

fonts

The Design and Use of a Multiple-Alphabet Font with Ω

Yannis Haralambous
Atelier Fluxus Virus,
187, rue Nationale, 59800
Lille, France.
Email:
yannis@fluxus-virus.com

John Plaiice
School of Computer
Science and Engineering,
The University of New
South Wales, Sydney 2052,
Australia.
Email:
plaiice@cse.unsw.edu.au

abstract

The Ω project aims to offer open and flexible means for typesetting different scripts. By working at several different levels, it is possible to offer natural support for different languages and scripts, and strictly respect typographical traditions for each of them. This is illustrated with a large PostScript Type 1 font for the commonly used left-to-right non-cursive alphabets, called `om1gc` (Ω Latin-Greek-Cyrillic). This font, which more than covers the Unicode sections pertaining to those alphabets, as well as those of IPA, Armenian, Georgian and Tifinagh (Berber), is built—virtually—out of smaller glyph banks. The Ω typesetting engine, based on that of \TeX , is used to print documents using this font. The characters can be accessed either directly, or through the use of filters, called Ω Typesetting Processes (Ω TPs), which are applied to the input stream.

1 Introduction

Typesetting in different scripts and languages is a problem that is arguably solved today, either by software giants that adapt their products to what they consider to be local typesetting specifications, or by individual users brewing their own limited, but practical, systems for their own needs.

However, typesetting in different scripts and languages—*while still keeping the quality of traditional typography*—and having an open system that can be adapted to any language and script, without loss of power or quality, is still an unachieved goal. The Ω project aims to solve precisely this problem.

The Ω project consists of the design of fonts for different languages and scripts, as well as of an engine that can be used for all possible situations.

As a result, in order to ensure efficiency and openness, one can work at different levels, each one adapted to a specific aspect of multilingual typesetting. These levels correspond to methods used in the Ω system. They will be illustrated through the `om1gc` font, designed for—currently—Latin, Greek, Cyrillic, IPA, Armenian, Georgian and Tifinagh.

This paper was typeset by the Ω engine, using fonts `om1gc` and `omarab`.

2 Ω Methods: an Overview

When developing an Ω typesetting convention for a given language, one can work at the following levels:

1. *The font level.* A font is a container of *glyphs* needed to typeset in a given language. These glyphs may or may not correspond to the graphical units of a script, whether these are called “letters”, “ideograms” or otherwise; these glyphs may be parts of graphical units, combinations thereof, or new independent symbols.

The glyphs must be provided to the screen previewer and as well as to the printing engine. Ω itself is not concerned with these glyphs: as with \TeX , Ω works only with metrics, not necessarily those for the fonts actually containing the glyphs.

The font level is the lowest development level, in the sense that glyphs are indivisible units that can be used in other higher level structures but cannot be dynamically modified by Ω itself.

2. *The virtual font level.* Once we have the glyphs we need, we combine them to form what is normally considered for a language to be a grammatically correct script entity (a ring accent alone is of no use, but å is part of the grammar for the Swedish language).

A virtual font character is a combination of glyphs taken from several “real” fonts, or of other virtual font characters. In the å example above, the glyphs of the ring and of the letter ‘a’ can actually be taken from entirely different fonts, in different formats (bitmap, PostScript, TrueType, etc.).

Passing from “real” fonts to virtual fonts is essentially a matter of optimization of storage and memory: taking the seven Greek vowels, the three accents, two spirits, diaeresis, subscript iota and macron/breve diacritics (that makes 16 glyphs) we produce 336 (!) virtual glyphs. Describing a character in a virtual font is hundreds of times smaller than the PostScript code describing a hypothetical similar glyph.

Virtual fonts are directly used by Ω : they can have up to 2^{16} positions and 2^{32} kerning pairs. The files created by Ω can be processed (previewed, printed) by utilities we have adapted; if the user has to use his/her own utilities which are not “16-bit clean”, there is a tool to “devirtualize” the files, i.e. replace virtual fonts by the underlying real fonts, and make them as standard as \TeX output files.

3. *The Ω TP level.* When Ω reads a document, it first tokenizes it and expands commands and then forwards the data to the typesetting engine. Between these two steps we introduce an arbitrary number of filters, which we call Ω Typesetting Processes (Ω TPs). These are written in a Lex-like syntax, are loaded while reading the document, and can be activated and de-activated dynamically.

A typical example for an Ω TP is contextual analysis of Arabic. Of course, this operation can also be done by macros, say using \LaTeX commands; but an Ω TP will do it much faster, it will avoid conflicts with other \LaTeX commands, and it will be much easier to create and configure.

