Making Type 1 fonts for Vietnamese

Hàn Thế Thành

University of Education
HoChiMinh City
Vietnam
email:: hanthethanh@gmx.net

Abstract

In this article I describe how I made VNR fonts and converted them to Type 1 format. VNR is the Vietnamese version of CMR fonts, which is written in METAFONT based on CMR and EC font sources. Conversion to Type 1 format was done based on the Type 1 version of CMR produced by BlueSky, with help of MetaFog, FMP, FontLab and a lot of hacking in VNR sources and Bash and Perl scripting. The result is a set of Type 1 fonts that is similar to the BlueSky fonts, but also contain Vietnamese letters with the same quality of outlines and hints.

Vietnamese letters and VNR fonts

Vietnamese is written with Latin letters and a few more accents in a system called *Quốc Ngữ* (which can be translated to English as "national language") developed by the Portuguese missionary Alexander Rhodes. What separates Vietnamese from other languages typeset with Latin characters is that in Vietnamese some letters can have two accents. The total number of accented letters in Vietnamese (including uppercase and lowercase letters) is 134. Table 1 lists all lowercase Vietnamese letters.

a	á	ạ	à	ả	ã
ă	ắ	ặ	ằ	ẳ	ẵ
â	ấ	ậ	ầ	ẩ	ẫ
e	é	ẹ	è	ẻ	ẽ
ê	ế	ệ	ề	ể	ễ
i	í	ị	ì	ỉ	ĩ

o	ó	ọ	ò	ỏ	õ
ô	ố	ộ	ồ	ổ	ỗ
ơ	ớ	ợ	ờ	ở	ỡ
u	ú	ụ	ù	ủ	ũ
ư	ứ	ự	ừ	ử	ữ
y	ý	ỵ	ỳ	ỷ	ỹ
đ					

Table 1: List of all Vietnamese lowercase letters

Vietnamese accents can be divided into three kinds of diacritic marks: tone, vowel and consonant. Table 2 shows them all with examples.

As Vietnamese letters are just like Latin letters, it is natural to write VNR as a set of METAFONT files that compose Vietnamese letters from English letters in CMR sources and appropriate accents. There were several works on this topic prior to VNR. The best among them was the package vncmr by Werner Lemberg, who also created some basic macro support for typesetting Vietnamese in LaTeX and plain TeX.

As I learnt more about TeX and METAFONT, I set out to create VNR fonts as I was not entirely happy with the accent shapes and positioning in previous packages. I borrowed quite a lot ideas from the vncmr package and other CMR-based fonts, like the Czech and Polish version of CMR fonts. It took about 2 years until the first version was released and used in practice.

Diacritic mark	Example
Vowel	
breve	băn khoăn
circumflex	hôm nay
horn	Qui Nhơn
Tone	
acute	Lái Thiêu
grave	Bình Dương
hook above	Thủ Đức
tilde	dĩ vãng
dot below	học tập
Consonant	
stroke	đã đời

Table 2: List of all Vietnamese diacritic marks

In the beginning I wrote VNR fonts based on CMR sources. Later Werner Lemberg and Vladimir Volovich convinced me that it would be better to switch to EC sources instead of CMR sources. The two main reasons are:

1. It will be easier to include Vietnamese letters in CM-Super fonts;
2. it will fix some problems in VNR fonts (mainly with encoding and some missing glyphs)

So I changed VNR fonts to be based on EC sources, which of course introduced a few new problems. The most noticeable is with naming: Should VNR fonts be named in EC-like naming convention (like `vnrm1000.mf`) or in CMR-like naming convention (like `vnr10.mf`)? In the end I decided to support both, but the sources (and thus the glyph shapes) are EC-based. Why not drop one of them? Because:

1. I want to support EC-like naming, so Vladimir can include Vietnamese letters in his CM-Super fonts;
2. I want to support CMR-like naming, because I want to use the Type 1 version of CMR fonts. Also, there are many packages which depend on CMR-style names (like `texinfo`).

