Conversion of the Euler Metafonts into the PostScript Type1 font language

Erik-Jan Vens

erikjan@lunatix.icce.rug.nl

1 History of the Euler Metafonts

Hermann Zapf designed the Euler family for the American Mathematical Society for mathematical use, not as a text face, in 1980–1981. This calligraphic family consists of Fraktur, Script, Upright Italic and Math extension. Several people have contributed to the Metafont programming, among them Don Knuth, John Hobby, David Siegel, Dan Mills and Carol Twombly. There is a description of Euler in TUGboat 10.1.

For this article I will concentrate on Euler Roman Medium. The chosen example is the capital A from this font. Figure 1 shows what the character looks like.

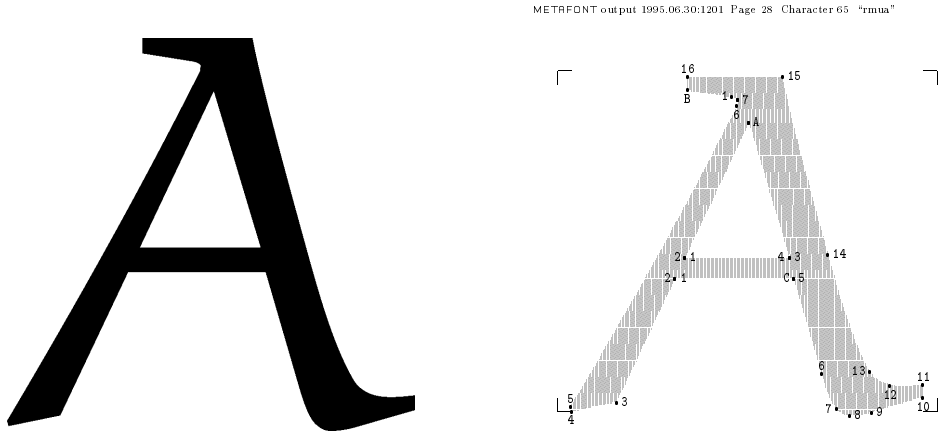

Figure 1: Left: The character capital A in PK format; Right: The same character in MF format

2 Different approaches to computerized curve description

There are several approaches possible to describe a curve. From the days of the Renaissance onwards people have tried to do so without good results. The first one to really succeed in doing so was Pierre Etienne Bézier. He applied the theories of Sergeĭ Bernshteĭn to his specific area, computer-aided design.

2.1 The richness of the Metafont language

METAFONT uses a method whereby curves can be drawn either by filling specified outlines or by simulating the movement of a pen. We can pick up a pen with a specified form of the head and start drawing, and we can use an eraser and remove ink from the paper. We can clip a form or rotate it. We can scale it or clip it out and paste it one or more times.

I will not go into further detail of METAFONT. It is sufficient to say that METAFONT is a very rich and powerful language.

2.2 The limitations of the Type1 language

The Postscript Type1 language is an enhanced subset of the Postscript language. It can be used to specify a path and fill it. This is also a part of the METAFONT language, so one can easily convert from Type1 to METAFONT, but the other way around provides us with a lot of difficulties. Again, I will not go into further detail of a language. Those persons who are interested in what comprises Type1 are encouraged to buy the so-called 'Black Book', the Adobe Type1 Font Specification.

3 Transformation of Metafont into Type1

One can discuss the necessity for the TEX community to use Postscript fonts. I feel that there are far too little good-quality fonts around in the METAFONT format, so one almost must use Postscript fonts. This doesn't mean the technical capabilities of METAFONT are under discussion.

The Postscript world is far larger than the TEX world. It would be nice to be able to donate what 'we' have to this Postscript world. This has to be done in Postscript form, hence the necessity for transformation.

There are several routes one can take to convert from METAFONT into Type1. I have outlined three of them. None of them is perfect in the sense that it can be fully automated. All of them need handtuning. This makes it a very tedious job, almost like starting from scratch. The first steps for all of them can be measured in minutes. The fine-tuning will take days, if not weeks or months per font.