Contextual analysis of Arabic is a typical example of a script property that is low level and that should not use *any* macros or other high-level structures. In the case of the Latin script one would not expect a user to constantly think of ‘fi’ and ‘ff’ ligatures and place them manually: it would be very bad strategy to use an “fi ligature command”; in Arabic this property is of the same nature, and hence should remain completely transparent.¹

But efficiency should not only be limited to speed of typesetting: sometimes it is very important to also optimize the configuration and customization time and effort. A typical example is the management of encodings, whether input or output: by using a universal encoding—which we call it *Unicode++*, since it is a superset of Unicode—as intermediate step for our Ω TPs, every new input encoding requires only a $\text{foo} \rightarrow \text{Unicode++}$ Ω TP and every new font encoding only a $\text{Unicode++} \rightarrow \text{foo}$ one; these are significantly easier to make than if one had to rewrite all processing steps, including contextual analysis and other bells and whistles.

4. *The hyphenation and sorting engines.* Hyphenation and sorting rules are grammatical properties of a language: they have nothing to do with typographical aspects, input methods, font encodings, etc. They only have to be performed on the Unicode++ level, i.e. the most abstract one. Once defined at this level, they can immediately be used with every input and output encoding. Ω hyphenation and sorting engines are still under development: for the time being, Ω does not sort, and hyphenation takes place as in \TeX , using the (virtual) font encoding.

1. And undoubtedly this would be the case if the *lingua franca* of computer science was Arabic or Urdu, and not English.

5. *The macro-command level.* L^AT_EX is a terrific construction, featuring commands at multiple levels: T_EX primitives, plain T_EX commands, internal L^AT_EX commands, higher L^AT_EX commands and environments, and user-defined commands and environments. Our goal is to keep all script-dependent processing independent of the L^AT_EX macro level; because of this goal, every L^AT_EX package is usable in any script and language, and both Ω package and L^AT_EX package developers are able to work independently, while still producing mutually compatible software.

By doing this, we also allow ourselves in the future to use a different typesetting engine than that found in T_EX, without having to completely redo our system. The ΩTPs might need modification, but the basic structure would remain the same.

3 Level 1: Designing glyphs

For us, a “glyph” is simply a character from a PostScript, TrueType or Metafont font; so why call it such? Because we want to make the distinction between the “images” and the “combinations of images”, the latter of which Ω will use as characters for typesetting.

For our om1gc font, the glyphs are produced so that the glyphs for different alphabets look similar. With the exception of Tifinagh, the history of the different alphabets making up this font is strongly interrelated; as such, it has been possible to design a single font with a characteristic feel. For Tifinagh, the script has evolved less than the others, so we are doing creative experimentation—only time will tell if it is successful, i.e. native speakers accept it.

For example, the following two words Հս Պալաւսանեանի are Armenian. Although the traditional Armenian script does not look so rounded, it is considered common practice today to typeset books in this manner, and Armenians are not in the least surprised.

The om1gc font is a modern looking font. One question that might arise is what to do with classical versions of these alphabets. For ancient Greek, it is common to use modern typefaces, even though there are typefaces specifically made for ancient Greek, such as Porson or Greek Sans Serif. Can this also be done for Coptic and Slavonic excerpts, without breaking the homogeneity of the surrounding “modern” font? We have tried to do this for Slavonic—including not only letters and accents found in Unicode, but also all the other diacritics and symbols needed for Old Church Slavonic—and our example looks like this: Цѣтъа ѣй ѣсъ ѣ речѣ ѣй. The result is not particularly exiting to look at, but that is the point: it should fit with the surrounding text, not look “exotic”.

Finally, a lot of work has gone into the design of IPA glyphs, which are problematic since many of them are basically font variations (italics, small caps, etc.) of other glyphs, thereby making it a nightmare to design fonts including IPA that have variations.

As we will see in the next section, a big part of the om1gc font is obtained by combining a limited number of glyphs. This corresponds to a grammatical reality: it is quite natural to assume that placing diacritics does not affect the shapes of either the base character or the diacritic itself. Often this is true, but there are times when typographical quality requires special shapes.

For example, one can imagine an ‘h’ with a lower vertical stem, so that the accent of the Esperanto character ĥ is not excessively high. This is really a matter of taste, and we include these kinds of characters as variant forms in our font, and leave the decision whether to use them to the user.

Another similar example comes from the use of diaereses over the Greek vowels iota and upsilon, whether lowercase or uppercase; the distance between the two dots of the diaeresis depends on the letter underneath: compare *ï*, *ü*, *ÿ*, *ÿ̂*, while in the Latin script the diaeresis (Umlaut, tréma) always has the same width: *ï*, *ö*, *ÿ̂*, etc.