I would like to point out that I am not a type designer and therefore the aesthetic aspect of Vietnamese letters (regarding accent shapes and positioning) is open to discussion. What I have done is heavily based on what I learnt from other Vietnamese fonts available to me, from the typography of other languages (mainly Czech and Polish), and from comments from various people. It's not to say that I take no responsibility for VNR fonts. If something looks bad, it's my fault of course.

It's good that I can use Vietnamese with TEX, but I want PDF

And PK fonts just look ugly in Acrobat Reader. So I need Type 1 fonts. My first attempt was to create a set of virtual fonts for Vietnamese, which refer to CMR fonts. The shortcoming is that it doesn't look good, as many needed accents are not available in CMR and must be substituted by some other glyphs.

I also tried to use TEXtrace to generate Type 1 versions of VNR fonts (using both EC and CMR naming). The result is useable, but the size is too large and the quality of auto-generated outlines and hints cannot be compared to BlueSky fonts. It seems a pity to me that I could not use the high-quality outlines and hints of English letters in BlueSky fonts. So I was looking for another method to convert VNR fonts to Type 1 format.

Reuse is a good idea

The main idea of efficient generating Type 1 format for VNR fonts is simple:

1. Take the accents from METAFONT sources and convert them to Type 1 format;
2. compose those accents and the English letters from BlueSky fonts to get accented letters.

Thus nothing is new; each step is just a reuse of existing data.

Use the right tool for each task

The two steps mentioned above can be done properly, given that we have the right tool for each of them.

Converting the accents to Type 1 format This can be done *very quickly* using TEXtrace, or *nearly perfectly* using METAPOST and MetaFog. As I got an evaluation copy of MetaFog and I want the result to be as good as possible, I chose the latter. In this step, some glyphs that are needed in T5 (Vietnamese TEX encoding) but are missing in BlueSky fonts are also converted. The number of glyphs needed for each font is 47, from which 22 are Vietnamese accents and letters. Table 3 shows all the glyphs that need to be converted to Type 1 format. The `vl` and `vu` prefixes stand for "Vietnamese lowercase" resp. "Vietnamese uppercase". At the moment they look identical, however they may be changed.

Vietnamese accents and letters	**Additional glyphs**
dotbelow	quotesinglbase
hookabove	guilsinglleft
vlgrave	guilsinglright
vlacute	quotedblleft
vlcircumflex	quotedblright
vltilde	quotedblbase
vldotbelow	guillemotleft
vlbreve	guillemotright
vlhookabove	endash
vugrave	emdash
vuacute	cwm
vucircumflex	zeroinferior
vutilde	uni2423
vudotbelow	quotedbl
vubreve	dollar
vuhookabove	less
Ohorn	greater
Uhorn	backslash
ohorn	asciicircum
uhorn	underscore
Dcroat	braceleft
dcroat	bar
	braceright
	asciitilde
	sfthyphen

Table 3: List of glyphs that need to be converted from METAFONT to Type 1 using MetaFog

Many of the additional glyphs are available in BlueSky fonts already, however their availability is not consistent. Each of those additional glyphs listed here is missing at least in some BlueSky font. To get rid of this headache, I chose to convert them all.

47 glyphs is quite a large number for each font, as the total number of glyphs to be converted is 2585 (47×55). Fortunately, not all of them required manual correction, and the additional glyphs need to be

converted only once, given that EC sources will not be changed. So, if I change the VNR sources, I will have to re-convert only those 22 Vietnamese accents and letters.

When MetaFog finished and we got the outlines of needed glyphs, hints for accents were auto-generated using FontLab.

Several papers have been published on this topic so I will not repeat it all here.

In short, using METAPOST and MetaFog gives the best result, however a lot of manual work is required. That's one of the reasons why I didn't convert all Vietnamese glyphs but only those that are *really* necessary. Why re-generate the English letters when they all already exist in BlueSky fonts at the high quality?

