Ancient TEX: Using XETEX to support classical and medieval studies

David J. Perry

Abstract

This article provides a brief background on Unicode and OpenType and then explains how they have become important to scholars in classics and medieval studies. XETEX, with its support for Unicode and OpenType, now makes TEX a good choice for scholars working in these fields — particularly on Windows and Linux, where OpenType support is not readily available otherwise.

1 The movement toward Unicode

(If you already have a good understanding of Unicode, you can skip ahead to section 2, or to 2.2 if you don't need an introduction to OpenType.)

Unicode is a project designed to make it possible to use all the living languages of the world, and many historical ones, in an efficient and standardized way. It is developed by the Unicode Consortium, a group that includes software companies, institutions such as universities and governmental agencies, and individuals. The Unicode Standard is developed in coordination with the international standard ISO-10646, known as the Universal Character Set; all characters added to one are also added to the other. (These two projects were begun separately in the late 1990s, but soon were merged since it was not beneficial to have two competing standards.)

ISO-10646 is essentially a list of characters. The Unicode Standard provides additional help to those who need to write software using various scripts; for instance, Unicode provides a bidirectional algorithm to integrate left-to-right and right-to-left scripts as well as guidance about how to work with scripts such as Arabic and the various Indic scripts that have complex shaping requirements. For more information, see the web site of the Unicode Consortium: `http://www.unicode.org`.

During the last 15 years or so, Unicode has become more and more important. All the major computer operating systems (Microsoft Windows, Linux, and Apple's Mac OS X) have been Unicode-based for some time, and much software has been written that takes advantage of Unicode.

Unicode is based on the *character/glyph model*. Under this system, Unicode encodes *characters*, basic phonemic or semantic units. It does not concern itself with the fact that these characters may appear in different forms on a page; the exact shape that a character assumes in a given context is referred to as a *glyph*. Two examples will clarify this distinction.

1. The character LATIN SMALL LETTER A may appear as a, **a**, **a**, *a* or as many other shapes, depending on the typeface and style (italic, bold, small capitals, etc.) chosen by the author or designer.
2. In Arabic, letters take on different shapes depending on whether they are the first letter in a word, appear in the middle of a word, or come as the last letter of a word. Unicode encodes one general set of Arabic letters, corresponding to the forms used in isolation (as when a reference book shows "the Arabic alphabet" in a table). In order to display Arabic properly, software must take a string of these basic Arabic letters and apply the correct forms as called for by the context.

The character/glyph model enables Unicode text to be stored in an efficient and permanently valid form. In the case of the Latin script, it would obviously be impossible and undesirable to attempt to encode permanently every different letter shape. For Arabic, the same text may be processed at the present time on a Windows system using OpenType or on a Mac using AAT, or new technologies may be developed for other computer systems in the future; but the underlying text remains valid.

For scholars in fields such as classics, biblical studies, and medieval studies, Unicode provides two important, related advantages:

- the ability to mix different scripts and languages easily in one document
- a standardized, internationally recognized, and permanent set of characters

A biblical scholar, for instance, might need to use ancient Greek, Hebrew, and Latin, along with one or more modern languages. While it has been possible for some time to mix languages on most computer systems, this was not always easy, particularly if one wanted to mix right-to-left and left-to-right scripts.

The case of ancient Greek provides a good example. It requires three accents, two breathing marks, a special form of the letter iota written below other vowels, and a few additional signs. Neither Apple nor Microsoft ever created any standard for ancient Greek, so each font maker set up his own system of matching Greek letters to various positions in the Latin alphabet and their corresponding keystrokes. (Prior to Unicode, users could access no more than 256 characters at one time, so a single font could not support, e.g., Latin and Greek.) By the time it became practical to use Unicode Greek (about

1996), there were several Greek fonts in use by classicists, each different from the others. Exchanging text with colleagues was difficult unless they happened to be using the same font. Without the appropriate font (or at least a table stating what Greek letters were mapped to what Latin ones), the meaning of a given text could be entirely lost. The situation with biblical Hebrew was similar.

This is very different from the situation in mathematics; the development of TEX and its adoption as a standard early in the personal computer era meant that mathematicians did not feel the same urgency as classicists did to move to Unicode.

Unicode changed the multilingual landscape. Classicists and biblical scholars eagerly adopted Unicode Greek and Hebrew, for they recognized the advantages of a standardized format that was internationally recognized and not dependent on the use of a particular font. Unicode fonts can contain more than 64,000 characters, although most contain far fewer. Therefore one can potentially use a font that contains Greek or Cyrillic letters designed to harmonize with the Latin forms; the text looks good and one need not worry about switching fonts.

