Graphics and TEX — A Reappraisal of METAFONT/MetaPost/PostScript

Kees van der Laan

Abstract

It is all about the author's first steps in METAFONT, to create graphics for inclusion in TEX documents, with a wink to MetaPost and ... PostScript. The graphics comprises graphs of math functions, 2-D pictures and 2.5-D images of 3-D objects via projection techniques; 4-D for varying viewing angles is touched upon. Some highlights on macro writing in METAFONT have been selected, and the appendix provides a list of examples I have collected.

Introduction

The handling of graphics in TEX scripts has a long history.[1] There are three approaches from the document preparation point of view:

- TEX alone
- TEX and METAFONT
- use of third-party graphics tools

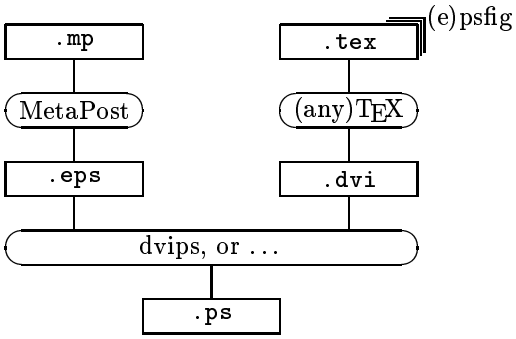
Almost everybody uses encapsulated PostScript (`epsf` for short) as the medium for merging graphics with TEX scripts, in order to get the results out.[2]

TEX alone. LATEX's picture environment is the most common example for this class, although plain TEXies might use the macros from `gkpmac`,[3] a subset of LATEX's picture functionality for use with plain TEX. An option I have developed within BLUe's format system introduces the use of 'turtle graphics,'[4] which was used in my "Publishing with TEX" (PWT) user's guide for simple fractals.[5] Interesting too is Gurari's approach (1994).

In scientific circles one problem is how to paste up mathematical graphs electronically. One approach is to calculate the graphs via Pascal, for example, letting it generate the TEX code for the graph. A few years ago I shuffled and typeset bridge hands via this method (van der Laan 1990). Now, however, I would solve the first problem via PostScript straightaway, and I would shuffle the cards, etc. via TEX alone.

Another problem deals with integrating text and graphics. For TEX alone integrating text with simple drawings is no problem at all; this is for me a reason to watch out continuously for what can be done via TEX alone. However, PostScript allows text to be integrated with graphics; better still, text is also considered as graphics, with the extra advantages of scaling or rotating.

Example *(Text integrated with drawings)*



The above graphic illustrates the process for combining TEX and MetaPost.[6]

TEX and METAFONT. John Hobby has recognized the power of METAFONT for designing (systematic) graphics and has married PostScript's outlines to METAFONT to arrive at his MetaPost program, banning the bitmap approach.[7] This is the path I am on, to emulate Naum Gabo's constructive art.[8]

Other approaches currently available include the `mftoeps` package by Jackowski and friends, which transforms METAFONT files into PostScript and vice versa; and `mfpic` from Leathrum and Tobin, which applies METAFONT's character handling technique to export graphics in general.

However, the main subject of this paper will be graphics via METAFONT.

Use of third-party tools. Of late more and more sophisticated graphics and multi-media software

[1] In the old mainframe days, documents were prepared with spaces left for graphics and tables, prepared by other tools, to be pasted up.

[2] Alas, there is no standard as yet for the use of `\special`-s. I hope Rokicki (1995) succeeds with his proposed standard.

[3] Used for typesetting *Concrete Mathematics* (Graham, Knuth, Pastashnik 1989).

[4] See my "Turtle Graphics and TEX — a child can do it," elsewhere in these proceedings.

[5] In this paper the Hilbert curves of order 1 and 2 have been handled via turtle graphics.

[6] Courtesy John Hobby.

[7] Designed under UNIX but also ported to DOS, Macintosh, ... AT&T has released MetaPost and add-ons in the public domain. Thank you.

[8] An artist; born Pevsner in Brjansk, 1890, died 1977 in the United States.

has been emerging, allowing *interactive* graphics (as opposed to systematic, reproducable, declarative graphics), among other things. Happily, the import and export of PostScript files is possible and therefore software, such as Adobe Illustrator, Photoshop, CorelDRAW, etc., can cooperate with TEX and METAFONT. Of course one could use PostScript throughout.

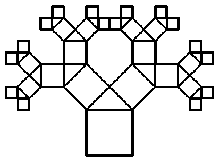
All of the approaches have their pros and cons. What to use — and when — depends as usual on your circumstances. However, third-party tools will not be dealt with in this paper.

