Accessibility of Unencoded Glyphs

Ken Lunde, Adobe Systems Incorporated
lunde@adobe.com
http://www.oreilly.com/~lunde/

1. Introduction

The purpose of this paper is to describe a generalized framework for accessing unencoded glyphs that are considered variants of standard (that is, encoded) characters in most contexts. Actual implementation details can be diverse, which involves both engineering and user-interface (UI) issues, but the underlying requirements remain constant. The ultimate goal is to significantly improve the input experience while not eroding the benefits of common text processing operations, such as searching and sorting. Many of the unencoded glyphs that users need to access can be considered Z-axis variations, according to the Unicode/ISO 10646-1:1993 character and glyph model. While the problem being addressed by this paper is described in the context of CJKV scripts, other scripts can also benefit from implementations of this framework.

All of today's input methods (IMs) allow users to access only encoded characters.[1] That is, characters that are accessible through a single code point, whether it is defined by the character set (a standard character) or is available in its vendor- or user-defined regions. While most encoding methods have a limited number of code points slated for vendor- or user-defined characters, Unicode provides significantly more: 6,400 for UCS-2 encoding, and 131,072 more for UTF-16 encoding. For Japanese, the Shift-JIS user-defined region can reliably handle up to 1,880 user-defined characters (0xF040 through 0xF9FC, or 10 rows of 188 code points). Many pre-packaged "gaiji" sets, such as those developed by DTP center Biblos[2] and Enfour Media[3], almost completely fill this region. These gaiji sets include a significant number of characters that are considered variant forms of standard characters.

Contemporary outline font formats, such as those based on PostScript and TrueType technologies, are not limited to any single encoding, and index glyphs through simple integer values called Glyph IDs (GIDs) in TrueType and OpenType, and Character IDs (CIDs) in CIDFonts. Many fonts include glyphs that are unencoded under most circumstances. For Japanese fonts, unencoded glyphs can include JIS78 (JIS C 6226-1978), JIS83 (JIS X 0208-1983), and other kanji variants. These same font formats include tables, 'mort' for GX-based fonts and 'GSUB' for OpenType-based fonts, that define glyph substitutions, which are subsequently exposed to users through an application's UI.

The framework for accessing unencoded glyphs that is described in this paper assumes that the process is performed dynamically by using the glyph substitution tables that are included in a font file. While the end result can be the same as when glyph substitution features are accessed from within an application using its

1. Input methods are also known as input method editors (IMEs) and front-end processors (FEPs).
2. http://www.biblosfont.co.jp/
3. http://www.enfour.com/

UI, the framework essentially moves the process upstream to the point when users input characters. Another approach to this problem was discussed in Martin Dürst's *Exploring the Potentials of Web Technologies for the Handling of Rare Ideographs and Ideograph Variants* (12th International Unicode Conference, Tokyo, Japan, April 1998).[1]

2. A Framework for Accessing Unencoded Glyphs

I firmly believe that glyphs that cannot be considered variant forms of encoded characters should be encoded in the user-defined region, then registered with an IM to enable input by reading or other convenient means. The link that allows access to unencoded glyphs, as described in this paper, is a relationship to an encoded character expressed in a glyph substitution table within a font file.

2.1 Encoded Versus Unencoded Glyphs

Most CJKV character set standards include not only "standard" forms of Chinese characters, but also some traditional or variant forms. However, not all traditional or variant forms are included. This is quite true of Japan's JIS X 0208:1997. The following table lists several Japanese kanji along with their traditional forms and some of their variant forms, categorized whether or not they are in JIS X 0208:1997 (and thus Unicode):

Standard	Traditional	JIS X 0208:1997 Variants	Non-JIS X 0208:1997 and Non-Unicode Variants
学	學	斈	
剣	劍	劒劍劔釼	
辺	邊	邉	邊邊邊邊邊邉邉邉邉邉邉邉邉邉邉邉
弁	辨瓣辯	辦	

There are also instances of other types of characters, such as kana, that have variant forms that are not available in character sets of other locales, nor in Unicode. Consider annotated forms of kana. For example, the following are variant forms of hiragana あ (read *a*):

These character sets were designed for information interchange, not professional publishing. Because professional publishers require access to a greater number of glyphs, primarily variant forms of standard characters, many fonts include additional glyphs to serve their needs. Under most circumstances, these additional glyphs are not easily accessed (because they are unencoded). For example, Adobe Systems' Japanese fonts, which have been shipping since the later 1980s, include 250 JIS78 (JIS C 6226-1978) kanji variants. MacOS users have been unable to access these glyphs, although they are included in every such font. It is only with the introduction of sfnt-wrapped CIDFonts that these glyphs have been made accessible.

To summarize, contemporary font formats allow some of their glyphs to remain unencoded. Until recently, this has made their access difficult.

1. http://www.w3.org/People/D%C3%BCrst/

2.2 Glyph Substitution Features

Contemporary outline font formats, such as QuickDraw GX and OpenType, provide facilities for performing glyph substitution. Adobe Systems' MacOS sfnt-wrapped CIDFonts, an extension to Apple Computer's QuickDraw GX technology, include several glyph substitution features that provide users access to glyphs in Japanese fonts that are otherwise unencoded. Glyph substitution features in these fonts include Traditional Kanji, JIS78, and Expert.

