Εὔτυπον, τεῦχος № 40-41 — Ὀκτώβριος/October 2018

Greek letters for the Fetamont typeface

Linus Romer

Oberseestrasse 7
CH-8716 Schmerikon
Switzerland
Email: `linus dot romer at gmx dot ch`

The glyph range of the Fetamont typeface has been expanded in order to support polytonic Greek. This article describes the problems and solutions that arose during the creation process.

Project initiation

When publishing my last article for *The Eutypon* [1], I was asked by the Greek TEX Friends if I could consider supporting Greek letters in the Fetamont typeface. On the one hand, I was glad to shape the new glyphs, on the other hand, I feared the struggle with additional encodings, as that time the Fetamont typeface only supported the T1 encoding.

Getting METAFONT along with Greek

The Fetamont typeface was written in METAFONT, but the production of the outline font formats relied on METAPOST [2] with the loaded base file `mfplain.mp`. As `mfplain.mp` assumes maximally 256 characters per font, I started to use the LGR encoding, which is a Greek encoding with less than 256 characters (see Figure 1) [3].

My first tests with the LGR-encoded Fetamont showed that the LGR encoding was insufficent for my purposes. One problem was that capital letters like ‘῎Ε’ were decomposed in the accent ‘῎’ and the base letter ‘Ε’. This meant many kerning pairs had to be defined because for letters like ‘῎Α’ the composition has to be much more tight [1]. Therefore, I decided to use Unicode for the whole font. Fortunately, I had already started my own METAPOST base file `mf2outline.mp` [4] for the METAFLOP project [5]. This was now the perfect opportunity to prove this experimental tool.

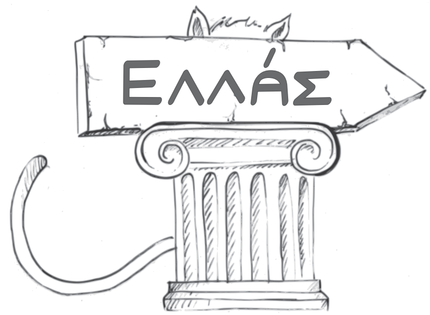

Figure 1: Meta the lioness gets on the way to Greece. Notice that the lambdas are two different randomized variants of the script face of Fetamont.

Using Unicode with METAPOST

Unicode is a bijective map between glyphs and the natural numbers from 0 to $1{,}114{,}111 = 17 \cdot 2^{16} - 1$. Hence, using Unicode with METAPOST means extending the METAPOST base in such way that it supports at least 1,114,112 glyphs at the same time. “At the same time” means that things like glyph composition or kerning tables can occur all over the whole glyph set and not only in a subset.

I began to support an extended version of the `beginchar` macro, which allowed large hexadecimal string codes. For example, the glyph “k” is mapped to the number 75, which is `6B` in hexadecimal representation and the glyph “ϟ” (koppa) is mapped to the hexadecimal number `3DF`:

Glyph	possibilities with `mfplain.mp`	possibilities with `mf2outline.mp`
k	`beginchar("k")`	`beginchar("k")`
	`beginchar(75)`	`beginchar(75)`
		`beginchar("6B")`
		`beginchar("06B")`
		⋮
		`beginchar("00006B")`
ϟ	(no possibility)	`beginchar("3DF")`
		⋮
		`beginchar("0003DF")`

Using six hexadecimal digits makes room for $16^6 = 16\,777\,216$ codes, which is sufficient for covering all the Unicode codes. (Actually, I began with the support of only four hexadecimal digits, because this would be enough for the so-called *Basic Multilingual Plane* of Unicode. But letters like 𐅄 lie outside the *Basic Multilingual Plane*.)

In the script faces of the Fetamont typeface, each glyph has 4 additional randomized variants:

Figure 2: Different faces of Greek Fetamont in use. The boxed capital sigma shows the construction in METAFONT.

Producing randomized variants of the glyphs forced me to store glyph information in a huge array. In order to access the items of the array fast, the hexadecimal string data like `"00005A"` needed to be converted to numerical indices that METAPOST can compute with. Signed numbers in METAFONT and METAPOST must lie between $-2^{12} = -4096$ and 4096 [6]. At first sight, this may lead to the conclusion that an array in METAPOST cannot use more than 4096 indices. However, the following circumstances allow us to increase the number of indices:

- The indices of a METAPOST array do not have to be natural numbers.
- METAPOST uses integer arithmetic based on the unit 2^{-16}.

Therefore, we have about

$$\underbrace{2 \cdot 2^{12}}_{\text{integers}} \cdot 2^{16} = 2^{29}$$

indices, which again are sufficient for covering all the Unicode codes. The whole production of the font outlines then was done by using `mf2outline.mp` with the Python script `mf2outline.py` [4].

Figure 2 shows Greek Fetamont font in use.

References

[1] L. Romer, “The evolution of the Miama typeface.” *The Eutypon*, (2016), no. 36–37, pp. 1–6.

[2] L. Romer, “Fetamont: An extended logo typeface.” *TUGboat*, vol. 35 (2014), no. 1, pp. 17–21.

[3] C. Beccari, “The CB Greek fonts.” *The Eutypon*, (2008), no. 21, pp. 1–13.

[4] L. Romer, MF2OUTLINE, 2018. URL: `https://github.com/linusromer/mf2outline`. (Accessed on Oct. 30, 2018.)

[5] A. Reigel, and M. Müller, METAFLOP, 2012. URL: `https://www.metaflop.com`. (Accessed on Oct. 30, 2018.)

[6] N.H.F. Beebe, “Extending TeX and METAFONT with floating-point arithmetic.” *TUGboat*, vol. 28 (2007), no. 3, pp. 319–328.