Open development of open font — Revaluation and application of Wada-lab Font Kit

Kazuhiko[1] and KANOU Hiroki[2]

Abstract
As the importance of free desktop environment by open source softwares getting larger, the importance of open fonts also become larger. We first describe the condition of open fonts in Japan and then explore the effective way of making huge amount of Kanji glyphs.

Keywords: open font, font development, font composition, Wadalab Font Kit

1 Introduction

1.1 Open Font Definition

There are many fonts that we can use free-of-charge. But what we need is not free as in 'free beer' but as in 'free speech', i.e. what we call 'Open Font' in this article.

The definition of 'Open Font' should be just like 'Open Source Definition**??**'.

- free redistribution
- source code
- derived works
- no discrimination against persons/groups
- no discrimination against fields of endeavour
- distribution of license
- license must not be specific to a product
- license must not restrict other software

1.2 Why is Open Font needed?

Under the licence mentioned above, the freedom of modification is guaranteed. We need to modify fonts to follow the changes, eg. CCS/CES (Shift-JIS ⇒ Unicode ⇒ ?), format (TTF ⇒ OpenType), design trends (line-oriented ⇒ curve-oriented) etc.

The freedom of usage is also guaranteed by open fonts. Of course we need the freedom of usage for our freedom of speech.

And the freedom of distribution is also important to exchange documents. When we can distribute fonts with documents, we can achieve yet another WISIWYG — What **I** See Is What You Get.

[1] http://www.fdiary.net/ mailto:kazuhiko@fdiary.net
[2] http://khdd.net/kanou/ mailto:kanou@khdd.net

1.3 Open Font Today

We have many open fonts for Latin (eg. URW, Bitstream Vera). And we have some open fonts for Chinese and Korean (eg. Arphic, Wang, Baekmuk). But we have few open fonts for Japanese.

We formerly had quality open fonts whose names were Kochi fonts. Kochi fonts based on a free TrueType font that was auto-converted from a 32dot bitmap font that was knows as 'free' at the time. And the author of Kochi fonts released them as public domain fonts.

But on June 15th 2003, it was revealed that the 32dot bitmap font that is a origin of Kochi fonts is illegally stolen. So we asked to cancel distribution of these fonts and then we negotiated with its copyright holders for several times. As the result of negotiation, they offered a 'free-of-charge' license. But the author of Kochi fonts rejected their offer, and the development was cancelled.

1.4 Next Generation Font Development

Since we have lost quality Japanese open fonts, we need their 'alternatives' soon. Therefore development speed is a first priority. But quality is second of course. For the rapid and efficient development, we tried revaluation and application of Wadalab Font Kit that was the automated glyph composer.

2 Revaluation of Wadalab Font Kit

2.1 Overview of Wadalab Font Project

Wadalab ('Wada-ken' in Japanese) Font is a one of the most successful attempt in the automated kanji (CJK ideographs) glyph generation. It was developed from April 1990 until March 1994 by Kanji Development Branch of Wada Laboratory, Faculty of Engineering, the University of Tokyo. Original distribution of the font have four typefaces of JIS X 0208-1990 (includes 6355 kanjis) and two of JIS X 0212-1990 (includes 5801 supplementary kanjis) and total number of kanji glyphs exceeds 37,000.

At the time they published the result of the research, a set of generated font was released. The Wadalab Font was the only available open Japanese font in the late 1990s and they are widely used on non-commercial operationg systems and commercial Unices together with TeX and Ghostscript, even though the font was a demonstration of the experiment and not tuned nor fixed by hand at all.

2.2 Technical Overview

Developers of Wadalab Font had an anbitious goal that a user, who adds a character by describing the combination of predefined parts, need no aesthetic judgment and all issues on balancing components are processed automatically by the system. The same policy is adopted in the realization of the outlines from the skeletons of glyphs. Only function to calculate outlines is defined for each type of elemtents of strokes and one function handles different sizes, aspect ratios, and slopes consistently. A set of function can also be tunable in the stroke width but different style of typefaces have different sets of rendering functions. Currently three styles—'mincho' (serifed), 'kaku-gothic' (sans serif with square strokes) and 'maru-gothic' (sans with rounded strokes)—are defined.

