Thai Printing Support in FOSS

Theppitak Karoonboonyanan

23 January 2006

1 Requirements Summary

1.1 Rendering

As summarized in [1] and [2], rendering or typesetting Thai texts involves the following issues:

1.1.1 Word Breaking

There is no word delimitor for Thai. Texts are written continuously. Word boundaries need to be determined when wrapping lines, for example.

1.1.2 Cell Clustering

Clustering means tokenizing text string into clusters, each comprising a base character and zero or more combining characters. In case of excessive combining characters, or invalid sequence, extra clusters without base character will be introduced to show the excessive characters. These extra clusters might be rendered with dotted circle as its base, for example.

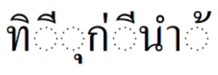


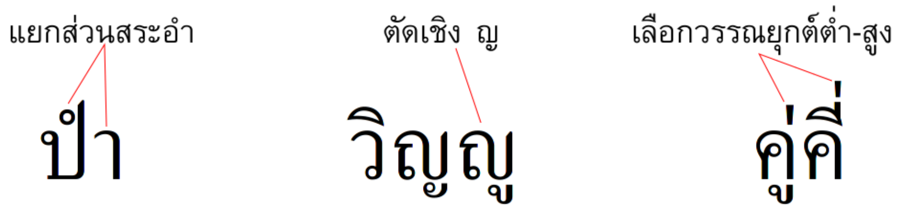
Figure 1: Thai cell clustering

1.1.3 Substitution

- U+0E33 (SARA AM) needs to be decomposed for proper rendering. The character is composed of two parts, U+0E4D (NIKHAHIT) and U+0E32 (SARA AA). NIKHAHIT is a combining character, while SARA AA is a base-line character. The glyph for SARA AM in most fonts are already prepared for placing both parts on normal base characters. But some base characters with upshooting stem will cause an overlap with NIKHAHIT. So, decomposing SARA AM will provide some chance for proper positioning.

  Then, the decomposed NIKHAHIT needs to be further reordered with any existing tone mark, so that it is stacked to the base character before the tone mark.

- The removal of descender component of some characters, namely U+0E0D (YO YING) and U+0E10 (THO THAN), when combined with a below-base combining character.
- The selection of alternate glyphs for tone marks between that combined directly to the base character and that combined to upper vowel, for elegant typography. The glyph for the former case looks better if it is bigger than the one for the latter case.



(a) SARA AM decomposition (b) Descender removal (c) Alternate glyph selection

Figure 2: Substitution

1.1.4 Positioning

Thai text can be rendered in 4-5 levels of stacking. One for the base-line character, two for upper vowels and tone marks, and one for lower vowels. To be more precise, most implementations have added the fifth level for placing lower vowels below some base characters with descender.

In general, Thai orthography always aligns all vertically stacking characters to the right edge of the base character. This coincidence has simplified Thai rendering process by making zero-width and negative-offset are enough to place Thai combining characters properly without the need of advance font technology. (This is not the case for Lao, where the character stacks are align at the center of the width of the base character.)

However, there are some exceptions

- Base characters with upshooting stems, namely U+0E1B (PO PLA), U+0E1D (FO FA), U+0E1F (FO FAN) and optionally U+0E2C (LO CHULA), requires the upper combining characters to be shifted left.
- Base character with downshooting stems, namely U+0E0E (DO CHADA) and U+0E0F (TO PATAK), requires lower combining characters to be shifted down.
- Tone marks and cancellation mark should be attached to the character it combines to. This can mean putting the mark in the same level as, or in a higher level than ordinary upper vowels, depending on the existence of an upper vowel in the cluster.

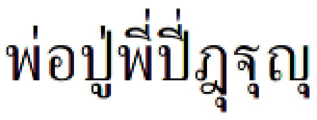


Figure 3: Combining character positioning

1.1.5 Character Spacing

This is a unique requirement for justifying paragraphs when the text is dominated by scripts with no word delimitor like Thai and Lao. Apparently, there are usually insufficient space characters to expand to fill the line. Justified paragraphs often look bad without character spacing support.

To be more precise, it should be called *cluster spacing,* i.e. the distribution of space between clusters to justify paragraphs. There should be no space between characters in the stack.

1.2 Fonts

Many efforts have been made to create proper typesetting systems for Thai texts, from the dot-matrix printers with monospace fonts to Type 1, TrueType, and OpenType. Rendering issues have been addressed in slightly different approaches based on the technology in use.

