Rendering in Dzongkha

Pema Geyleg
Department of Information Technology
pema.geyleg@gmail.com

Abstract

The basic layout engine for Dzongkha script was created with the help of Mr. Karunakar. Here the layout engine could not support reordering and was not efficient. However the layout engine was further customized to be efficient and specific for Dzongkha script only.

1. Introduction

One of the dictionary meanings of rendering is converting the coded content to a required format for display or printing. For instance, an HTML page is officially rendered by displaying it. Of all the different meanings for rendering,this was the one which shared similar meaning to the one that I needed to speak of. However we will be concerning with the rendering of different scripts by software. That is how different scripts are displayed by particular software.

There are scripts like Hebrew which is written from right to left, Thai where two characters changes places while being rendered called reordering and doesn't get displayed in the sequence typed by the user. Then there is Dzongkha script which is written from left to right with lots of stacking above and below the base character. This leads to the question of how do the software display different scripts.

All these work are being done by the rendering engine or layout engine incorporated by that particular software. Here it identifies the script that the user wants, and displays the text using that script correctly.

2. Methods

2.1. Why! We need it?

The answer in the nutshell would be to have ones script displayed/ rendered correctly.

The software developers use the most common and the least complex script called the Latin scripts especially to write English. The Latin script displays the glyphs for character from left to right in order to store in memory.

The "complex scripts" requires complicated rendering behaviour and the text written in these scripts are called "complex text". Examples of complex scripts are:

- Indic scripts(Devanagiri, Tamil, Telegu, and Gujurati).
- Thai
- Dzongkha
- Arabic.

Usually most software comes with rendering support for Latin script. However most software does not come with support for Dzongkha script, for instance the rendering engine for the Latin script is not enough to support the proper display of complex scripts.

In that case we need to incorporate our script support in the software either by developing one for open source software or ask them to include it if the software does not fall under open source.

2.2. Types of rendering engine

Few names of rendering engines found in different software are as follows.

- Uniscribe is used by the Microsoft software.
- Pango: Pan in Greek means "all" and go in Japanese means "language". It is also an Open-source framework for layout and rendering of internationlized text.The Gnome

applications use it for rendering.

- ICU Layout engine: ICU stands for International component for Unicode Maintained by IBM and this rendering engine is being used in Open office application.

2.3. How does it work?

The prerequisite for the rendering engine in any software are as follows:

- The particular script should be supported by the software. Here most languages use scripts encoded in the Unicode & ISO 10646 Standards.
- A working font for that script should exist. Open type fonts are preferred here due to its compatibility with most of the software.
- A keyboard driver for that script should be developed.

The font for a particular script contains rules falling under two main categories called "GPOS" (glyph positioning) and "GSUB" (glyph substitution). Then there are features like "ccmp" (composition and decomposition), "blws" (below base substitution) etc. falling under GSUB rule. Other features like: "blwm"(below base mark positioning)

- "abvm"(above base mark positioning)
- "kern" etc.

fall under GPOS rule.

The fonts may contain language tags for the languages they support. All combinations of characters used by particular languages are accessed by rules or lookups defined in the fonts.

The rendering engine has to identify the script, select the fonts, apply correct rules from the fonts and display it.

The stages of rendering or layout engine are as follows:

- User input is stored in a buffer/memory.
- Identify a script by looking at the Unicode values in the buffer.
- Determine the bidirectional levels for the text.
- Update the language tag using information.
- Decide upon a language engine from the updated language tag and script.
- Decide upon a set of possible fonts from updated language tag and font properties for the character.The sorting of these fonts are achieved on the basis of how well they match the language tag and font properties.
- Apply the rules defined in the font to the Unicode values stored in the buffer.
- Do character, word, line boundary analysis.

The output of this process is usually per line which are then fed into the renderer.

3. Results

3.1. Encoding Model for Dzongkha script

3.1.1. Regular and Combining Consonants

The Dzongkha script text consist of frequently occurring vertically combined conjuncts of consonants and vowels. However, whether or not two neighbouring characters stack vertically or be written left to right, one following the next, cannot always be determined by simply applying contextual or grammatical rules. For this reason and also because of the frequency and complexity of these vertical conjuncts in Dzongkha text, a model, which is different from the one adopted for Devanagiri and other Indic scripts, was adopted while encoding Dzongkha script in the UCS. This Dzongkha encoding model is an explicitly stacking model.