4 Level 2: Combining Glyphs into Virtual Fonts

A character of a virtual font can be a combination of characters of other fonts (whether real, in which case we actually have glyphs, or again virtual, in which case we have sub-combinations etc.), of black boxes of arbitrary height and width, and of PostScript code. (Since the latter feature is not implemented in all DVI drivers, one should refrain from using it, at least for the moment.)

We have chosen to work intensely with virtual fonts for two reasons: first because by combining glyphs we can optimize space (and space management is crucial when you deal with 16-bit fonts), and second, to be able to use 16-bit fonts without re-writing all of the world's drivers for T_EX's DVI format (the underlying real fonts are 8-bit only, so that a devirtualized 16-bit font becomes 8-bit and can be processed by any decent DVI driver).

Virtual fonts are built using a (portable) Perl script, which reads the configuration file `omlgc.cfg`. This file contains one entry for each character of the virtual font. This line consists of (a) a 4-digit hexadecimal number, designating the character's position in the font, (b) an operator, (c) one or more character names, depending on the operator, (d) a certain number of optional operators and values.

The N operator.

The N stands for "NAME", and simply means that the string following the operator is the (PostScript) name of one or more characters in some of the fonts loaded. For example,

```
03A5 N Upsilon
```

means that at position 03A5 of the virtual font, we have placed the Greek letter capital Upsilon Υ. The PostScript names we have used try to be as standard as possible. However, most of the time there is no standard, so we just have to invent names.

The same glyph can be used for several font positions: for example, the Croatian Dze Đ has exactly the same shape as the Icelandic Eth Ð: we use the same glyph, and hence the same PostScript name. Nevertheless we try to optimize the use of glyphs so that one can typeset in one script without necessarily loading the PostScript fonts for other scripts: for example, although the capital Latin 'A' has the same shape as the Greek capital Alpha, we use two different glyphs in two different PostScript fonts, so that when typesetting Greek one can avoid loading the Latin PostScript font. These considerations will be irrelevant when we will be able to use 16-bit monobloc PostScript fonts; for the moment, this is not part of the Adobe Type 1 font specification.

There are several options we can use: `#KRN`, `#KRNLEFT`, `#KRNRIGHT`, `#HOFFSET`, `#VOFFSET`.

The first three concern kerning: we can state that a given character has the same kerning behavior as some other character, which we give by name. For example, ç will be kerned exactly like the letter 'c': everytime there is a kerning pair in the kernings file using 'c' a new kerning pair will be defined, using ç instead of 'c' (and if there is a 'c-c' kerning pair, three new kerning pairs will be defined: 'ç-c', 'c-ç', 'ç-ç'). Sometimes we kern a letter like some given letter on the right and like some other letter on the left: this is typically the case of ligatures: æ will be kerned as 'a' on the left, and as 'e' on the right; in that case, we use the `#KRNLEFT` and `#KRNRIGHT` operators:

```
00C6 N AE #KRNLEFT=A #KRNRIGHT=E
00E6 N ae #KRNLEFT=a #KRNRIGHT=e
0110 N Eth #KRN=D
```

The operators `#HOFFSET` and `#VOFFSET` will offset the glyph without affecting the box of the character.

The XHAC and CHAC operators.

These operators will place diacritics over letters, which are considered to have the same height as the lowercase letter ‘x’ (x-height) or the one of an LGC uppercase letter (cap-height). The idea is that the height of letters can fluctuate: a round letter, like ‘o’, is slightly higher than a flat letter, like ‘z’, to counterbalance a well-known optical effect. The height of accents over these letters must be the same, even if they aren’t exactly of x-height, or cap-height: take for example ó and ú; if we would take the real height of these letters, the accent on the former would be slightly higher than the one on the latter.

How is this accent placed precisely? The Perl utility centers the bounding box of the accent over the bounding box of the letter, with a fixed distance of EPSILON (a global value) between the lower boundary of the accent and either the x-height or cap-height of the font (those two are also global values provided with the utility).

Available options are #KRN, #KRNLEFT, #KRNRIGHT, #LETTERLIKE, #ACCENTLIKE and #LETTERREVLIKE.