Composing the accented letters Composing the base letters and accents to create accented letters has two advantages:

1. The base part of the letter will have the outlines and hints at the same quality as the BlueSky fonts;
2. the final size of fonts will be considerably reduced, as base letters and accents will be put into subroutines in the final Type 1 fonts. This way each glyph is included only once (again, reuse is good).

The tool used in this step is FMP (Font Manipulation Package) from Y&Y. In this package, there is a tool that can create composite glyphs from existing glyphs in a font. The question is how to place an accent over a letter *exactly* like the VNR sources do. The solution is simple: I added some hooks into VNR sources, so all information about accent positioning is written to a log file. Then I wrote some Perl scripts to extract the data from the log file and use them with the composite tool from FMP.

What about the result?

A rough comparison on average font size gave the following result:

BlueSky	25 KB
Type 1 VNR (TEXtrace)	70 KB
Type 1 VNR (as described herein)	40 KB

Compactness is gained thanks to FMP, which puts all accents and English letters into subroutines so they can be reused without duplication as mentioned above.

Regarding the quality of outlines and hints of each accented letter:

- The base part has the the same outlines and hints as in BlueSky fonts;
- the accent part has the outlines produced by METAPOST (and MetaFog), hints are auto-generated by FontLab.

As the accent shapes are quite simple, hints auto-generated by FontLab are quite reasonable. It's really hard to do better without a great deal of experience in manual hinting.

A sample of font vnr10 is shown in figure 1.

Anything missing?

Yes: The VNR sources are EC-based, while the English letters from BlueSky are Computer Modern-based. So it is possible that the Type 1 version of VNR fonts will have slightly different glyph shapes from their counterpart generated by METAFONT. I did some quick comparisons: The difference is only visible at very high resolution, and can be tolerated in my opinion. Hopefully nobody will notice it.

Related works

I applied a similar method to add Vietnamese letters into the URW fonts. The result is a package I called URWVN. Regarding technical aspect, URWVN fonts are very similar to VNR fonts:

1. They are very compact; the average size of the URWVN fonts is 35 KB;
2. the base part of accented letters has the same outlines and hints as in URW fonts;
3. the accent part of accented letters was drawn manually,[1] hints auto-generated by FontLab.

The aesthetic aspect is open to discussion, as I am not a type designer. Comments or suggestions are very welcome.

A sample of font Times Roman is shown in figure 2.

[1] or more precisely they are derived from existing glyphs in URW fonts with manual modification

Chí Phèo

Hắn vừa đi vừa chửi. Bao giờ cũng thế, cứ rượu xong là hắn chửi. Bắt đầu chửi trời. Có hề gì? Trời có của riêng nhà nào? Rồi hắn chửi đời. Thế cũng chẳng sao: đời là tất cả nhưng chẳng là ai. Tức mình hắn chửi ngay tất cả làng Vũ Đại. Nhưng cả làng Vũ Đại ai cũng nhủ, *"Chắc nó trừ mình ra!"* Không ai lên tiếng cả. Tức thật! Ồ! Thế này thì tức thật! Tức chết đi được mất! Đã thế, hắn phải chửi cha đứa nào không chửi nhau với hắn. Nhưng cũng không ai ra điều. Mẹ kiếp! Thế thì có phí rượu không? Thế thì có khổ hắn không? Không biết đứa chết mẹ nào đẻ ra thân hắn cho hắn khổ đến nông nỗi này? A ha! Phải đấy, hắn cứ thế mà chửi, hắn chửi đứa chết mẹ nào đẻ ra thân hắn, đẻ ra cái thằng Chí Phèo! Hắn nghiến răng vào mà chửi cái đứa đã đẻ ra Chí Phèo. Nhưng mà biết đứa nào đã đẻ ra Chí Phèo? Có trời mà biết! Hắn không biết, cả làng Vũ Đại cũng không ai biết...

