Making Type 1 and OpenType fonts with MetaType1 and FontForge

Karel Píška

Institute of Physics, Academy of Sciences
Prague, Czech Republic

24 August 2008

2nd ConTEXt Meeting

Bohinj, Slovenija

Contents

- Type 1 v.s. OpenType
- OpenType fonts today
  - TeX text OpenType fonts
  - OpenType math fonts today
- Stage 1: Font creating with MetaType1
  - Examples with Latin Modern
- Stage 2: From Type 1 to OpenType
  - Construction of OpenType
- Conclusion and suggestions
- Conclusion and suggestions
  - TeX Gyre fonts and math
  - OpenType tables
  - Last comments

Type 1 v.s. OpenType

(probably everybody knows)
Limitations in Type 1

- max.number of encoded glyphs – 256
- we need many encoding files to cover various languages and their encodings
  (9 or more in today's Latin Modern and TEX Gyre)
- metrics data (also ligatures, kernings, . . .) in additional separated files (× the number of encodings)

OpenType fonts

- can cover all characters together with metrics and "advances typographic facilities"
- are available for XeTEX, LuaTEX
- allow to unify access to glyphs, hyphenation patterns, . . .

OpenType fonts today

TEX text OpenType fonts

- Latin Modern (LMRoman10-Regular)
  old style digits present
- TEX Gyre (TeXGyreTermes-Regular)
  old style digits, small caps
- Antykwa Torunska
- Iwona
- Kurier

and, maybe, other

OpenType fonts today

OpenType math fonts

- Cambria Math [MicroSoft]
  - old style digits
  - math symbols
  - letters: regular, bold, (math) italic, small caps, subscript, superscript, script-script, etc.
  - see (MS specification)
- Minion Math [Jonannes Küster]
- Asana Math [Apostolos Syropoulos] (glyph list)
  special optical sizes for scripts and scriptscrips are absent (?)
- STIX not available after beta-testing (?)
- other OpenType math ?

Stage 1: Font creating with MetaType1

The fonts can be generated with the MetaType1 package [authors B. Jackowski, J. Nowacki, P. Strzelczyk].
The raw information about fonts and their glyphs is described in the METAPOST source files; additional macros are defined in MetaType1 macro extension or may be appended by user.

Stage 1: Font creating with MetaType1

Examples with Latin Modern

```
% LATIN MODERN font: a driver file for lmr10
input fontbase;
vardef cm_pal = "cmr10" enddef;
input comm_mac;      % common defs, CM params
input comm_mph;      % common header
input lmr10.mpm;     % metric data
input lmr10.mph;     % PS-oriented header
beginfont
input lmr10.mpg;     % ‘‘frozen’’ glyphs
input comm_mpg;      % common glyphs (mainly diacritics)
if known generating: % optimize proofing time
 input lmr10.mpl;    % ligatures and kerns
fi
endfont
%%%% EOF
```

Stage 1: Font creating with MetaType1

Examples with Latin Modern

```
beginglyph(_a);
 save p; path p[];

 z0 0=(493,89);
 z0 1=(493,145);
 z0 2=(468,145);
 z0 3=(468,89); z0 3a=(468,31); z0 4b=(443,25);
 z0 4=(432,25); z0 4a=(399,25); z0 5b=(395,70);
 z0 5=(395,75);
 z0 6=(395,275); z0 6a=(395,317); z0 7b=(395,356);
 z0 7=(359,393); z0 7a=(320,432); z0 8b=(270,448);
 z0 8=(222,448); z0 8a=(140,448); z0 9b=(71,401);
 z0 9=(71,335); z0 9a=(71,305); z0 10b=(91,288);
 z0 10=(117,288); z0 10a=(145,288); z0 11b=(163,308);
```