To date, Adobe Illustrator (Version 5.5J and Version 7.0 or later), Macromedia FreeHand Version 8.0J, and QuickDraw GX applications can access these glyph substitution features.[1] Glyph substitution is effected through the 'sfnt' resource's 'mort' table. Under OpenType, the 'sfnt' resource's 'GSUB' table accomplishes the same goal.

Glyph substitution features provide the foundation for accessing unencoded glyphs. However, there are clever ways to significantly improve the user experience. Because typical CJKV input is done at the IM level, the IM is the ideal place to provide users with access to unencoded glyphs. Moving this process upstream has significant benefits.

2.3 Single- Versus Double-Stage Input

Given today's state of IMs and application support for glyph substitution features, the process of accessing unencoded glyphs requires two stages:

1. The first stage is the input of encoded characters through the IM. The user may spend a lot of time at this stage ensuring that the characters that are input are those that are desired. This describes today's user input experience.
2. The second stage requires the user to first select one or more characters, then use application-provided glyph substitution features (exposed through the UI) to select the appropriate glyph for the selected characters. This is currently possible using today's fonts and some applications.

One way to improve the user input experience is to effectively combine both stages into a single stage. The IM, in simple terms, needs to provide users with access to unencoded glyphs as though they were encoded, by dynamically reading a font's glyph substitution tables.

The following Japanese phrase, coined by Kazuo Koike on the Font-G mailing list, is a good example for illustrating why single-stage input is critical for enhancing the user input experience:[2]

学校の桜並木に黒い虫

Rendering this phrase using older forms requires access to "Traditional" and "JIS83" (JIS X 0208-1983) glyph substitution features. Changed glyphs have been underlined:

學校の櫻竝木に黑い蟲

The only JIS83 glyph in this example is 校 (compare this with its JIS90/JIS97 glyph: 校)—the rest are considered traditional forms. As you can well imagine, individually selecting these six characters then applying glyph substitution can be a time-consuming process. Integrating the selection of glyph variants into the IM level can significantly reduce the time it takes to input text. In essence, selection of *glyphs* (not only characters) can be performed at the IM level.

---

1. QuickDraw GX applications can also access kana and kanji ligature features that are included in these fonts.
2. This Japanese phrase means "black bugs stick on the cherry trees in the schoolyard."

However, single-stage input requires the input method and applications to perform tasks or feats that have never been done before—by either of them. This additional required functionality is described in the following section.

2.4 Fundamental Changes Necessary For IMs and Applications

In order to enable single-stage input, IMs and applications must fundamentally change many of the processes that they currently perform. These changes include how these two types of tools interact with one another. The following bullet items detail the ways in which these tools must evolve:

- An application must provide access to glyph substitution features through its UI in order to support editing operations or work with legacy documents. The application's file format, of course, must support glyph substitution.
- An IM must communicate with the active application in order to determine what font is selected at the current insertion point.
- An IM must then query the OS through an API to determine whether the selected font has glyph substitution features that can be exposed at the IM level.
- An IM must read the select font's glyph substitution features in order to dynamically enhance the conversion dictionary to include unencoded glyphs along side encoded characters.
- IM must send not only character codes to the active application, but also information that effectively "tags" unencoded glyphs so that the appropriate feature is enabled within the application. Information to be sent includes a tag plus an encoded character (in a format that is understood by both IM and application).

As you can well imagine, an extremely high level of cooperation between application, font, input method, and operating system developers is required to fully realize single-stage input. A common "tagging" system must be developed, for example.[1]

The production workflow for some types of professional publishing requires the use of simple text editors. Access to unencoded glyphs can also be useful in this context, through the use of human-readable tags that are generated by the IM then sent to the application. Such texts can then be imported into an application that understands these tags, and activates the appropriate glyph substitution feature on a per-character basis.

Of course, for users who do not require constant access to variant forms while working with their IM, a switch to disable access to unencoded glyphs will be a welcomed feature. Otherwise they would potentially get inundated with lots of variants (consider the numerous 辺 variants in Section 2.1). There may also be users who prefer to use an applications UI for selecting unencoded glyphs. In the end, the user should be provided with enough flexibility to customize their own working environment.

3. Advantages of Unencoded Versus Encoded Access

It is possible to encode all glyphs, but that would erode some of the benefits gained through glyph substitution beyond the encoding level. Advantages of encoding variant glyphs as "standard form plus feature" include the following common text processing operations:

- Searching
- Sorting

In other words, unencoded glyphs are treated the same as their encoded standard forms because they are associated with them through a font's glyph substitution tables. Consider the numerous annotated forms of

1. XML, for example, may serve as the basis for such a tagging scheme.

hiragana あ provided in Section 2.1. The power of this association can go well beyond unencoded characters by also offering the benefit to encoded variant forms. The Japanese phrase rendered with old forms, that was provided in Section 2.3, uses several traditional forms that *are* included in JIS X 0208:1997.

4. Conclusion

I strongly encourage IM and application developers to work together to develop an open specification for enabling unencoded glyph access at the IM level. The framework described in this paper, which basically calls for making fonts' glyph substitution features available at the IM level (in addition to the application level where it currently is located), can serve as the basis for such work to commence.