Figure 1: A text sample of vnr10

Chí Phèo

Hắn vừa đi vừa chửi. Bao giờ cũng thế, cứ rượu xong là hắn chửi. Bắt đầu chửi trời. Có hề gì? Trời có của riêng nhà nào? Rồi hắn chửi đời. Thế cũng chẳng sao: đời là tất cả nhưng chẳng là ai. Tức mình hắn chửi ngay tất cả làng Vũ Đại. Nhưng cả làng Vũ Đại ai cũng nhủ, *"Chắc nó trừ mình ra!"* Không ai lên tiếng cả. Tức thật! Ồ! Thế này thì tức thật! Tức chết đi được mất! Đã thế, hắn phải chửi cha đứa nào không chửi nhau với hắn. Nhưng cũng không ai ra điều. Mẹ kiếp! Thế thì có phí rượu không? Thế thì có khổ hắn không? Không biết đứa chết mẹ nào đẻ ra thân hắn cho hắn khổ đến nông nỗi này? A ha! Phải đấy, hắn cứ thế mà chửi, hắn chửi đứa chết mẹ nào đẻ ra thân hắn, đẻ ra cái thằng Chí Phèo! Hắn nghiến răng vào mà chửi cái đứa đã đẻ ra Chí Phèo. Nhưng mà biết đứa nào đã đẻ ra Chí Phèo? Có trời mà biết! Hắn không biết, cả làng Vũ Đại cũng không ai biết...

Figure 2: A text sample of Vn Nimbus Roman No9 L Regular

Future works

At the moment, VNR and URWVN fonts have only one version for each accent. Therefore uppercase and lowercase forms of a letter have the same accent, e.g. `aacute` and `Aacute` have the same shape for the accent. This should be improved by introducing two separate versions for each accent, one for uppercase and one for lowercase. This is important for Vietnamese, as many letters have double accents, which causes such letters to be very high. Uppercase letters should have wider and lower accents than their lowercase counterparts.

The METAFONT sources of VNR fonts could also be improved to make the accents look better, especially for sans serif fonts.

References

- The VNR fonts and vntex are available at: http://vinux.sourceforge.net/vntex;
- The URWVN fonts are available at: http://vinux.sourceforge.net/urwvn;
- Samples of VNR fonts are available at: http://vinux.sourceforge.net/vntex/vnfontsample.pdf.gz;
- Samples of URWVN fonts are available at: http://vinux.sourceforge.net/urwvn/urwvnsample.pdf.bz2.

Acknowledgments

I would like to thank following people; without them my work would never be finished:

- Werner Lemberg for creating the vncmr package from which I learnt a lot;
- Tom Kacvinsky for donating a copy of FMP;
- Richard Kinch for creating MetaFog and donating an evaluation copy;
- Paul Watry for donating a copy of FontLab;
- and William Adams for correcting this paper.

A Step-by-step description of generating Type 1 VNR fonts

As an example, `vnr10.pfb` was generated as follows:

1. Run METAPOST on `vnr10.mf` to write information about accent positioning into a log file; the result of this step is `vnr10.log` that contains the precise positioning of accents.
2. Run some Perl scripts to extract the needed information from the log file and convert it to a format suitable for use with FMP. The result are two files `vnr10.ac1` and `vnr10.ac2`.

   The file `vnr10.ac1` contains accent positioning instructions for letters with single accent. Those instructions are similar to the `CC` commands often found in AFM files; they look like:

   ```
   CC aacute 2 ; PCC a 0 0 ; PCC vlacute 146 495
   CC abreve 2 ; PCC a 0 0 ; PCC vlbreve 85 495
   ```

   `vnr10.ac2` contains accent positioning instructions for letters with double accents. As the composite tool from FMP does not allow compositing a letter with two accents, this must be accomplished in two passes. `vnr10.ac1` is used in the first pass and `vnr10.ac2` in the second pass. The instructions in `vnr10.ac2` look like:

   ```
   CC abreveacute 2 ; PCC abreve 0 0 ; PCC vlacute 205 631
   ```

   To make it clear, a character with double accents is constructed as follows: In the first pass, the base letter and the first accent are composed to create a new glyph. In the second pass, this new glyph is composed again with the second accent to get a double-accented letter.
3. Run METAPOST on `vnr10.mf` to generate PS outlines of glyphs that need to be converted to Type 1 format; those are listed in table 3.