The UCS consists of two complete sets of consonants, which are encoded as separate characters:

- first set of headline consonant characters [U+0F40-U+0F6A], used for single consonant or for consonants occurring in the topmost position of any conjunct stack;
- and secondly, a set of combining consonant characters [U+0F40-U+0F6A], used for all additional consonants occurring in a stack.

The characters for Dzongkha vowels, normally written as marks combining with or dependent on

consonants or consonant stacks, are encoded between these two sets of consonants [U+0F71-U+0FB1] in the UCS.

3.1.2. Character Order

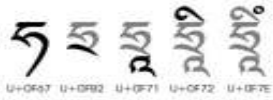
The order for encoding the conjunct stacks are in the order of the parts written, first the character for consonant in the topmost or headline position, and the characters for any combining consonants following that and then the character(s) for any vowel(s):

Figure 1: Conjunct Stacks

That way, it is possible to represent very long stacks which are found in some religious texts:

Figure 2: Long Stacks in Religious Texts

After entering or writing the final below base letter, then entering or writing the vowels or marks occurring above a base glyph from the top of the first consonant upward, is how the stacking is normally done:



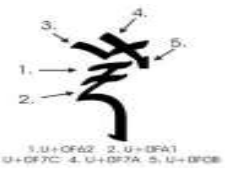


Figure 3: Vowel Stacking

3.1.3. Syllables & Encoding

The "tsheg" bar, usually known as a "syllable" is the basic unit of meaning or morpheme in Dzongkha. Words consist of one or more syllables or tshegs.

Figure 4: Syllable Format

Each syllable contains a root letter (ming zhi) which may have any or all of the following parts, additionally :prefix, head letter, sub-fixed letter, vowel sign, suffix, and post-suffix. The syllables are normally delimited by a tsheg or another punctuation character and there are no inter-word spaces in Dzongkha. The base or root glyph in a dzongkha stack discussed in OpenType rendering for Dzongkha should not be confused with the base or root letter (ming zhi) in a Dzongkha syllable(tsheg bar).

3.1.4. Special Characters

U+0F0c NON BREAKING TSHEG

The properties of normal Tsheg character (U+0F0B) in the Unicode Standard shows that the text-processing applications may wrap a line after any occurrence of this character, meaning that this character provides a line breaking opportunity. Sometimes, a tsheg occur after the letter nga and before a shad, in which case its desirable to oppress this behavior.So in such cases, the non-breaking tsheg (U+0F0C) [inappropriately named "delimiter tsheg"] can be used.

U + 0F6A FIXED FORM RA

In place of RA(0F62), the character FIXED FORM RA(U+0F6A) is used to override the normal contextual shaping of RA:

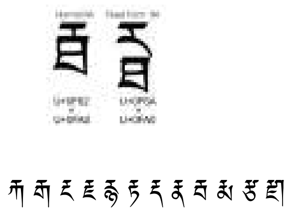


Figure 5: Fixed Form RA

U+0FBA, U+0FBB, U+0FBC: FIXED FORM SUB-JOINED WA, YA & RA

Similarly, there are fixed form variants of the sub-joined consonant like WA, YA and RA which should be used only when it is necessary to override normal contextual shaping behaviour:

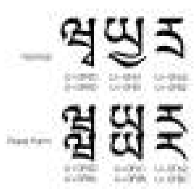


Figure 6: Fixed Form WA, YA & RA

It's is important to note that 0FAD, 0FB1 and 0FB2 should not always be rendered in short form. Hence the WA, YA and RA occurring mid-stack are normally written in their full form.

U+0FC6 DZONGKHA SYMBOL PADMA GDAN

It is an unusual combining symbol character, and can be used to combine with letters or other symbols.

This character are normally entered after the sequence with which it combines:

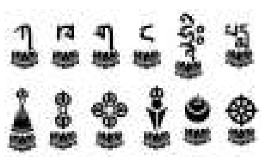


Figure 7: SYMBOL PADMA GDAN

3.2. Features for Dzongkha Fonts

The 4 stages for an Open Type shaping engine for Dzongkha processes text:

1. Analyzing syllables.
2. Reordering characters.
3. Shaping (substituting) glyphs using GSUB features & lookups in the font.
4. Positioning glyphs using GPOS features & lookups in the font.

3.3.Analyzing the syllables

The Dzongkha syllable units, that are received by the shaping engine for the purpose of shaping, are sequential strings of UCS characters. The characters may not be in the sequential order in the order of which they need to be rendered when composing a stack of syllable. First,the shaping engine should identify the first consonant in each stack and then classify all other elements, according to their position that are relative to this character.