The options #LETTERLIKE and #ACCENTLIKE allow us to use given glyphs, with the metrics of other glyphs. These options are extremely important in certain cases. A typical example is Vietnamese: the letter ‘o with hook’ ơ is significantly wider than the plain ‘o’, nevertheless accents have to be centered on the “o part” of the letter: ớ, ò, ỏ, õ. This trick allows us to correctly place an accent on the vertical stem of a ‘b’ or an ‘h’: ắ, ằ, ẳ; to obtain this result, we simply ask the accent to be placed as if the letter was an ‘l’:

```
0603 CHAC b dot #LETTERLIKE=1
0623 CHAC h dot #LETTERLIKE=1
0627 CHAC h dieresis #LETTERLIKE=1
```

We have the same functionality with accents: using the #ACCENTLIKE operator we can place an accent as if it was some other accent. The typical example is again Vietnamese, where there are combined accents ‘circumflex + grave’, ‘circumflex + acute’, which have to be centered with respect to the middle axis of the circumflex (and hence as if there were no acute or grave accent):

```
06D0 CHAC 0 circumflexacute #KERN=0 #ACCENTLIKE=circumflex
06D1 XHAC o circumflexacute #KERN=o #ACCENTLIKE=circumflex
06D2 CHAC 0 circumflexgrave #KERN=0 #ACCENTLIKE=circumflex
06D3 XHAC o circumflexgrave #KERN=o #ACCENTLIKE=circumflex
06D4 CHAC 0 circumflexhook #KERN=0 #ACCENTLIKE=circumflex
06D5 XHAC o circumflexhook #KERN=o #ACCENTLIKE=circumflex
06D6 CHAC 0 circumflextilde #KERN=0 #ACCENTLIKE=circumflex
06D7 XHAC o circumflextilde #KERN=o #ACCENTLIKE=circumflex
```

Another example is Slavonic, with letters such as ѣ, where the accent has to be placed on the right part of the ligature, as in ѣ́. This means that we should use the metrics of a given character, justified on the right of our box: this is the rôle of the #LETTERREVLIKE operator.

The ADJ operator.

This operator allows us to concatenate two characters, using a box with width equal to the sum of the widths of the two boxes, ± the eventual kerning between those characters, and height/depth the maximum height/depth of the two characters. We first wanted to use that operator for the Croatian digraphs ‘Lj’, ‘Nj’, etc. but then decided that the whole idea of having code positions for these digraphs was so silly that we could very well do without. The operator nevertheless proved very useful for cases such as the Greek capital vowels with accent Α, Ε, Η, Ι, Ο, Υ, Ω.

This operator takes a special option, #MOVELEFT, which allows us to override a kern between two characters and move the second one horizontally, together with its box.

5 Levels 3 and 4: ΩTPs and L^AT_EX Macros

Once the structure of fonts is well organized, we use ΩTPs for low level script- or language-dependent operations and macros for higher level operations. To give an idea of the power of ΩTPs, consider the following Latin transcription for Berber:

```
Tifinagh, d--tira timezwura n .imazighen.
Llant di tmurt--nnegh dat tira n ta.erabt d--tla.tinit.
Nnulfant--edd dat .imir n ugellid Masinisen. .Imazighen n
.imir--en, ttarun--tent ghefi.zra, degg .ifran, ghef .igduren,
maca tiggiti ghef i.zekwan : ttarun fell--asen .isem n umettin,
d wi--t--ilan, d wayen yexdem di tudert--is akken ur t ttettun
.ina.tfaren.
```

This piece of text can be printed using the Latin script, as in:

```
Tifinay, d_tira timezwura n imaziyen. Llant di tmurt_nney dat tira n taerabt
d_tla_tinit. Nnulfant_edd dat imir n ugellid Masinisen. Imaziyen n imir_en,
ttarun_tent yefizra, degg ifran, yef igduren, maca tiggiti yef izekwan : ttarun
fell_asen isem n umettin, d wi_t_ilan, d wayen yexdem di tudert_is akken ur t
ttettun inatfaren.
```

or in the original Tifinagh script, as in:

```
XZJEZI:, A_XZO· XZJZ=:O· I ZC·KZ:I. IIV·IX AZ XC:H_I\A: A·X XZO· I X·O·HH
A_XII·EIZIX. I\IIE·IX_AA A·X ZEZO I :XIIWZΛ E·MIZMI. ZC·KZ:I I ZEZO_I,
XX·O·I_XIX :JE#O·, AXX ZJEO·I, :JE ZXA·OI, E·C· XZXK+Z :JE Z#=:I : XX·O·I
JCIW_·MI ZME I :E+XZ:I, A =Z_X_ZII·I, A =·ΣI Σ::AE AZ X:AH_ZM ·=:I :O X XXXX:I
ZI·EJC·OI.
```

In fact, Berber can also be written right-to-left in the Arabic script (font omarab):

```
تيفيناغ، دتيرا تيمزورا ن ئمازيغن. لانت دي تمورت نغ دات تيرا ن تاعرابت
دتلاطينيت. تولفانت د دات ئمير ن وقليد ماسينيسن. ئمازيغن ن ئميرن، تارون تنت
غفيزا، دق نفران، غف نفورن، ماشا تيفتي غف يزكوان : تارون فل اسن سم ن
ومتين، د وي تيلان، د واين يخدم دي تودرت يس اكن ور ت تتون ناطفارن.
```

In the latter case, a font and direction change were necessary. Otherwise, only the output ΩTP needed to be changed for the three cases. Of course, L^AT_EX macros can be used to encapsulate these changes.

