Using Adobe Type 1 Multiple Master Fonts with TEX

Michel Goossens
CN Division, CERN
CH-1211 Geneva 23
Switzerland
Email: m.goossens@cern.ch

Sebastian Rahtz
Elsevier Science Ltd
The Boulevard, Langford Lane
Kidlington, Oxford OX5 1GB
UK
Email: s.rahtz@elsevier.co.uk

Robin Fairbairns
University of Cambridge Computer Laboratory
Pembroke St, Cambridge CB2 3QG
UK
Email: rf@cl.cam.ac.uk

Abstract

Adobe's Multiple Master font format has some of the properties that METAFONT pioneered to express many fonts of the same family from the same (set of) sources. The advent of multiple master fonts could offer a significantly better choice of fonts to users of TEX; however, there are problems integrating them with TEX, and the paper presents a first solution to those problems.

The paper is derived (by Robin Fairbairns) from an article written by Michel Goossens and Sebastian Rahtz for the UKTUG magazine Baskerville volume 5, number 3. As a demonstration of the effectiveness of the techniques described, it is being typeset using Adobe Minion and Myriad multiple master fonts.

Introduction

The multiple master font format is an extension of the Type 1 font format, which allows the generation of a wide variety of typeface styles from a single font program. This capability allows users and applications control over the typographic parameters of fonts used in their documents, in a manner reminiscent of Knuth's ground-breaking METAFONT. This article describes the multiple master system in some detail, and describes the procedures needed to make instances, and create the appropriate font metrics for use with TEX.

Multiple Master overview

A multiple master font program contains two or more outline typefaces called *master designs*, which describe one or more *design axes*. The master designs that constitute a design axis represent a dynamic range of one typographic parameter, such as the weight or width. This range of styles is defined in a multiple master font program by specifying one master design to represent each end of an axis, such as a *light* and *extra-bold* weight, as well as any *intermediate master designs* that are required. The maximum number of master designs allowed is sixteen.

A *font instance* consists of a font dictionary derived from the multiple master font program (or from another font instance). It contains a `WeightVector` array with k values that sum to 1.0 and which determine the relative contributions of each master design to the resulting interpolated design.

All derived font instances share the `CharStrings` dictionary and `Subrs` array of the main multiple master font program, making it relatively economical to generate a variety of font instances. Multiple master fonts can be made compatible with the installed base of PostScript language interpreters by including several PostScript language procedures and a new set of `OtherSubrs` routines in the font program. The procedures include the new `makeblendedfont` operator, the interpolation procedure `$Blend` and a new definition of the `findfont` operator.

Multiple Master Design Space It is possible to think of the master designs as being arranged in a 1, 2, 3, or 4 dimensional space with various font instances corresponding to different locations in that space. The entries in the `FontInfo` dictionary specify what this space is and where the master designs are located in it. This information is necessary for interactive programs that allow users to create new font instances, and should be included in the font's Adobe Multiple Font Metrics (`AMFM`) file.

Figure 1 illustrates an example of the design space of a three axis multiple master font. In this example, the axes are *weight*, *width*, and *optical* size. It is recommended that a font program be organized to have the lightest weight, narrowest width, and smallest design size mapped to the origin of the blend space.

Multiple master coordinates are of two types: those which represent the design space and those which represent the blend space. *Design coordinates* are integers whose range for a particular typeface is chosen by the designer. They are used in font names and in the user interface for software which creates and manipulates multiple master font programs. The theoretical range for a weight or width axis is from 1 to 999 design units; however a typical typeface, with styles ranging from light to black, might have a dynamic range of from 200 (for light) to 800 units (for black).

Another type of optional axis would be for optical size, in which the character design changes with the point size. The design coordinates for the optical size axis might have a dynamic range of from 6- to 72-point, which represents the practical extremes of sizes for typefaces designed for publishing purposes.

Blend coordinates are normalized values, in the range of 0 to 1, which correspond to the minimum and maximum design space coordinates. They are used by the Type 1 rasterizer because they are more convenient for mathematical manipulations. The linear space of blend coordinates is related to the (potentially) non-linear space of the design coordinates by the `BlendDesignMap` entry in the font dictionary.