METAFONT

For me, learning METAFONT was easier than learning TEX. I picked up the flavor from Knuth's first book in the field, *TEX and* METAFONT*: New Directions in Typesetting* (1979). Next, I read *The* METAFONT*book*, to absorb the ideas, possibilities and details. Finally, and inevitably, I practised with graphics examples borrowed from the literature.[9] A superb summary of the language and what you can achieve with it is given in Hobby (1992). As well, there are introductory documents such as Tobin's 'METAFONT for Beginners' (1993) and Jackowski's GUST tutorial (alas, only available in Polish).

As I built up my list of graphics (see Appendix), I found the path data structure an eye-opener. It shed new light on algorithms for the drawing of Hilbert curves, Sierpiński curves, Pythagorean trees[10] and the like, which are usually formulated recursively.

Example *(Pythagorean tree)*



The use of the path data structure in combination with METAFONT operations on pictures (e.g. `addto`), yields elegant, concise and fast non-recursive programs.[11]

MetaPost and extensions. MetaPost is (nearly) upwards compatible with METAFONT, and concentrates on graphics. It has turned away from the bitmap approach and combines the goodies of PostScript with METAFONT. MetaPost also provides for the integration of text and graphics, next to suitable I/O.

Hobby's graph extension (1993) is all about typesetting scientific graphs — the functionality of troff's grap recast in MetaPost (from Hobby's Introduction):

- automatic scaling
- automatic generation and labeling of tick marks or grid lines
- multiple coordinate systems
- linear and logarithmic scales
- separate data files
- ability to handle numbers outside the usual range
- arbitrary plotting symbols
- drawing, filling and labeling commands for graphs

Why?

I have a keen interest in the work of Naum Gabo. When I first heard of METAFONT it occurred to me that I could emulate his works. To put it in another way. I was curious whether METAFONT could also be used conveniently as a *design* tool for 3-D objects. From a computer science point of view, Gabo's constructive sculptures are very interesting, especially those composed of regular surfaces.[12] I supposed that if Gabo were alive today, he would have exploited the use of computers, since programming a computer — professionally known as software engineering — is a constructivistic activity.

PostScript straightaway. PostScript is a de facto standard and can be included in (LA)TEX documents. For the moment PostScript is intermediate

[9] This has resulted in files which I can easily walk through on my Mac, using Blue Sky's public domain METAFONT; it is not (yet) a database to load selectively from.

[10] Done in TEX with turtle graphics from my BLUe's format.

[11] Wirth (1976) has discussed the trade-off between data structures and algorithms. Apparently, he had not thought of the path data structure at the time.

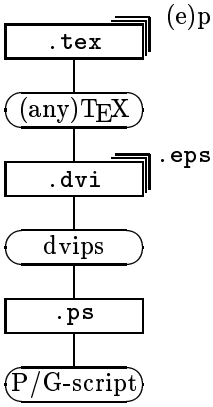
[12] A regular surface is determined by its boundary of which points are connected by straight lines. It conveys a picture of a 3-D object, via 1-D information.

in the chain dvi→ps→pdf;[13] a nodding knowledge of PostScript can only be beneficial.

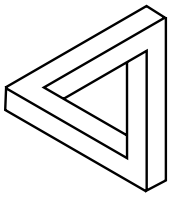
If we consider MetaPost to be PostScript with METAFONT as the user interface, then using MetaPost will bring you sooner or later to PostScript. After METAFONT, learning just a little bit of PostScript only cost me a couple of weeks. Known teasers — typesetting along curved paths, symmetric drawings, accurate drawing of math functions — these can be done elegantly by PostScript alone, and then included at the .dvi level. The problem of rotating document elements in general requires more interaction between (LA)TEX and PostScript.[14]

One interesting thing: because of METAFONT's path data structure, I uncovered a new coding for the Hilbert and similar curves, along with a systematic de-recursion technique.

Example *(Process flow using only TEX and PostScript)*



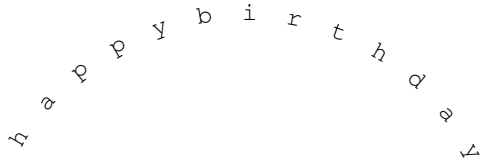
Example *(Escher's impossible triangle)*



```
%!PS EPSF
%%Title: Escher's impossible triangle
%%Creator: cgl (inspired  by Guy Shaw)
%%CreationDate: 1996.05.23:1858
%%BoundingBox: -40 -40 40 40
%%Pages: 1
%%EndProlog
%%Page: 1 1
%100 500 translate
3{25  34 moveto
  25 -34 lineto
  17 -38.2 lineto
  17 20 lineto
 -17.6 0 lineto
120 rotate
}repeat
stroke
showpage
```

Remark. I have also written a variant of the above, with only the first point specified and the rest obtained via (symmetry) operations. Although much more elegant I suspect it less intelligible, and I have therefore suppressed that code here.