6 Discussion

As this document shows, the om1gc font is now usable for typesetting, using Ω, any language that uses the Latin, Greek, Cyrillic and Tifinagh alphabets, and will soon be ready for the other alphabets in this group.

The om1gc font is not the only multi-script font that has been developed. For example, PLAN 9 [1], Bitstream [2], and Windows [13] all have their own fonts, based on the Unicode encoding.

What does distinguish om1gc from the others is the emphasis that has been placed on typesetting. It is assumed right from the beginning that the set of font positions is much greater than the set of positions in, say, the Unicode definition.

This can best be visualized in Figures 3 and 4, given after the references. For the Greek page, we show the complete set of vowels, with all spirits and breathings, with the macron diacritic, used to denote a long vowel in classical Greek. None of these presentation forms appear in the Unicode specification---hence do not appear in the above fonts---but they are absolutely necessary for typesetting Greek.

Similarly, the Cyrillic page shows all the vowels, with both grave (old style) and acute (new style) accents to place emphasis. Once again, none of these forms are in the Unicode specification, but they are needed for typesetting texts in Russian.

But having these extra forms is of no cost, since they are all virtual characters (hence no extra memory), which can be accessed as needed in a text using the ΩTPs.

We complete the paper with four examples. We begin with three versions (roman, italic, bold) of a Greek text ΠΟΛΛΕΣ ΦΟΡΕΣ ΤΗΝ ΝΥΚΤΑ, by Ἄνδρέας Ἐμπειρικός. We follow with three pages of the om1gc font.

References

1. C. A. Bigelow and K. Holmes. The design of a Unicode font. *Electronic Publishing* 6(3):289--305, 1993.
2. Bitstream Inc. Bitstream Cyberbit. <http://www.bitstream.com/cyberbit.htm>.
3. Y. Haralambous and J. Plaice. First applications of Ω: Greek, Arabic, Khmer, Poetica, ISO 10646/Unicode, etc. *TUGboat*, 15(3):344--352, 1994.
4. Y. Haralambous and J. Plaice. Ω, a T_EX extension including Unicode and featuring lex-like filtering processes. In Wlodek Bzyl and Tomasz Plata-Przechlewski, editors, *Proceedings of the European T_EX Conference*, pages 153--166, Gdansk, Poland, 1994. GUST.
5. Y. Haralambous and J. Plaice. ΩTimes and ΩHelvetica Fonts under Development: Step One. *TUGboat*, 17(3):126--146, 1996.
6. Y. Haralambous and J. Plaice. A font for typesetting large mathematical symbols in Ω. *EuroT_EX'98*, Saint-Malo, France, 29--31 March 1998.
7. J. Plaice and Y. Haralambous. Typesetting French, German and English in Ω. *EuroT_EX'98*, Saint-Malo, France, 29--31 March 1998.
8. Y. Haralambous, J. Plaice, and J. Braams. Never again active characters! Ω-Babel. *TUGboat*, 16(4):418--427, 1995.
9. Y. Haralambous, J. Plaice, and A. Picheral (trans.). Ω, une extension de T_EX incluant UNICODE et des filtres de type Lex. *Cahiers GUTenberg*, 20:55--80, 1995. Translation of refereed conference publication [4].
10. J. Plaice. Language-dependent ligatures. *TUGboat*, 14(3):271--274, 1993.
11. J. Plaice. Progress in the Omega project. *TUGboat*, 15(3):320--324, 1994.
12. J. Plaice and Y. Haralambous. The latest developments in Ω. *TUGboat*, 17(3):181--183, 1996.
13. Microsoft Inc. True Type in Windows 95. <http://www.microsoft.com/truetype/tt/win95tt.htm>.