Stage 1: Font creating with MetaType1

Examples with Latin Modern (cont.)

```
z0 11=(163,334); z0 11a=(163,346); z0 12b=(158,379);
z0 12=(112,380); z0 12a=(139,415); z0 13b=(188,426);
z0 13=(220,426); z0 13a=(269,426); z0 14b=(326,387);
z0 14=(326,298);
z0 15=(326,261); z0 15a=(275,258); z0 16b=(205,255);
z0 16=(142,225); z0 16a=(67,191); z0 17b=(42,139);
z0 17=(42,95); z0 17a=(42,14); z0 18b=(139,-11);
z0 18=(202,-11); z0 18a=(268,-11); z0 19b=(314,29);
z0 19=(333,76); z0 19a=(337,36); z0 20b=(364,-6);
z0 20=(411,-6); z0 20a=(432,-6); z0 21b=(493,8);
z0 21=(493,89);
p0=compose_path.z0(21);
```

Stage 1: Font creating with MetaType1

Examples with Latin Modern (cont.)

```
z1 0=(326,140); z1 0a=(326,45); z1 1b=(254,11);
z1 1=(209,11); z1 1a=(160,11); z1 2b=(119,46);
z1 2=(119,96); z1 2a=(119,151); z1 3b=(161,234);
z1 3=(326,240);
p1=compose_path.z1(3);

if turningnumber p0>0: Fill else: unFill fi \\ p0;
if turningnumber p1>0: Fill else: unFill fi \\ p1;

fix_hstem(21)(p0,p1);
fix_hstem(31)(p0,p1);
fix_hstem(22)(p0,p1);
set_hstem (288,378);
fix_vstem(77)(p0,p1);
fix_vstem(69)(p0,p1);
fix_vstem(25)(p0,p1);
standard_exact_hsbw("a");
```

Stage 1: Font creating with MetaType1

In a similar way all Type 1 fonts and all their glyphs are described. We can combine two approaches

- take existing Type 1 (e.g. LM or AMS fonts) and create additional fonts for absent fonts (styles) and glyphs
- collect all glyphs (already present or designed as new) together into one (intermediate, working) Type 1 font; then we have to distinguish instances of one font representing various styles, sizes, weights, etc.
- create new fonts for missing fonts and glyphs

Stage 2: From Type 1 to OpenType

In OpenType the glyphs are accessed (addressed) by their Unicode numbers; glyphs names are usually missing in a font (a large font like math)

In FontForge (scripting language) we copy the information about glyphs from intermediate Type 1 (or from existing fonts, like LM)

```
Open($3.pfb); # open Type 1
Select("a"); # "regular a"
Copy();Close();
Open($1.sfd); # working internal file
Select("a"); #
Paste();Close();
```

Stage 2: From Type 1 to OpenType

Construction of OpenType

Bold style “a” (“a.bf” artificial internal working name)

```
Open($3.pfb); # open Type 1
Select("a.bf"); # "bold a"
Copy();Close();
Open($1.sfd); # working internal file
Select("u1D44E"); # Unicode number
Paste();Close();
```

“a”	“a”	regular “a”
“a.bf”	“u1D41A”	bold “a”
“a.mi”	“u1D44E”	(math) italic “a”
“a.mib”	“u1D482”	(math) italic bold “a”

etc.

Conclusion and suggestions

Conclusion and suggestions

LM fonts and OpenType math

I think we could compile the glyphs (most of them) form Latin Modern sources and AMS Type 1 fonts (mathematical symbols)

TeXGyreTermes-Regular (Termes-Regular)
TeXGyrePagella-Regular (Pagella-Regular)
old style digits, small caps
cover all glyphs with consistency would be difficult,
i.e. to produce all styles, scripts, script-scripts in proper optical sizes; cover all math symbols.
With MetaType1 and/or FontForge we could to create new glyphs, execute (semi-)automatically various transformations to produce slanted, bold forms or various weights— "re-introduction of MM (multi master)" in a more general way — to write MetaType1 programs for more complex transformations. All it would be possible but I think work and time consuming.

Conclusion and suggestions

OpenType tables

Next step would be define feature and lookups (GPOS, GSUB) and MATH tables—I will not present that in my talk. It will be possible to try using FontForge or Adobe Development Kit.

Conclusion and suggestions

Last comments

We can continue our discussion started in Bachotek this spring.