A four axis design might also be considered; an example of a fourth axis would be having an axis for a typographic style (serif/sans serif) or contrast (high/low: the ratio of thick to thin stem widths). If four axes are defined, sixteen master designs are required. Also, since sixteen is the maximum number of designs allowed, there can be no intermediate designs with four axes.

Multiple Master Font Programs

Multiple master typefaces may contain from two to sixteen master designs, organized as having from one to four design axes. Since the maximum number of master designs allowed in a multiple master font is 16, the number x of intermediate masters is subject to the restriction $2^n + x \leq 16$, where n is the number of design axes.

The values used for calculating the weighted average are stored in the font dictionary in an array named `WeightVector`. The multiple master font program, as shipped by the font vendor, can have a default setting for the `WeightVector`; it is recommended that it is set so the default font instance will be the normal roman design for that typeface.

Multiple Master Keywords `BlendAxisTypes` is a (required) array of n PostScript language strings where n is the dimension of the design space and hence the number of axes. Each string specifies the corresponding axis type. In the case of the Minion 3-axis example, this value would be:

```
/BlendAxisTypes [/Weight /Width /OpticalSize]
```

`BlendDesignPositions` is a (required) array of k arrays giving the locations of the k master designs in the design space. Each location subarray has n numbers giving the location of the design in the n dimensions of the blend space, with a minimum value of zero and a maximum value of one. Table 1 with eight master designs is based on the example shown in Figure 1. It corresponds to the blend space of a 3-axis multiple master font like Minion.

For the Minion MM font, the `BlendDesignPositions` array becomes:

```
/BlendDesignPositions
        [[0 0 0][1 0 0][0 1 0][1 1 0]
        [0 0 1][1 0 1][0 1 1][1 1 1]] def
```

`BlendDesignMap` is a required entry consisting of an array of n arrays where n is the dimension of the design space. Each array contains m subarrays that describe the mapping of design coordinates into normalized coordinates for that design axis. The minimum value allowed for m is two, and the maximum is twelve. The order of the subarrays corresponds to the order of design axes in `BlendAxisTypes`. In the case of the Minion font this array is three dimensional ($n = 3$) and has the following form:

```
/BlendDesignMap [
 [[345 0] [620 1]]  [[450 0] [600 1]]
 [[6 0] [8 0.35] [11 0.5] [18 0.75] [72 1]] ]
```

The first number in an individual subarray is in design coordinates with a minimum value of 1 and a maximum value of 999. The second number in the subarray is in normalized coordinates, that is, in the range of 0 to 1. In the above example, the weight ranges from 345 to 620, while the width ranges from 450 to 600 in design space. The third axis, optical size, ranges from 6 to 72 (corresponding to the point sizes for which the typeface can be adjusted for optimal legibility).

The `makeblendedfont` Operator

blendedfontdict weightvector **makeblendedfont** *blendedfontdict*

This operator creates a font dictionary with blended entries. The *blendedfontdict* argument is a font dictionary of an existing multiple master font; it can be from either

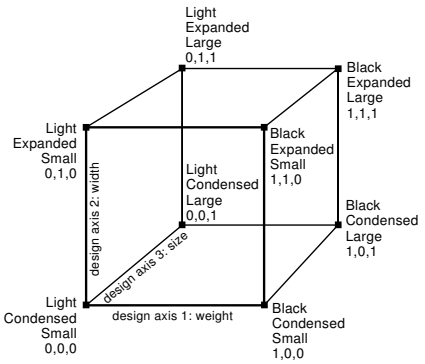

Figure 1: Multiple master typeface blend space arrangement

Design label	*Blend space coordinates*
design 1: light condensed small	0 0 0
design 2: black condensed small	1 0 0
design 3: light expanded small	0 1 0
design 4: black expanded small	1 1 0
design 5: light condensed large	0 0 1
design 6: black condensed large	1 0 1
design 7: light expanded large	0 1 1
design 8: black expanded large	1 1 1

Table 1: Design labels and blend space values for the Minion 3-axis multiple master font

the original multiple master font itself, or from an interpolated font instance since any `Blend` dictionary contains all elements needed to derive additional font instances.

The *weightvector* argument is an array of numbers summing to 1.0 to be used as the weights for creating the new font instance. The value of `WeightVector` in *blendedfontdict* is set to the values in the array weightvector. Blended values are calculated for entries in the `Private` and `FontInfo` dictionaries. The result is a font dictionary that can be used as an argument to `definefont`. The resulting dictionary and its contents are still read-write, so the caller of `makeblendedfont` can make further modifications if necessary.