1.2.1 Complex Text Supports

For Type 1 and TrueType fonts, all rendering issues are handled by the rendering engines, with specific requirements upon the glyph set to be provided by the fonts. In particular, the glyph positioning is done by substitution with pre-adjusted glyph variants, encoded in Unicode Private Use Area (PUA).

OpenType introduces new capabilities to describe glyph substitution and positioning in the font, so that not so much language-specific capabilities have to be loaded in the rendering engines. Some OpenType fonts have been locally created and have been supported by Windows Uniscribe and GNOME Pango. With OpenType fonts, the tasks left at the rendering engine are word breaking, clustering, and processing GSUB and GPOS tables in the fonts.

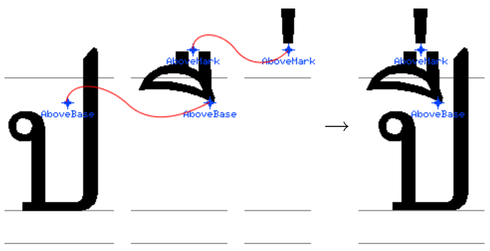


Figure 4: Combining character positioning with GPOS

1.2.2 Metrics

Like many Asian scripts, relative size of Thai font to match a given English font is a controversial issue. Since Thai texts are rendered in multiple levels, the glyph design boxes are divided into more parts, leaving less space than ordinary English fonts for base-line characters. This makes Thai fonts relatively smaller than English fonts of the same size. As a rule of thumb, English glyphs in English font of size s will match those in Thai fonts of size $1.33s$. On the other hand, the default "Sans 10" font used by many applications will effectively get "Sans 10/1.33" or about 7.5 points for Thai fonts. This is true for typical Thai fonts in Windows.

Another approach is to try to match English fonts at the same size. This means the increse in base-line character size. Fonts in this category handle this change in two different ways:

1. Try to retain the same glyph design box by reducing the size of upper/lower marks, sacrificing the proper shapes for readablility without the risk of being clipped. Fonts in this category include Tahoma in Windows and Loma from NECTEC.
2. Simply scale up everything, without care on getting out of the design box. This can retain the proportionality of glyphs, and is rendered properly in some rendering engines like Pango and OpenOffice.org. However, the characters are clipped or the fonts cause incorrect dialog size calculation in many cases (such as Mozilla).

1.2.3 Quality

The lack of hinting in TrueType fonts is the most obvious problem of many free fonts. This is important for low solution situations like embeded devices, but of course not a big problem for printing.

2 Current Situation

2.1 Fonts

There are some free fonts available in the market, for examples:

- National Fonts, created by a committe formed by NECTEC, now maintained by TLWG:
  - Serif: Kinnari, Norasi
  - San Serif: Garuda
- Created by NECTEC
  - UI font: Loma
- Created by TLWG
  - Monospace: TlwgMono

  - Semi-monospace (with zero-width combining characters): TlwgTypewriter
  - Handwriting: Purisa

All above are available in FontForge project files, with OpenType features.

2.2 LaTeX

- Word breaking with `swath` and `cttex` filters
- Type 1 fonts
- Substitutions with TeX virtual fonts
- Positioning by substitution with PUA glyphs

2.3 GNOME

- Word breaking with `pango-libthai` third-party plug-in
- TrueType and OpenType support
- TrueType: substitutions by Pango Thai module; positioning by substitution with PUA glyphs
- OpenType: substitution with '`ccmp`' table; positioning with '`mark`' and '`mkmk`' tables

2.4 KDE

- Word breaking with `libthai` dlopen()
- TrueType support
- TrueType: no substitution; calculated positioning without assistance from PUA glyphs (poor quality)

2.5 OpenOffice.org

- Word breaking with internal engine
- TrueType support
- TrueType: substitutions by internal engine; positioning by substitution with PUA glyphs
- Substitution and positioning do *not* work if OpenType features exist in the font! Line spacing will also be huge.

References

[1] Theppitak Karoonboonyanan. *Status of Thai Support in GNU/Linux/X.* `http://linux.thai.net/~thep/thaisupp/#OM_Render`.

[2] Theppitak Karoonboonyanan. *Spec for Thai OpenType Font Creation.* `http://linux.thai.net/~thep/th-otf/`.