The process of outline generation is divided in three stages:

- exapnding abstract definition to the concrete coordinates of the skeleton
- calculating center lines from skeleton and extending to the stroke
- processing joins of elements and generating serifs at the end of elements.

Balancing of components with different complexity is handled by two strategies:

- displacing stems uniformly as far as possible
- ensuring the minimum distances of strokes for each pair of type of elements.

This algorithm make possible automatic kerning of components for vertical and horizontal combination of the components. For a nested combination, the initial position of the bounding boxes of surrounded components are defined in the surrounding component by the designer. Balancing engine performs repetitive computation to maximize inner components around the initial positions. The components are scaled by liniar transform after their relative sizes are settled. Detailed processes are described in two papers [2] and [3].

2.3 Outline of the Wadalab Font Kit

Wadalab Font Kit?? is the glyph generator of Wadalab Font and written in UtiLisp, a dialect of Lisp widely used through 1980s in Japan. Renderers have about 7,500 lines (comments and empty lines not included) and graphical skeleton editor have 6,700 lines. Additionally, 12,700 glyph definitions and 2,600 master skeletons of primitive components (including non-kanji) is needed to build the Font.

The Kit was not publicly available because it was thought that very few people have interest in designing fonts and the system was not so user-friendly. But the retraction of Kochi Font raised public interest in digital typography, so Dr. Tetsurou Tanaka, who built the core of Wadalab Font Kit decided to release it just after the incident.

Nonetheless the design quality of machine generated glyphs are not so good, the basic algorithms they developed have a large generality and the system have very high flexibility enough to implement additional ideas. The quality of specific glyphs is essential for the quality of generated skeletons and concrete shape of strokes can easily spoil the balanced skeleton. That was partly because GUI of the glyph editor was too slow to use them interactively and they had no advisor of professional typographer. We believe if they had better enviroment, they accomplished much better glyph designs.

2.4 Past Development

At June 15th 2003, we need to supply a pair of provisional font which substitute the original two typefaces of Kochi Font. Priority was given to (1) clearness of the source of the glyphs, (2) rapid release, (3) compatibility. In this time, the Font

Kit was not referred and commonly used glyphs of Wadalab Font in past was used. The tasks were shared by many peaple. Compilation of glyphs by Mr. Akagaki (Momonga Project), preparation of script to replace the glyphs depending on 32dot bitmaps by Kanou, registration to sourceforge.jp and other coordination by Kazuhiko, and so on. First version of substitute font was released at evening of June 18th. Scripting language of PfaEdit font editor**??** was used to replace glyphs to clarify the operations what we applied to the original Kochi Font and to make the operation repeatable by everyone.

The second stage was porting of the Kit to current environment for revaluation and compilation. X11 interface code in UtiLisp/C was modified to operate the skeleton editor. Then several miscomposed glyphs were redefined and a few bugs in rendering code were fixed. Most of the bugs were in geometrical construction of outlines. For example, in case two auxilary lines are aligned nearly parallel directions, the crossing points go far unexpedtedly and the resulting glyph is totally broken. Procedures for anomalistic cases are added to make acceptable arbitrary combination of glyph sets, stroke styles and width parameters.

2.5 Development in Current Plan

Analisys of source code for porting, bug fix and revaluation gave us some lessens that helps to plan our development in near future. The emphasis are put on:

- stimulating potential developers
- making collabration between programmers and designers easier
- postponing labor-intensive jobs as long as possible.

To encourage developers, we are porting these codes to Common Lisp and allow casual trial on user's favorite Lisp compilers. We should attract not only programmers, but people who can contribute through their aesthetical senses. For such people, programmers should prepare much rooms for cutomization without customizing the glyph. We should abondon some policies to calculate everything that programs can do and introduce example-oriented approach more extensively. On this point, we must have many things to learn from the teaching approach used in KAGE. Improvement of skeleton editor is also anticipated. This is not necessarily porting of original implementation. Alternatively, it is possible to add a feature to edit skeleton to an existing font editor.