The `Blend` dictionary data structures provide the information needed by the `makeblendedfont` operator, without needing to have the specific list of entries to be blended built into the procedure. This allows a single copy of the procedure to be used even if the set of entries to be blended varies in future fonts.

Multiple Master findfont Procedure Multiple master font programs from Adobe include a procedure which redefines the `findfont` operator in `systemdict`. This is necessary because of the need to generate font instances on-the-fly to satisfy multiple master font references in a PostScript language document. The procedure creates all necessary font instances before calling the standard `findfont` procedure.

Two procedures, `NormalizeDesignVector` and `ConvertDesignVector`, which are referenced in `findfont`, must be configured for the number of axes and master designs in the font program in which they are used. The `NormalizeDesignVector` procedure must calculate the normalized equivalent of the design coordinates in the FontName, using the values in the `BlendDesignMap` array. These normalized coordinates must be left on the stack for the `ConvertDesignVector` procedure. This procedure should take the normalized coordinates, generate `WeightVector` values, and leave them on the stack for the `makeblendfont` operator.

Using Multiple master fonts with TEX

Multiple master fonts come with a set of multiple master AFM files, which are called "AMFM" (Adobe Master Font Metrics) files. This file contains information about the number of master designs, the number of axes, the `BlendDesignPositions` and `BlendDesignMap` arrays, as well as the names, and `weightvector` for the master designs, from which all font instances are derived.

To get the actual metric information for the characters in a font instance, one has to combine the metric information of the master designs (eight, in the case of Minion). To do this one needs to calculate the `weightvector` for the given instance. Starting from design-coordinate space one can use the `NormalizeDesignVector` operator to transform to the normalized coordinate space, and from there with the `ConvertDesignVector` operator one obtains the weightvector. These two operators are particular to a font (since they depend on the master designs), and are present in the multiple master font dictionary. One can decode the PostScript code for calculating the weightvector and translate it into another computer language, and then use the procedure to combine the values in the AFM files for the master designs to calculate the values needed for the font instance. For instance, in the case of the MinionMM font, the PostScript code defines the eight components of the weightvector as follows:

$$\begin{array}{lcllcl} w_1 &=& xyz & w_2 &=& (1-x)yz \\ w_3 &=& x(1-y)z & w_4 &=& (1-x)(1-y)z \\ w_5 &=& xy(1-z) & w_6 &=& (1-x)y(1-z) \\ w_7 &=& x(1-y)(1-z) & w_8 &=& 1-\sum_{n=1}^{7} w_n \end{array}$$

where x is the normalized weight, y the normalized width, and z the normalized optical size.

These eight numbers w_i allow the calculation of all needed parameters in an AFM file for a font instance. One

```
/y 140 def
300 100 700 % outer loop on width>>>>>>>>>>>>>+
    { /x 25 def       % reset x              v
      /y y 25 sub def % calculate y          v
      /Wi exch def    % width from for loop<<+
      220 100 820 % inner loop on weight>>>>>>>>>>>+
         { x y moveto      % go to new coordinate  v
           /We exch def    % weight from for loop<<+
           /MM /MyriadMM findfont dup begin [
             We Wi NormalizeDesignVector ConvertDesignVector
             ] end makeblendedfont definefont 20 scalefont setfont
             (Hxkp) show
           /x x 55 add def % recalculate x
         } for
    } for
```

Figure 2: PostScript code for generating a a two-dimensional matrix showing instances of the multiple master font Myriad

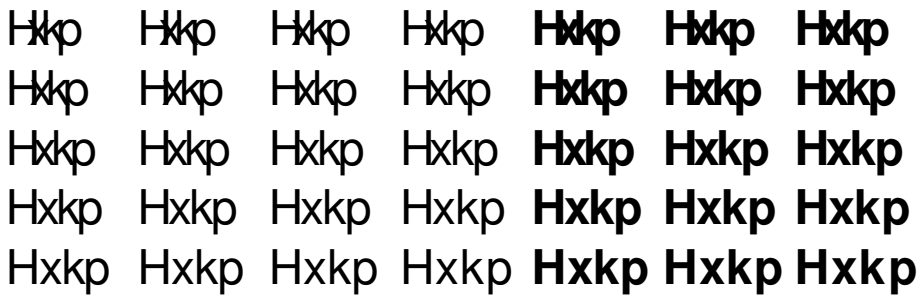

Figure 3: Two dimensional grid showing various instances of the two axes multiple master Myriad sans serif font.