Example *(Text set along an arc)*



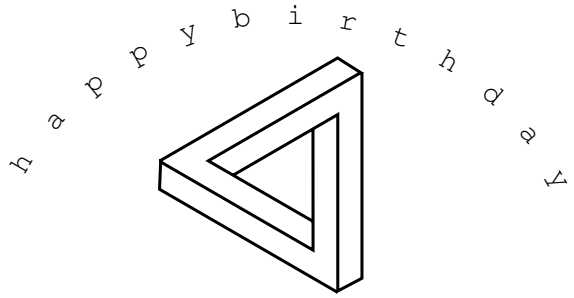

```
%!PS EPSF
%%Title: Typesetting along arcs
%%Creator: cgl
%%CreationDate: June 4 1996
%%BoundingBox: -100 50 100 125
%%Pages: 1
%%EndProlog
%%Page: 1 1
/Courier findfont 10 scalefont setfont
%150 650 translate
%
/text (happybirthday) def
60 rotate
0 1 12{0 100 moveto
  text exch 1 getinterval show
  -10 rotate
}for
stroke
showpage
```

Remark. Joseph Romanovsky (personal communication) has suggested that kshow — which allows kerning and positioning to be under the user's control — is the PostScript operator to typeset a string in such an unintended way.

[13] Adobe's Portable Document Format, which abstracts from the preparation tools and concentrates on the user side — the consumer — by the concise pdf format and the Acrobat reader, distiller (cross-referencing), exchange and ... multi-media tools.

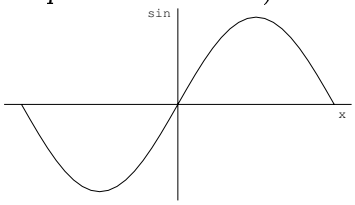
[14] There exists rotatebox.tex.

Example *(Seal)*




```
%!PS EPSF
%%Title: Typesetting along arcs: Seal
%%Creator: cgl
%%CreationDate: June 6 1996
%%BoundingBox: -100 10 100 110
%%Pages: 1
%%EndProlog
%%Page: 1 1
/Courier findfont 10 scalefont setfont
150 650 translate
%
/text (happybirthday) def
gsave
60 rotate
0 1 12{0 100 moveto
  text exch 1 getinterval show
  -10 rotate
}for
stroke
grestore
0 50 translate
3{25  34 moveto
  25 -34 lineto
  17 -38.2 lineto
  17 20 lineto
 -17.6 0 lineto
120 rotate
}repeat
stroke
showpage
```

Example *(Graph of sine function)*




```
%!PS EPSF
%%Title: Sine function
%%Creator: cgl (inspired  by Batagelj)
%%CreationDate: May 27 1996
%%BoundingBox: -200 -110  200 110
%%EndProlog
/Courier findfont 10 scalefont setfont
%figure scaled by 100
%x-axes
-200 0 moveto 200 0 lineto
%label
-5 -10 rmoveto (x) show
%y-axes
0 -110 moveto 0  110 lineto
%label
-25 -5 rmoveto (sin) show
%function
-180 0 moveto
-180 10 180{%from step to
   dup sin 100 mul%(x, 100sin x)
   lineto} for
stroke
showpage
```

The inclusion of `.eps` files can be done via `(e)psfig.tex` in cooperation with dvips.

Examples from my METAFONT Anthology

The various examples of codes outlined below were processed on a Mac, using Blue Sky's public domain METAFONT. How to code the pictures, unblurred by the `shipit` details, was the purpose.

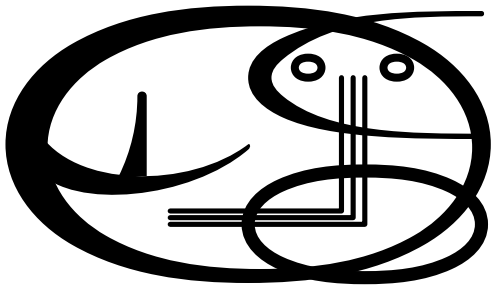
When the METAFONT shipping out of characters is used, the code must be modified: enclosing it with `beginchar` and `endchar` and providing `beginchar` with the appropriate arguments (as treated in *The* METAFONT*book*).