ΠΟΛΛΕΣ ΦΟΡΕΣ ΤΗΝ ΝΥΚΤΑ

Ἐνδρέας Ἐμπειρικός

Ἵσοι ἀπὸ σᾶς γυρίζετε τὴν νύκτα μέσ' στοὺς δρόμους, ἀμέριμνοι ἢ σκεπτικοί, τὴν ἄνοιξι, κατὰ τὴν ἐποχὴ τοῦ Ἐπιταφίου Θρήνου, ἢ ἐκεῖ κοντὰ στίς ὥρες τίς χαρούμενες πού ὀδηγοῦν στὴν θριαμβευτικὴν τὴν ἄνωσιν πού πάει νὰ γίνῃ Πάσχα, πρὶν ἀκουσθοῦν οἱ ἀναστάσιμες καμπάνες, καί, ἀκόμη περισσότερο, τίς νύκτες τοῦ καλοκαιριοῦ στοὺς δρόμους τοὺς ὄνειρικούς τοῦ σκοτεινοῦ Λονδίνου, στοὺς ἄλλους τοὺς πλατεῖς ἢ τοὺς στενοὺς, πού ἐκτείνονται γύρω ἀπὸ τὸν Μόσχοβα στὴ Μόσχα, ἢ στὰς ὁδοὺς τῆς κάτασπρης Ἀθήνας, σὲ δορυάλωτες στιγμὲς τῆς θλίψεως, ἢ σὲ ἀφρόεσσες στιγμὲς εὐδαιμονίας, ὅταν παράθυρα καὶ ἐξώφυλλα χαίνοῦν διάπλατα ἀνοικτὰ γιὰ νὰ δεχθοῦν δροσιὰ καὶ μῦρα, ὅσοι ἀπὸ σᾶς νύκτωρ γυρίζετε στοὺς δρόμους πανευτυχεῖς πού ἐκσπερματίσατε, ἢ δυστυχεῖς πού κάποια γυναῖκα δὲν ἔστερξε νὰ σᾶς δεχθῆ καὶ δὲν ἐστάθη, λίγο ἂν προσέξετε, θὰ ἀκούσετε πολλά, ὅσα στὴν τύρβη τῆς ἡμέρας δύσκολον εἶναι νὰ ἀκουσθοῦν.

Ἵσοι ἀπὸ σᾶς γυρίζετε τὴν νύκτα μέσ' στοὺς δρόμους, ἀμέριμνοι ἢ σκεπτικοί, τὴν ἄνοιξι, κατὰ τὴν ἐποχὴ τοῦ Ἐπιταφίου Θρήνου, ἢ ἐκεῖ κοντὰ στίς ὥρες τίς χαρούμενες πού ὀδηγοῦν στὴν θριαμβευτικὴν τὴν ἄνωσιν πού πάει νὰ γίνῃ Πάσχα, πρὶν ἀκουσθοῦν οἱ ἀναστάσιμες καμπάνες, καί, ἀκόμη περισσότερο, τίς νύκτες τοῦ καλοκαιριοῦ στοὺς δρόμους τοὺς ὄνειρικούς τοῦ σκοτεινοῦ Λονδίνου, στοὺς ἄλλους τοὺς πλατεῖς ἢ τοὺς στενοὺς, πού ἐκτείνονται γύρω ἀπὸ τὸν Μόσχοβα στὴ Μόσχα, ἢ στὰς ὁδοὺς τῆς κάτασπρης Ἀθήνας, σὲ δορυάλωτες στιγμὲς τῆς θλίψεως, ἢ σὲ ἀφρόεσσες στιγμὲς εὐδαιμονίας, ὅταν παράθυρα καὶ ἐξώφυλλα χαίνοῦν διάπλατα ἀνοικτὰ γιὰ νὰ δεχθοῦν δροσιὰ καὶ μῦρα, ὅσοι ἀπὸ σᾶς νύκτωρ γυρίζετε στοὺς δρόμους πανευτυχεῖς πού ἐκσπερματίσατε, ἢ δυστυχεῖς πού κάποια γυναῖκα δὲν ἔστερξε νὰ σᾶς δεχθῆ καὶ δὲν ἐστάθη, λίγο ἂν προσέξετε, θὰ ἀκούσετε πολλά, ὅσα στὴν τύρβη τῆς ἡμέρας δύσκολον εἶναι νὰ ἀκουσθοῦν.