2 The importance of OpenType

All is not perfect in the marriage of scholarship and Unicode, however. Classicists and medievalists have embraced Unicode because we appreciate its many benefits and because we do not want to be left out as the computing world becomes more Unicode-centric, but the character/glyph model is not a perfect fit for our needs. There are three important issues for which Unicode by itself does not provide a good solution: glyph variants, unusual combinations of diacritical marks and base letters, and non-standard ligatures. OpenType provides a solution for all these issues. Before discussing how scholars can use OT to address their specific needs, we give some background about OT in general.

2.1 OpenType basics

The OpenType specification was created jointly by Microsoft and Adobe. It provides many different tools that enable a string of Unicode characters to be displayed in ways are linguistically appropriate and typographically attractive. These tools are referred to as *features*.

Some features are used to render a string of Unicode characters in ways that are required for text to be considered correct by users. For Arabic, OT provides features to replace the basic letters with the forms needed if a letter is the first or last in a word, as explained above. An Arabic-capable word processor applies these features automatically as the user types, so the user does not have to worry about them; the resulting text displays in normal Arabic fashion. The font developer must do what is required to ensure that the features operate correctly. In the case of Arabic, this means putting additional glyphs into the font for initial, medial and final forms and setting up tables so that when, for instance, the application calls for the word-initial form of a letter, it can locate the proper glyph to use.

Another example: the Serbian language may be written in either the Latin or the Cyrillic script. When using the latter, Serbians employ a few letter shapes that are slightly different from those used in Russia. OT fonts can contain a feature that specifies which shapes to use for which language. There is no question that the same alphabet — Cyrillic — is used for both languages, and it would be very undesirable to encode the Serbian shapes separately. OT makes it possible to have standard Unicode text displayed appropriately for Serbian or Russian readers.

Other OT features are used to provide high-quality typography in scripts such as Latin, Greek, and Cyrillic that, unlike Arabic or Indic scripts, do not require complex processing. An OT font can contain true small capitals, various varieties of numbers (lining numerals, oldstyle ["lowercase"] numerals, and both proportionally spaced and monospaced versions of either style), ligatures (fi, ff, etc.), and many other typographic refinements. These features, unlike those required for correct display of Arabic, usually do not display unless specifically requested by the user. An application that supports high quality typography via OT must provide an interface for this purpose.

In short, OT is a two-headed beast. Microsoft originally adopted it as a means to get Unicode text to display properly in languages that have complex script requirements. Adobe has been more interested in the typographic possibilities of OT in standard scripts and has promoted its use by releasing OT versions of Adobe fonts and by providing access to OT features in programs such as the advanced InDesign page layout program.

We should note that Mac OS X includes a technology called AAT (Apple Advanced Typography) that does many of the same things as OT, both to implement complex scripts and to provide high-quality typography in standard scripts. AAT has not met with great success, partly because it is more difficult for font developers to create AAT than OT fonts. In response, Apple has enhanced OS X (beginning with version 10.4) so that it now processes and displays many features found in OT fonts.

OT font files are cross-platform (Mac, Windows,

Figure 1: Shapes of the centurial sign.

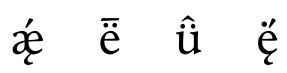

Figure 2: Some medieval combinations of diacritics.

Unix); the basic text will always display properly, but implementation of advanced OT typographical features is up to the application.

2.2 OpenType and scholars

As mentioned above, there are three areas in which Unicode does not adequately meet the needs of classicists and medievalists. Let's look at each in turn.

Some characters appear in shapes that vary considerably, depending on when and where the text was created. For instance, Roman inscriptions often contain a symbol that represents the word *centurio* (centurion, the Roman equivalent of a sergeant) or *centuria* (century, a military unit of 100 men). This centurial sign may take the shapes shown in Figure 1, which are referred to as *glyph variants*.

The centurial sign was recently accepted into Unicode. This is good because the character can now be stored in electronic texts in such a way that its identity will always be understood. But what if the editor wants to display the same shape as found on the original stone, when that is not the same as the Unicode reference glyph? Recall that under the character/glyph model, Unicode does not normally encode variant shapes for characters. An OT font can contain a number of alternate glyphs for a character, using the Stylistic Alternates feature. After entering the standard Unicode value for the centurial sign, the user can apply the Stylistic Alternates feature and select the desired glyph shape. This is a neat solution to a difficult problem. If a character from the Private Use Area were used to print the variant, its value might be lost if the proper font was not available in the future or if the text was copied and pasted into another application. (The Private Use Area is a range of codepoints that will never be defined by Unicode, i.e., they will always be officially left empty. Users can create customized fonts and put non-Unicode characters in the PUA for their own purposes. While the PUA can be useful, it is inherently unstable and characters in it should never be used in texts intended to have a long life, such as electronic editions of literary works.)