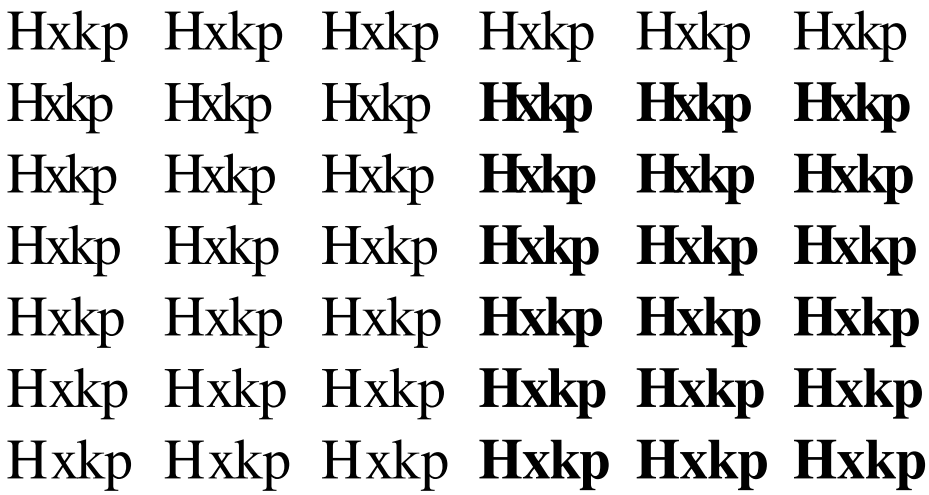

Figure 4: The three axes multiple master Minion serif font. The top line shows various optical sizes (6pt, 8pt, 11pt, 18pt, 40pt, and 72pt) normalized to 20pt. The bottom matrix shows various weights (increasing from left to right) and widths (increasing from top to bottom).

reads each parameter value in turn in the eight master design AFM files, applies the relevant weight, and the weighted sum thus obtained is the desired interpolated value of the given parameter for the font instance.

Myriad is a sans serif companion font to Minion. It has two design axes and four master designs. The weights for deriving font-instance parameters in normalized coordinate space in function of the four master designs are given by:

$$w_1 = (1-x)(1-y) \quad w_2 = (1-x)y$$
$$w_3 = x(1-y) \quad w_4 = xy$$

where x is the normalized weight and y the normalized width. The corresponding mapping parameters between design space and normalized coordinates are:

```
BlendDesignPositions [ [0 0] [1 0] [0 1] [1 1] ]
BlendDesignMap [[[215 0][830 1]][[300 0][700 1]]]
BlendAxisTypes [/Weight /Width ]
```

Now one can extract any of the boundingbox and kern entries for a given font instance by getting the element in question from the eight (or four, in the case of Myriad) master files and calculating the interpolated value. To make matters simpler an explicit example will be given for the Myriad font, since it involves only four numbers in each case. Figure 5 shows some parts of the four master-design AFM files

When the instance AFM file has been created, a suitable metric for TEX can be built with `afm2tfm` or the *fontinst* package.

In practice

We have instantiated the ideas outlined above by developing Unix shell scripts, and adapting an AFM-parsing program distributed by Adobe. The main script takes the following actions:

1. create a small PostScript file to invoke multiple master operators with values passed to the script;
2. run GhostScript on this file to derive normalized weights, and write them to a temporary file; note that this must be version 3.33 or later of Aladdin GhostScript — earlier versions of the program did not have the code to realize multiple master fonts;
3. run our "mmafm" program to read master AFM files, write a new instance AFM file, and create a TEX metric (our initial setup uses `afm2tfm` to create 8r base-encoded metrics, and EC-encoded virtual fonts for actual use);
4. write a `dvips` map entry and header file to tell the driver about the new font.