Ἵσοι ἀπὸ σᾶς γυρίζετε τὴν νύκτα μέσ' στοὺς δρόμους, ἀμέριμνοι ἢ σκεπτικοί, τὴν ἄνοιξι, κατὰ τὴν ἐποχὴ τοῦ Ἐπιταφίου Θρήνου, ἢ ἐκεῖ κοντὰ στίς ὥρες τίς χαρούμενες πού ὀδηγοῦν στὴν θριαμβευτικὴν τὴν ἄνωσιν πού πάει νὰ γίνῃ Πάσχα, πρὶν ἀκουσθοῦν οἱ ἀναστάσιμες καμπάνες, καί, ἀκόμη περισσότερο, τίς νύκτες τοῦ καλοκαιριοῦ στοὺς δρόμους τοὺς ὄνειρικούς τοῦ σκοτεινοῦ Λονδίνου, στοὺς ἄλλους τοὺς πλατεῖς ἢ τοὺς στενοὺς, πού ἐκτείνονται γύρω ἀπὸ τὸν Μόσχοβα στὴ Μόσχα, ἢ στὰς ὁδοὺς τῆς κάτασπρης Ἀθήνας, σὲ δορυάλωτες στιγμὲς τῆς θλίψεως, ἢ σὲ ἀφρόεσσες στιγμὲς εὐδαιμονίας, ὅταν παράθυρα καὶ ἐξώφυλλα χαίνοῦν διάπλατα ἀνοικτὰ γιὰ νὰ δεχθοῦν δροσιὰ καὶ μῦρα, ὅσοι ἀπὸ σᾶς νύκτωρ γυρίζετε στοὺς δρόμους πανευτυχεῖς πού ἐκσπερματίσατε, ἢ δυστυχεῖς πού κάποια γυναῖκα δὲν ἔστερξε νὰ σᾶς δεχθῆ καὶ δὲν ἐστάθη, λίγο ἂν προσέξετε, θὰ ἀκούσετε πολλά, ὅσα στὴν τύρβη τῆς ἡμέρας δύσκολον εἶναι νὰ ἀκουσθοῦν.

Figure 1. Roman, italic and bold versions of a Greek text

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0600	À	á	Â	â	Ã	ã	Ä	ä	Å	å	Ā	ā	Ă	ă	Ą	ą
0610	Ȧ	ȧ	Ȩ	ȩ	Ȫ	ȫ	Ȭ	ȭ	Ȯ	ȯ	Ȱ	ȱ	Ȳ	ȳ	ȴ	ȵ
0620	Ġ	ġ	Ĥ	ĥ	Ħ	ħ	Ĩ	ĥ	Ī	ī	Ĵ	ĵ	Ķ	ķ	Ĭ	ĭ
0630	Ķ	ķ	Ḳ	ḳ	Ḵ	ḵ	Ḷ	ḷ	Ḹ	ḹ	Ḻ	ḻ	Ḽ	ḽ	Ḿ	ḿ
0640	Ṁ	ṁ	Ṃ	ṃ	Ṅ	ṅ	Ṇ	ṇ	Ṋ	ṋ	Ṍ	ṍ	Ṏ	ṏ	Ṑ	ṑ
0650	Ṓ	ṓ	Ṕ	ṕ	Ṗ	ṗ	Ṙ	ṙ	Ṛ	ṛ	Ṝ	ṝ	Ṟ	ṟ	Ṡ	ṡ
0660	Ṣ	ṣ	Ṥ	ṥ	Ṧ	ṧ	Ṩ	ṩ	Ṫ	ṫ	Ṭ	ṭ	Ṯ	ṯ	Ṱ	ṱ
0670	Ṳ	ṳ	Ṵ	ṵ	Ṷ	ṷ	Ṹ	ṹ	Ṻ	ṻ	Ṽ	ṽ	Ṿ	ṿ	Ṱ	ṱ
0680	Ẁ	ẁ	Ẃ	ẃ	Ẅ	ẅ	Ẇ	ẇ	Ẹ	ẹ	Ẹ	ẹ	Ẹ	ẹ	Ẹ	ẹ
0690	Ẹ	ẹ	Ẹ	ẹ	Ẹ	ẹ	Ẹ	ẹ	Ẹ	ẹ	Ẹ	ẹ	Ẹ	ẹ	Ẹ	ẹ
06A0	À	á	Â	â	Ã	ã	Ä	ä	Å	å	Ā	ā	Ă	ă	Ą	ą
06B0	Ȧ	ȧ	Ȩ	ȩ	Ȫ	ȫ	Ȭ	ȭ	Ȯ	ȯ	Ȱ	ȱ	Ȳ	ȳ	ȴ	ȵ
06C0	Ġ	ġ	Ĥ	ĥ	Ħ	ħ	Ĩ	ĥ	Ī	ī	Ĵ	ĵ	Ķ	ķ	Ĭ	ĭ
06D0	Ķ	ķ	Ḳ	ḳ	Ḵ	ḵ	Ḷ	ḷ	Ḹ	ḹ	Ḻ	ḻ	Ḽ	ḽ	Ḿ	ḿ
06E0	Ṁ	ṁ	Ṃ	ṃ	Ṅ	ṅ	Ṇ	ṇ	Ṋ	ṋ	Ṍ	ṍ	Ṏ	ṏ	Ṑ	ṑ
06F0	Ṓ	ṓ	Ṕ	ṕ	Ṗ	ṗ	Ṙ	ṙ	Ṛ	ṛ	Ṝ	ṝ	Ṟ	ṟ	Ṡ	ṡ