Medieval manuscripts contain dozens of combinations of letter plus diacritical mark(s) that are not used in any modern language and therefore are not directly supported by any operating system; see a few examples in Figure 2. (These examples are taken from the Character Recommendation of the Medieval Unicode Font Initiative, `http://www.mufi.info/`.) A few such combinations are also needed for ancient Roman inscriptions. Unicode provides all the needed diacritics in the Combining Diacritical Marks and Combining Diacritical Marks Supplement ranges. However, if a user simply types a base letter followed by a diacritic, there is no guarantee that the diacritic will be centered or otherwise placed appropriately over the base. Furthermore, good typographic practice is to replace the normal dotted i with the dotless 'ı' before applying an accent above the i. OpenType fonts can be set up to handle proper placement of diacritics and the substitution of dotless i as needed. (The original design of Unicode envisioned that operating systems would be able to place any combining diacritic appropriately and automatically. This vision is taking a very long time to be realized. Mac OS X was the first to attempt it, by looking at the widths of the characters in the font. The results are frequently acceptable, though some combinations need manual adjustment. Windows Vista has now taken some very limited steps to implement combining diacritics. But for now, and probably for some time to come, we need to rely on information built into each font in order to get diacritics working properly.)

Finally there is the matter of ligatures. These are found in ancient Greek and Roman inscriptions and even more frequently in medieval manuscripts. They were used to save space on stones and to save time for scribes. OpenType supports the standard f ligatures used in modern printing (fi, fl, ff, ffi, and ffl) through its Standard Ligatures feature. It also provides a feature called Historical Ligatures. An OT font designed to support epigraphy could include an entry in the Historical Ligatures feature to replace the letters NT with the ligature commonly found in Roman inscriptions, if the user applied this feature to a run of text.

It should be emphasized that even if an alternate glyph or an historical ligature is presented to the reader via OT features, the underlying Unicode text is not changed. This is important in regard to searching and reusing text. A user, for instance, might not know about all the varying shapes of the Roman centurial sign; even if he or she did know them all, it is not desirable to require multiple searches in order to cover all possibilities. If the user enters the standard Unicode value for the centurial sign

when searching, the proper results will be returned, regardless of which glyph is shown in the document.

Likewise, a user can copy some text that is displayed with unusual ligatures and paste it into an application that cannot handle OT features. The underlying letters will be shown, not some random characters, so that the text is still meaningful, even if not displayed in its historical form.

2.3 Software support for OT features

So it seems that classicists and medievalists now have a good solution to many of their needs, using OT features for display on top of Unicode text. The problem is that support for OT has been slow in coming. Mac users are best off. The word processor Mellel was developed around OT (rather than AAT) and provides good support. Some of Apple's own applications, such as the word processor Pages, include a Typography palette that provides access to AAT or OT features, whichever a specific font offers. Both Mellel and Pages are reasonably priced. The high-end page layout programs Adobe InDesign and Quark Express (v7 or later) offer outstanding Unicode and OT support, but are prohibitively expensive for many users.

On Windows, support for high-end typography is provided only by InDesign and Quark Express. Windows Vista includes some APIs that make it easier for software developers to access OT features, but so far developers have not taken advantage of them — including those responsible for Microsoft's own Office suite. The situation is equally bleak in the Linux world. Neither OpenOffice nor Scribus yet supports OT features on any platform.

This situation is very frustrating to scholars. We need to use Unicode, for the reasons explained above, and we understand that the character/glyph model just does not allow for glyph variants or unusual ligatures or diacritic combinations to be encoded. OT does provide a solution that works, but software support is extremely limited, particularly for Windows and Linux users.

What does the TEX world offer for our needs?

3 XeTEX brings it all together

3.1 XeTEX basics

Released in 1994 by Jonathan Kew, XeTEX was originally available for Mac OS X and then was ported to Unix and Windows. It extends the functionality of TEX and LATEX in three important ways.

- XeTEX provides direct Unicode support. Users can mix scripts, use large fonts, and access any Unicode character, as explained above. They can also use the standard methods to which they are accustomed when entering text. For example, if a Windows system is set up to handle polytonic Greek or Hebrew as well as English, the user can employ the icon in the system tray or the normal ALT-LEFT SHIFT combination to switch easily between languages and their associated keyboard layouts.
- XeTEX allows users to take advantage of OT and AAT features that may be present in a font.
- XeTEX enables users to access all fonts installed on the system without the need to create special configuration files for each font.