Thus a call to our scripts consists of the parameters

```
MinionMM  zmnl8ac6 360 460 6
```

This creates a metric file called `zmnl8ac6`, using Karl Berry's scheme to name "Minion, light weight, 8a-encoded, condensed, at 6pt design size". The entry in the map file reads

```
zmnl8rc6 zmnl8ac6 "TeXBase1Encoding \
 ReEncodeFont" <8r.enc <MinionMM.pfb \
  <zmnl8ac6.pro
```

where the prologue file `zmnl8ac6.pro` contains instructions to the PostScript interpreter as to how the given font instance should be generated from the multiple master font code in `MinionMM.pfb`: `zmnl8ac6.pro` contains the code:

```
/zmnl8ac6 /MinionMM findfont
dup begin [
  360 460 6   NormalizeDesignVector
  ConvertDesignVector
  ] end makeblendedfont definefont
```

Note the presence of the `NormalizeDesignVector`, `ConvertDesignVector` and `makeblendfont` PostScript operators described earlier.

In addition, we hand-wrote "fd" files to tell LATEX how to match up the various weight and width instances we created to its notions of series and shape. The only complication here was that the Minion font has an optical size axis, and we built four instances which we wanted LATEX to use at different user sizes:

```
\DeclareFontShape{T1}{zmn}{lc}{n}{%
        <-7>zmnl8tc6 %
        <7-10>zmnl8tc8  %
        <10-15>zmnl8tc11 %
        <15->zmnl8tc18}
{}
```

The effect of the optical sizes is demonstrated by Figure 6 which shows the 6pt and 18pt instances scaled to the same size. The differences in design are as apparent as the corresponding examples from Computer Modern would be.

The tools we developed served to test the ideas, and build a set of metrics; they are available from us on request, but users should beware that they are neither intuitive in use, nor necessarily robust. It is to be hoped that a more functional, portable, solution will be developed in time. The keen TEXie may be interested in developing a `MakeTeXTFM` script for Unix `web2c` systems to apply the programs on the fly from within TEX.

```
FontName MyriadMM-LightCn
FamilyName Myriad MM
Weight Light
ItalicAngle 0
IsFixedPitch false
FontBBox -52 -250 970 818
...
StartKernPairs 974
KPX A z 10
KPX A y -31
KPX A x 4
KPX A w -36
KPX A v -42
KPX A u -9
KPX A t -17
KPX A s 0
KPX A r -4
KPX A quoteright -90
KPX A quotedblright -90
KPX A q -9
KPX A p -4
KPX A o -12
  ...
EndKernPairs
```

```
FontName MyriadMM-BlackCn
FamilyName Myriad MM
Weight Black
ItalicAngle 0
IsFixedPitch false
FontBBox -64 -250 970 843
...
StartKernPairs 974
KPX A z 10
KPX A y -10
KPX A x 0
KPX A w -10
KPX A v -10
KPX A u 0
KPX A t 0
KPX A s 10
KPX A r 0
KPX A quoteright -20
KPX A quotedblright -20
KPX A q 0
KPX A p 0
KPX A o 0
  ...
EndKernPairs
```

```
FontName MyriadMM-LightSemiEx
FamilyName Myriad MM
Weight Light
ItalicAngle 0
IsFixedPitch false
FontBBox -58 -250 1100 825
...
StartKernPairs 974
KPX A z 25
KPX A y -10
KPX A x 0
KPX A w -10
KPX A v -25
KPX A u -10
KPX A t 0
KPX A s -10
KPX A r 0
KPX A quoteright -30
KPX A quotedblright -30
KPX A q -10
KPX A p 0
KPX A o -10
  ...
EndKernPairs
```

```
FontName MyriadMM-BlackSemiEx
FamilyName Myriad MM
Weight Black
ItalicAngle 0
IsFixedPitch false
FontBBox -48 -250 1432 867
...
StartKernPairs 974
KPX A z 7
KPX A y -44
KPX A x -6
KPX A w -47
KPX A v -62
KPX A u -22
KPX A t -32
KPX A s -6
KPX A r -10
KPX A quoteright -90
KPX A quotedblright -90
KPX A q -18
KPX A p -10
KPX A o -18
  ...
EndKernPairs
```

Figure 5: The four AFM files for the Myriad master designs

Figure 6: Minion instances from opposite ends of the optical size axis set at the same size (exaggerated)