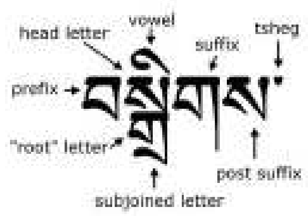


Figure 8: Analyzing the Syllables

3.4. Reordering Characters

The shaping engine creates and manages a buffer of character codes, that are grouped into “clusters” corresponding to Dzongkha the tsheg-bar. And if necessary, the characters are recorded within this clusters according to the script-dependant rules. Then next, a glyph string, corresponding to the character is obtained by mapping characters in sequence to their nominal glyph forms. And all subsequent Open Type lookups are based on these glyphs, not on the underlying characters.

Dzongkha is written from left to right and within Dzongkha words and syllables, there are often vertical stacks of consonants that are written from top to bottom

and then any vowel signs applied to the stack are written from bottom to top.

- head position consonant (U+0F40-U+0F6A, U+0F88, U+0F89)
- tsa 'phru (U+0F39) [if any]
- sub-joined consonants (U+0F90-U+0FBC)[if any]
- sub-joined a-chung U+0F71 [if any]
- sub joined vowel zhabs-skyu U+0F74[if any]
- srog med ('halant') U+0F84 [if any]? occasionally used for Sanskrit transliteration
- above base vowel(s) U+0F71, U+0F7A - U+0F7E [if any]
- above base U+ 0F82 or 0F83
- other above base marks (0F86, 0F87)

The Bhutanese are taught to write the part of stack in this order and also it's the best order for shaping Dzongkha. Also in Dzongkha, it is important to shape the below base conjunct before the above base, as shaping of above base glyphs may be dependent on the final shape of the complete below base conjunct.

3.5. Shaping Glyphs

After the characters have been recorded and mapped to their glyph forms, then the next step taken by the shaping machine is to apply contextual shaping or glyph substitution(GSUB) features to the glyph string.

For the Microsoft Windows, Uniscribe (shaping engine) does this, by calling the Open Type Layout Service Library to shape the Dzongkha syllable.

All the OpenType processing are divided into set of features that are defined in OpenType specification and each feature is applied ,one by one ,to appropriate glyphs in the syllable.

For instance, in Microsoft Windows,Uniscribe makes as many calls to OTL Service Library as there are lookups for each feature, which ensures that the lookups will be executed in the desired order.

3.5.1. Shaping Features:

- Glyph Composition Decomposition: To preprocess any glyph that require composition or decomposition, apply lookups under 'ccmp' feature.

- Conjuncts: To create conjuncts or ligatures, apply lookups under 'blws' feature.

- Below-base Marks: To get any additional below-base combining consonants and any below-base vowel marks; and other below-base marks, apply additional lookups under 'blws' feature.

- Above-base Marks: To get any above-base vowel conjuncts; above-base vowel modifiers; and above-base marks, apply lookups under ' abvs' feature.

3.6. Positioning Glyphs

After the shaping or substitution features have been done ,the shaping engine will apply GPOS(OpenType Positioning) features to position glyphs.

3.6.1. Positioning Features:

- Applying the lookups under 'blwm' feature to position dependent below base marks.

- Applying lookups under 'abvm' feature to position the dependent above-base marks.

- Applying lookups under 'kern' feature to adjust spacing of base glyphs or conjuncts with respect to each other.

4. Discussion

Dzongkha script is now supported from ICU 3.6 and Pango 1.8. Microsoft's Uniscribe also support Dzongkha script.

Dzongkha computing is now possible in Open Office and Gnome Applications in Linux operating system.

5. Conclusion

There are different rendering engines being used by different open source software applications. Open Office uses ICU layout, Gnome application uses Pango and Mozilla uses its own rendering engine. We can customize these rendering engines to include support

for different scripts. It would have been nice if we could have a single rendering engine for all the applications.

However Microsoft uses Uniscribe as a renderer or Open Type layout engine for all its applications. We don't have the option of customizing it for our needs.

We have to depend on the people in Microsoft to decide whether they would like to incorporate support for our script.

6. References

[1] "Pango Layout Engine" www.pango.org

[2] "Layout Engine" www.icu.sourceforge.net/ userguide/layoutEngine.html

[3] "Uniscribe" www.microsoft.com/ typography/ developers/uniscribe/default.htm