I have used commercial tools in almost all of the stages. I don't know of any public domain or shareware programs that can do all of the tricks needed to bring a font up.

3.1 Using the GNU fontutils

Karl Berry and Kathy Hargreaves have written a suite of utilities to manipulate bitmapped fonts and trace them into outline fonts. All of their data is created by scanning fontbooks. But one can also use METAFONT to render a bitmap font in the form of a `GF` file and use this as starting point for tracing.

I created a very huge `GF` file by using

```
mf "\mode:=proof; mag:=3.171875; input eurm10"
```

I have not tried out whether smaller input files also create good results. This magnification was simply the largest before METAFONT issued a `value too large` error-message. Since I could create such large file, I didn't bother experimenting with the size. Then I used their program limn to create a `bzr` file:

```
limn -verbose -corner-surround=24 -filter-surround=24 \
     -filter-alternative-surround=12 -subdivide-surround=24 \
     -tangent-surround=24 -reparameterize-threshold=30 \
     eurm10.8252gf
```

A `bzr` file is binary representation of curve information. It can be converted to several formats, amongst them METAFONT. But I converted it to Type1 with:

```
bzrto -pstype1 eurm10
```

This does not create a 'real' Type1 font, but a Ghostscript version of the same, which will only run under a Ghostscript version of the Postscript interpreter. I therefore had to convert this to a normal Type1 font with two little programs which I hacked together from two utilities I already had:

```
gsf2ps < eurm.gsf > eurm.chr
chr2ps eurm.chr eurm.aapje
```

Now I could import the `Charstrings` section from the `eurm.aapje` file into a template file which I had created from a valid Type1 font.

I imported the thus created font into Fontmonger and edited all the characters. The amount of editing depends on the parameters as chosen for `limn`. It is very important to chose these as good as possible. The parameters I have shown in this example might not be the best around, but these can only be found by fiddling with them.

To show how much editing has to be done, look at Figure 2, which shows the character capital A before and after editing.

3.2 Using a commercial program

Fontographer or Fontmonger

In this case I used METAFONT to render the bitmaps which I converted to a picture format with Ghostscript. I made a small TEX file which consisted of a loop which put

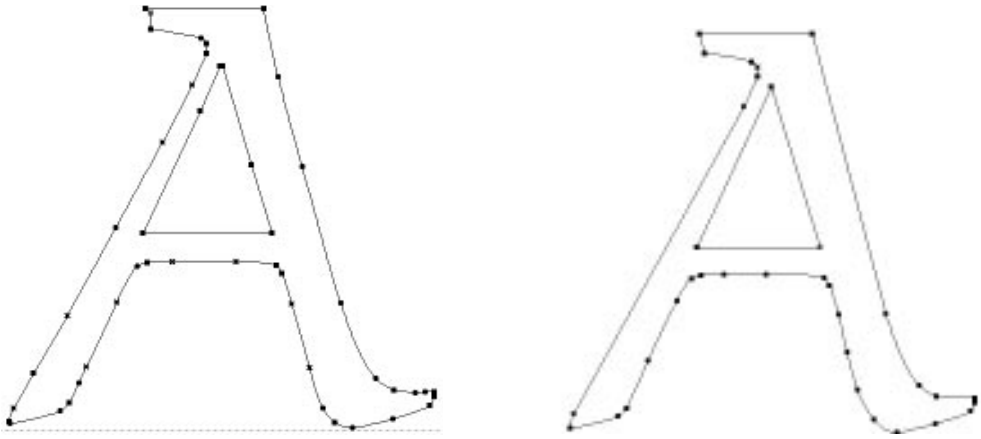

Figure 2: Left: The character before editing; Right: The character after some editing

a large character on each page, and then used dvips to create a Postscript file, and Ghostscript to render a picture for each page of the input file.

Then I started up Fontographer and for each character I imported a background which I then traced using the Fontographer trace function. This is a very fast process which gives fairly good results.

The real problem lies with the hinting. Contrary to the approach of METAFONT where each instance of a font at a specific pointsize can be rendered at it's best rasterizing for every conceivable outputdevice, Type1 uses a system called hints to specify how the rasterizer should decide which pixels to turn on or off.

