requires acclimatizing to a system that implements the full collation specification, then moving to a system that

gives only a binary sort. Since the direction of software development is towards implementation, it essentially requires that someone move backward in technology while going forward in time in order to experience the negative consequences of the binary sort.

Please note that while I do agree that the collation specification may be more intuitive, especially for those few of us familiar with all of the Duployan shorthands, it is not a current expectation of any user community. The fact is, inasmuch as end-users will even be aware of collation, they will acclimatize to any sorting behavior they are presented with, as long as it represents a predictable order. The current allocation order of the characters used in a given shorthand does just that.

Lastly, **section 5** presents some hypothetical characters that could be added in the future. While Mr. Everson is correct that those *could* be the characters needed, we, in fact, do not know what form they will take. By placing the two gaps in U+1BC6x, it accommodates ***any*** T, F, or K-based additions in the first gap, and any L-based additions in the second, with some additional room for any W-vowels to overflow from 1BC5x – several W-vowels being known, without any current evidence of their actual use. Accommodating gaps with the same functionality will require 13 spaces in the collation-based allocations, rather than the six in the current allocation, leaving only 4 spaces in the 10 columns for extra vowels and affixes, instead of 10 – I have no idea what kind of character should end up in U+1BC17 as of now.

In short, I vehemently disagree with document N3931, sections 3-5, and believe that the conclusions therein reflect a perspective that is contrary to the needs of actual end-users, the needs of the amateur community that will be tasked with implementing input devices for the hobbyists that need them, and the goal of developing a script encoding that encourages conformant implementation.

-Van Anderson
2010-10-05