We think that migration from a calculation-oriented approach to an example-oriented approach will derive better appearance of for two reasons. Giving multiple master patterns allows us to different proportions in different aspect ratio. For example, while horizontal bar in 十 (ideographic number ten) is set on optical center (slightly above the geometrical center), when if it is used in right part of a glyph (the cases of 汁/什/計/針) the horizontal bar is appearently raised from the center. When the part comes in left (協/博), this component has narrower width and higher horizontal bar. Similarly, algorithms used in outline generation has difficulties to meet all sizes and thickness of the same type. This makes the customization very difficult other than a skilled programmer with a excellent sense of beauty. Designers will prefer to give examples even the number of strokes to define is a tenfold of explicit programming.

In some cases that current algorithm gives bad balance, the glyph must be defined as an inpartible component to get better appearance. It is waste of man-power to adjust all glyphs manually because when the algorithm is improved manual glyphs become unnecessary. We are trying automatic extraction of outlines from Ayu 20dot bitmap font**??** (Figure 1). This is selected because of the simple line style and clear provenance—it is derived from a design by one of the authors.

3 Challenges in Open Font Development

The reason that makes open font development difficult is various. Each of them are not critical, but the combination of them makes work difficult. If you don't require design consistency at all, open development can easily provide you a set of glyph by simply dividing designing task. This type of group work necessary brings to a gross disunity. Differences among glyphs designed by different workers

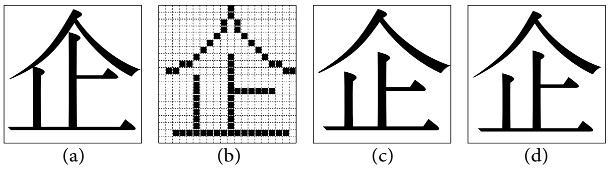

Figure 1: **(a) Result of automatic balancing of Wadalab Font Kit. (b) Ayu 20 dot bitmap font. (c) Rendered outline that endpoints of the skeleton are taken from bitmap. (d) Rendering outline that endpoints are specified by human.** Results from bitmap is not so beautiful because the coarseness of the coordinates but perceptibly better. All glyphs are rendered from a skeleton to outlines by the Wadalab Font Kit.

are appearent even in the extension of a 16 dot bitmap font. Authors have no idea to solve this problem without having intinsive standardization by daily face-to-face meeting as designers in commercial font developing team is adopting.

We expect that font development gets better performance out of Harlan Mills' 'surgical team' approach**??** than homogeneous sharing of glyph making. In the most successible case in recent open font developments**??**, a designer designs all glyphs and two experts works with converting issues (PfaEdit scripting and fixing bugs), and widely invites a opinion from public. These advices, helps to oversights from the author. This cooperation makes glyphs of very high quality undoubtedly, but it takes too much time to design all the glyphs in a character set with thousands of glyphs. We have to develop faster way to design glyphs infrequently used. One way is to supply preliminary glyphs by subsidiary designer, and the another is use glyph generatior. The efficiency will highly depends on the quality of initial glyphs. but in most cases glyph generators will help designers.

Commercial developers will enjoy benefit of technologies invented in open developments. Open font devlopers have lesser merits in most skills needed to design a typeface. Automatic glyph design will be exceptional domain that open development have an advantage, because we have the freely customizable sources. Japanese font vendors are at disadvantaged position in labor costs. We hope automatic glyph design helps commercial font vendors by helping the decrease of designer's simple tasks and competiton in aesthetic sense, not in cost.

Bibliography

[1] http://www.opensource.org/docs/ definition_plain.php

[2] Tanaka, T., Ishii, Y., Takeuchi, M., and Wada, E., Sharing Skeleton Data by Multiple Kanji Fonts through Programmable Rendering. IPSJ Journal **36** 177–186 (1995) [in Japanese].

[3] Tanaka, T., Iwasaki, H., Nagasaki, K., and Wada, E., Making Kanji Skeleton Fonts through Composing Parts. IPSJ Journal **36** 2122–2131 (1995) [in Japanese].

[4] http://gps.tanaka.ecc.u-tokyo.ac.jp/ wadalabfont/cvsweb/cvsweb.cgi/wadalabfont-kit/

[5] http://fontforge.sourceforge.net/ (PfaEdit is renamed to FontForge).

[6] http://x-tt.sourceforge.jp/ayu.html

[7] Brooks, F. P., The Mythical Man-Month, 2nd ed., Chapter 3 (1995).

[8] http://mplus-fonts.sourceforge.jp/mplus-outline-fonts/