One can let Fontographer try to decide which hints are general hints and then see for oneself whether it has chosen right hints.

The result of the capital A is shown in Figure 3.

CorelTrace

Figure 4 shows what the output with CorelTrace looks like.

This is definitely not good. There are a lot of parameters one can adjust, but I haven't found any better settings. So, using CorelTrace is not a real option.

3.3 Using MetaPost

When one has a METAFONT source, it might be a good idea to directly convert it to Postscript using MetaPost. This program is METAFONT with some changes, and it can read almost all of the standard METAFONT files. There are a few exceptions, notably the cutoff function is not implemented. But the overall results are good and very fast.

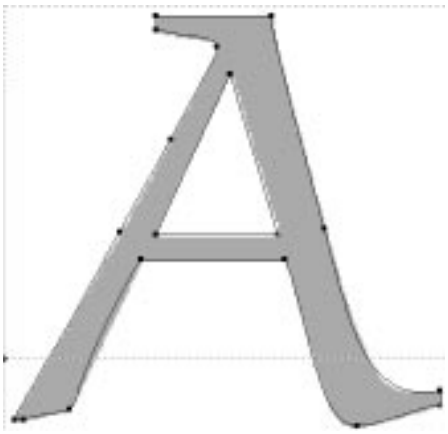

Figure 3: Capital A as traced by Fontographer

Just as with the other options, the main problem lies with the hinting. One has still to do the hinting after the conversion is done. I have used a commercial program to import the Postscript files for each character. The general result is shown in Figure 5.

4 Conclusions and general remarks

To quote from Don Knuth's METAFONT book: 'We should realize before we begin that it would be a mistake to set our hopes too high. Mechanically generated letters that are untouched by human hands and unseen by human eyes can never be expected to compete with alphabets that are carefully crafted to look best on a particular device. There's no substitute for actually looking at the letters and changing their pixels until the result looks right. Therefore our goal should not be to make hand-tuning obsolete; it should rather be to make hand-tuning tolerable. Let us try to create meta-designs so that we would never want to change more than a few pixels per character, say half a dozen, regardless of the resolution. At low resolutions, six pixels will of course be a significant percentage of the whole, and at higher resolutions six well-considered pixel changes can still lead to worthwhile improvements. The point is that of our design comes close enough, a person with a good bitmap-editing program will be able to optimize an entire font in less than an hour. This is an obtainable goal, if rounding is done judiciously.'

The other realization one has to make is whether it is worthwhile to spend so much time into converting a font which is only relevant for mathematicians, who are usually already using TEX, and hence do not need a Postscript font. On the other hand, the work of a genius such as Mr. Zapf should never be underestimated, and therefore the more widespread it's use, the better.

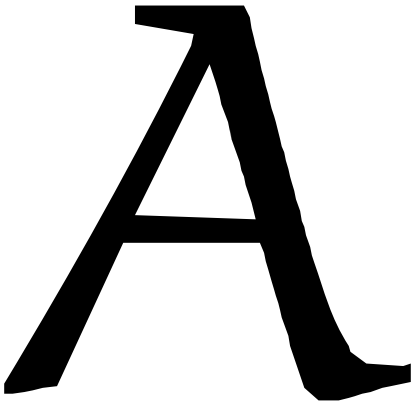

Figure 4: Capital A as traced by CorelTrace

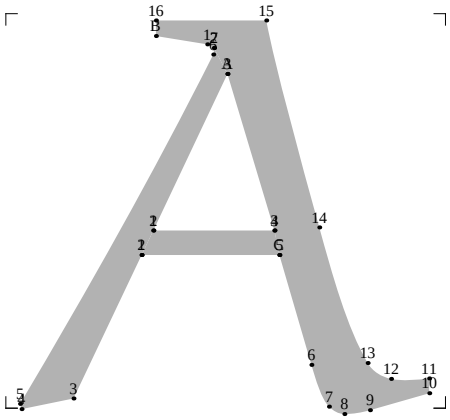

Figure 5: Capital A as created by MetaPost