3.2 Encouraging new users to try XeTEX

Until the creation of XeTEX, TEX was not an ideal choice for classicists and medievalists. Their world is becoming more Unicode-centric, and they are hoping that OT will solve many of the problems that Unicode presents for their work. Furthermore, they very often need special fonts — after all, support for ancient epigraphy or medieval manuscripts is not a concern to most font makers — and such fonts are nowadays all Unicode-based. Being able to use installed Unicode system fonts without the complicated configuration process previously required by TEX removes an important barrier for new users. Since support for advanced OT typography in standard scripts is available only in a very small number of expensive applications under Windows and not at all in Linux except for XeTEX, those who have a real need for OT features should seriously consider using XeTEX.

New TEX users, and old hands who advise them, should be aware of the following:

- XeTEX is now included in most TEX distributions, so users will already have it.
- To take full advantage of XeTEX, a Unicode-based text editor or integrated environment is necessary; some of those still in use in the TEX world can handle only ASCII, such as WinEdt and TEXnicCenter (the latter will be Unicode-capable in v.2, according to its web site); Texmaker handles Unicode but knows nothing about XeTEX yet.
- Jonathan Kew and others are now developing TEXworks, an easy-to-use integrated environment for document creation that fully supports XeTEX. While it has not yet been officially released, working versions can be obtained from the project's web site: `http://www.tug.org/texworks/`. Alain Delmotte has written an introductory manual for TEXworks and also provides up-to-date binaries for those who do

not wish to compile the software themselves; see `http://www.leliseron.org/texworks/`. I used TEXworks to prepare this article, so it is certainly functional, albeit with a few rough edges. I regard it as the best choice for beginners with XETEX at the present time.

- For those who are willing to work with a plain text editor, Notepad (bundled with Windows) and BabelPad (at `http://www.babelstone.co.uk/Software/BabelPad.html`) will do the job; the latter is particularly Unicode-friendly.
- I have written an article intended for scholars in classics and medieval studies who want to begin using XETEX; it is available from `http://scholarsfonts.net`. Experienced TEX users, especially those who have read this article, will not find much new there, but they might want to pass it along to colleagues who seek aid in using TEX. It does contain more information about the fontspec package and OT, including a table that sorts out the names of features. (fontspec uses names that do not exactly match the standard OT names, which can be confusing.)

3.3 Using XETEX

Using XETEX is not difficult. You need to add a few packages to your preamble: fontspec, xunicode, xltxtra, and perhaps polyglossia. The first, fontspec, is very important because it helps XETEX select fonts and is the only practical way to apply OT or AAT features. Documentation for it is included and will be accessible to those experienced with TEX; newcomers will find it a bit tricky. The second, xunicode, enables users to employ traditional TEX shortcuts such as `---` for an em-dash; neither it nor xltxtra requires any action on the user's part once added to the preamble.

To add support for language-specific hyphenation and punctuation, use the polyglossia package; see its documentation for the various options, which should be understandable by anyone with a basic knowledge of TEX. It is a replacement for the babel package, which should not be used with XETEX.

If you are using TEXworks, you can start a new file by using `File / New` from `Template ...` and choosing one of the XELATEX templates. This will get you fontspec and other packages you need.

3.4 Some samples

To conclude this article, we will provide some samples of what can be done with XETEX and OT features. The following is by no means a complete illustration of what OT can do, but it will, we hope, whet the appetites of readers to explore OT further. All samples make use of Junicode, a font for medievalists described in section 4 below. OT features are called through the fontspec package. In these examples I used fontspec's `\addfontfeature{}` command, which provides an easy way to apply features to short runs of text. There are other ways, such as setting defaults in the preamble if you want a feature to be used throughout a document.

Keep in mind that even though some of these samples look unusual, the underlying text consists of regular letters and numbers, and (for instance) a PDF file containing such text can be easily searched without inputting any special characters. The first three samples illustrate OT features that are helpful for setting high-quality text in any Latin-script language, while the rest are specific to medieval studies.

3.4.1 Oldstyle numerals

The following code produces the result shown in Figure 3.

```
default numbers: \quad 1234567890 \\
{\addfontfeature{Numbers=OldStyle}
with Oldstyle on: \quad 1234567890 }
```

default lining numbers: 1234567890
with Oldstyle feature on: 1234567890

Figure 3: Lining versus oldstyle figures.