4. Run MetaFog on the result of the previous step to convert them to Type 1 format (without hinting). If some glyphs were not correctly converted, manual intervention is needed in this step.
5. Autohint the font generated by MetaFog in the previous step, using FontLab. The result is a font named `vnr10-t5supp.pfa`.
6. Process `cmr10.pfb` from BlueSky fonts by a Perl script so it doesn't contain the `div` operator in glyph descriptions. This is needed because some tools from the FMP package don't like fonts containing this operator. The result is a font called `cmr10.pfa` (FMP tools work with PFA format only).
7. Run a Bash script that uses FMP tools to compose `vnr10.pfb` from `cmr10.pfa`, `vnr10-t5supp.pfa`, `vnr10.ac1` and `vnr10.ac2`. This script does the following:
   - Remove unwanted glyphs from `cmr10.pfa`;
   - merge `cmr10.pfa` and `vnr10-t5supp.pfa` to get all needed glyphs for composition in a single font (FMP requires that);
   - compose all letters with single accent, according to instructions in `vnr10.ac1`;
   - compose all letters with double accents, according to instructions in `vnr10.ac2`.
8. Post-process the result `vnr10.pfa` to fix a few final things, like UniqueID, encoding, font name and the like.

B Step-by-step description of generating URWVN fonts

The process is similar to the case of VNR fonts. However, there are two main differences:

1. The accents must be drawn manually instead of being generated from METAFONT sources;
2. accent positioning is also done "manually" instead of being generated from the METAFONT sources.

As an example, `utmr8v.pfb` (—8v— is the abbreviation suggested by `fontname` for Vietnamese encoding) was generated as following:

1. Open `utmr8a.pfa` in FontLab and add the Vietnamese accents plus some letters that don't exist yet: `uhorn`, `Uhorn`, `ohorn`, `Ohorn`.
2. Use the program `a2ac` (written by Petr Olšák, with some modification to fit my need) to create a "rich" AFM file where composite instructions look like:

   ```
   CC Ocircumflex 2 ; PCC O 0 0 ; PCC vucircumflex 194 181 ;
   CC Ocircumflexacute 3 ; PCC O 0 0 ; PCC vucircumflex 194 181 ; PCC vuacute 116 339 ;
   ```

   The reason to use `a2ac` instead of writing these instructions from scratch is that `a2ac` allows generating `CC` instructions in a clean, systematic and efficient way; it also allows automatic generation of kerning information for newly composed glyphs. The functionality of `a2ac` is very similar to the famous `fontinst` (with some limitations).
3. Run a script that takes the above AFM file, generate a virtual font and run TEX (pdfTEX) on a test file to display how composited glyphs look. The purpose of this step is to check quickly whether accent positioning is already good, or if it needs further improvements. If so, we come back to the previous step. This allows doing the cycle edit-compile-test-edit very efficiently.

   Why don't I do accent positioning in FontLab but use `a2ac` and all this hacking? Because it allows accent positioning in a precise, consistent and more efficient way. If I did it in FontLab, there would be some disadvantages:

   (a) It would take longer;
   (b) accents could not be placed consistently;[2]
   (c) if some accent is changed, all must be done again.
4. When accent positioning is reasonable, run a script that uses FMP tools to compose accented letters, like in case of VNR fonts.
5. Post-process the result to fix various things.

[2] well, they could, but at a very high price