Figure 2. Unicode Row 1E: Latin Extended Additional

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0800																
0810	ᾀ	ᾁ	ᾂ	ᾃ	ᾄ	ᾅ	ᾆ	ᾇ	ᾈ	ᾉ	ᾊ	ᾋ	ᾌ	ᾍ		
0820	῀	῁	ῂ	ῃ	ῄ	῅	ῆ	ῇ	Ὲ	Έ	Ὴ	Ή	ῌ	῍		
0830	ἒ	ἓ	ἔ	ἕ	἖	἗	Ἐ	Ἑ	Ἒ	Ἓ	Ἔ	Ἕ	἞	Ἕ		
0840	ἢ	ἣ	ἤ	ἥ	ἦ	ἧ	Ἠ	Ἡ	Ἢ	Ἣ	Ἤ	Ἥ	Ἰ	Ἱ		
0850	ἰ	ἱ	ἲ	ἳ	ἴ	ἵ	ἶ	ἷ	Ἰ	Ἱ	ἰ	ἱ	ἲ	ἳ		
0860	ἰ	ἱ	ἲ	ἳ	ἴ	ἵ	ἶ	ἷ	Ἰ	Ἱ	ἰ	ἱ	ἲ	ἳ		
0870	ὀ	ὁ	ὂ	ὃ	ὄ	ὅ	὆	὇	Ὀ	Ὁ	ὀ	ὁ	ὂ	ὃ		
0880	ὐ	ὑ	ὒ	ὓ	ὔ	ὕ	ὖ	ὗ	὘	Ὑ	ὐ	ὑ	ὒ	ὓ		
0890	ὠ	ὡ	ὢ	ὣ	ὤ	ὥ	ὦ	ὧ	὘	Ὑ	ὠ	ὡ	ὢ	ὣ		
08A0	὘	Ὑ	὚	Ὓ	὜	Ὕ	὞	Ὗ	ὠ	ὡ	὘	Ὑ	὚	Ὓ		
08B0																
08C0																
08D0																
08E0																
08F0																

Figure 3. Greek vowels with macron, for long syllables (not in Unicode)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0A00	Г	Г	Ж	ж	З	з	Қ	қ	Ң	ң	С	с	Х	х	Г	Г
0A10	Д	Д	Л	л	Н	н	Т	т	Ч	ч	Ā	ā	Ō	ō	О	о
0A20	У	у	У	у	Ÿ	ÿ	Ŷ	ŷ	Э	э	Э	э	Ю	ю	Я	я
0A30	І	і	І	і	Ī	ī	Ȳ	ȳ	Á	á	É	é	Ě	ě	É	é
0A40	Й	й	І	і	Ī	ī	Ó	ó	У	у	Ы	ы	Ь	ь	Э	э
0A50	Ю	ю	Я	я	Ђ	ђ	Ў	ў	А	а	Ѓ	ѓ	Ж	ж	Ѕ	ѕ
0A60	Ї	ї	О	о	Ў	ў	Ѓ	ѓ	Ќ	ќ	Ў	ў	Ѕ	ѕ	Æ	æ
0A70	Э	э	Ө	ө	У	у	Ђ	ђ								
0A80	À	Ĥ	À	È	Ї	Й	Ì	Ò	Ù	Ю	О	Ў	Ђ			
0A90	à	ĥ	à	è	й	й	ì	ò	ù	ю	о	ў	ђ			
0AA0	Â	Ĥ	Â	Ê	Ї	Й	Î	Ô	Û	Ю	О	Ў	Ђ			
0AB0	â	ĥ	â	ê	й	й	î	ô	û	ю	о	ў	ђ			
0AC0	Ǻ	Ĥ	Ǻ	Ǽ	Ї	Й	Ǻ	Ǻ	Ǻ	Ю	О	Ǻ	Ǻ			
0AD0	ǻ	ĥ	ǻ	ǽ	й	й	ǻ	ǻ	ǻ	ю	о	ǻ	ǻ			
0AE0	Ǻ	Ĥ	Ǻ	Ǽ	Ї	Й	Ǻ	Ǻ	Ǻ	Ю	О	Ǻ	Ǻ			
0AF0	ǻ	ĥ	ǻ	ǽ	й	й	ǻ	ǻ	ǻ	ю	о	ǻ	ǻ			

Figure 4. Accented Cyrillic characters, for emphasis (not in Unicode)