3.4.2 Fractions

The following code produces the result shown in Figure 4.

```
Without fractions:
\quad 1/2 \quad 2/5 \quad 2/3 \quad 7/8 \\
{\addfontfeature{Fractions=On}
With fractions on:
\quad 1/2 \quad 2/5 \quad 2/3 \quad 7/8 }
```

Without fractions: 1/2 2/5 2/3 7/8
With fractions on: ½ ⅖ ⅔ ⅞

Figure 4: Creation of true typographical fractions.

3.4.3 Small capitals

Many OT fonts contains properly designed small capitals. (This is not the same as the "small capitals" found in programs like Microsoft Word, which are scaled-down capitals that do not follow traditional design principles for small caps.) OT provides a feature to invoke small capitals and another that changes only uppercase letters to small caps. The latter is useful for abbreviations that are typed in caps but look better as small caps when mixed in running text. The following code produces the result shown in Figure 5.

```
quick brown fox
{\quad \addfontfeature{Letters=SmallCaps}
quick brown fox}\\
the NATO alliance
{\addfontfeature{Letters=UppercaseSmallCaps}
\quad the NATO alliance }
```

quick brown fox   QUICK BROWN FOX
the NATO alliance   the NATO alliance

Figure 5: True small capitals.

3.4.4 Historical forms and historical ligatures

OT's Historical Forms feature allows the user to turn on shapes that are appropriate only in historical contexts, such as the long s and its ligatures, which were used in English through the 18th century. Junicode uses the Historical Ligatures feature to access ligatures found in medieval manuscripts. Note that one can turn on more than one feature at a time in fontspec by separating the features with a comma. The following code produces the result shown in Figure 6.

```
same silly distant \quad AA aa AY ay ag al}
{\addfontfeature{Style=Historic,
  Ligatures=Historical}
same silly distant  \quad AA aa AY ay ag al}
```

same silly distant   AA aa AY ay ag al
ſame ſilly diſtant   Ꜳ ꜳ Ꜽ ꜽ ag al

Figure 6: Historical forms and historical ligatures applied to text.

3.4.5 Language-specific features

The letters thorn and eth were used in Old English and are still employed in modern Icelandic. Junicode's default is to use the Old English shapes. Those who prefer the Icelandic forms can access them as shown here. The following code produces the result shown in Figure 7.

```
Default Old English shapes: \\
\quad {\Large Þþ Ðð}  \\
\addfontfeature{Language=Icelandic}
Icelandic shapes now used:
\quad {\Large Þþ Ðð}
```

Default Old English shapes: Þþ Ðð
Icelandic shapes now used: Þþ Ðð

Figure 7: Use of language-specific forms.

4 Resources

To learn more about Unicode, OpenType, and XeTeX, an excellent place to start is Michel Goossens's *The XeTeX Companion: TeX Meets OpenType and Unicode*, currently available at `http://xml.web.cern.ch/XML/lgc2/xetexmain.pdf`. Written with an eye toward those who already have some familiarity with TeX, it provides more in-depth information than what is found in this article.

The web site of the Unicode Consortium, `http://www.unicode.org`, offers a great deal of information, including the entire text of *The Unicode Standard* in downloadable PDF form.

Here are some options if you want to experiment with the advanced typographical features of OpenType:

- The Junicode font by Peter Baker, freely available from `http://junicode.sf.net/`. The zip download includes some documentation that was created with XeTeX.
- Linux Libertine by Philipp Poll (freely available from `http://linuxlibertine.sf.net/`) is another nice font family with many OT features; despite its name, it also works on Windows and Mac OS X.
- TeX Gyre is a project to update and extend the fonts distributed with the open-source Ghostscript page description language. It includes a number of fonts, each in OpenType and Type 1 formats. The OT versions contain many features for advanced typography, all of which are identified in the documentation. Latin Modern does the same for TeX's Computer Modern fonts. See `http://www.gust.org.pl/tex-gyre` and `http://www.gust.org.pl/lm`, respectively.
- If you have access to any of Adobe's Pro fonts (Warnock Pro, Minion Pro, etc.), these also contain OT features. Adobe's online font catalog at `http://www.adobe.com/type/` shows what features are included in the various fonts they sell (not all fonts have all features).

If you are curious about how characters, particularly scholarly ones, get added to Unicode, you can look at the proposals for medieval characters prepared by the Medieval Unicode Font Initiative at `http://www.mufi.info/` or at my proposals for classical Latin characters at `http://scholarsfonts.net/latnprop.html`.

◇ David J. Perry
Rye High School
Rye, New York
USA
hospes02 (at) scholarsfonts dot net
http://www.scholarsfonts.net