To use MetaPost, a few adaptations are needed: e.g., deleting the bitmap operations `cullit`, `screenstrokes`, and so on, and enclosing the picture with `beginfig` and `endfig` (or `begingraph` and `endgraph`, when the graph extension is used). The effects of reverse video via

```
addto blackbackground also
                -blackpicture
```

has to be adapted too; for example, via the use of (white) color.

Cat. This example is all about the use of a variable width pen. It gives an impression of what can be attained by METAFONT/MetaPost with respect to classical drawing.[15]



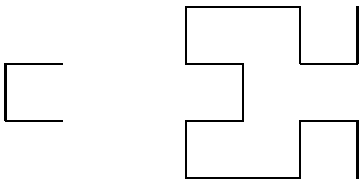
[15] I also developed a version still using a variable width pen but where but `fill` and `unfill`, more suitable for PostScript, were used.

METAFONT **code** The code is too lengthy (≈50 lines) to be included here. A MetaPost version will soon be available.

PostScript code The MetaPost code yielded a PostScriptfile — much bigger, alas — which is also available, and can be postprocessed if one wishes.

Hilbert curve. In Wirth (1976), the drawing of Hilbert and Sierpiński curves has been treated as essentially recursive. Via the path data structure and copying and rotating of paths built so far, it can be nicely de-recursified.

A Hilbert curve consists of 4 (rotated) copies of a base element connected by 3 straight lines, the 3 edges of a square. H_0 is a dot. The base element of H_k — a Hilbert curve of order k — is H_{k-1}, $k = 1, 2, \ldots$ H_1 and H_2 have been drawn below.[16]



The above graphic results from the following code:

```
$$\unitlength5ex
  \hbox{\qquad
  \vbox to3\unitlength
  {\offinterlineskip
   \vss\W1\S1\E1\vss\vss}
  \kern25ex
  \vbox to3\unitlength
  {\offinterlineskip
     \S1\W1\N1%rotated H_1
     \W1      %connector
     \W1\S1\E1%H_1
     \S1      %connector
     \W1\S1\E1%H_1
     \E1      %connector
     \N1\E1\S1%rotated H_1
    \vss}
}$$
```

TEX code A general TEX macro — parameterized over the order — reads as follows, where `\global<wind>` does what you expect:

```
\def\hlb{\ifnum\orderh=0 \blh\fi
   {\advance\orderh-1
   {\trsfst\hlb}\globalE1
   \hlb\globalN1
   \hlb\globalW1
   {\trssec\hlb}}\relax}
\def\trsfst{\let\exch\globalE
 \let\globalE\globalN
 \let\globalN\exch
    \let\exch\globalW
 \let\globalW\globalS
 \let\globalS\exch}
\def\trssec{\let\exch\globalE
 \let\globalE\globalS
 \let\globalS\exch
    \let\exch\globalN
 \let\globalN\globalW
 \let\globalW\exch}
\def\blh#1\relax{\fi}
```

METAFONT **code**

```
%Hilbert curve, variant.
%METAfont experiments, code builds upon
%Wirth's A+DS=P, pp130-133 (more concise,
%  uses path data structure; no redoing
%  of already constructed paths.)
%The code has been adapted to build up
%the *path*, and is non-recursive.
%December 1995, cgl@rc.service.rug.nl.
tracingstats:=1;
proofing:=1;screenstrokes;
autorounding:=0;
pickup pencircle scaled 1;
def openit = openwindow currentwindow
from origin to (screen_rows,screen_cols)
at (-20s,15s)enddef;
%
path p; s=10;
sz:=0;p:=origin;%H_0 size and path
n=5;            %Order of H-curve
for k=1 upto n: %H_1,...H_n
p:= p transformed (identity rotated 90
         reflectedabout (origin,up))--
    p shifted ((-sz-1)*s,0)--
    p shifted ((-sz-1)*s,(-sz-1)*s)--
    p transformed (identity rotated -90
             reflectedabout (origin,up)
            shifted (-sz*s,(-2sz-1)*s));
sz:=2sz+1;
draw p shifted(0,-10sz-10); showit;
endfor
end
```

Remarks. Order 4 overflows the rounding table limit (300) with the default autorounding. The connecting straight lines are implicit via `--`. Note the building up of the paths in `p`.

PostScript code Romanovsky has transliterated my (recursive) METAFONT code[17] into a concise PostScript program. A rewrite in the spirit of my TEX macro reads as follows:

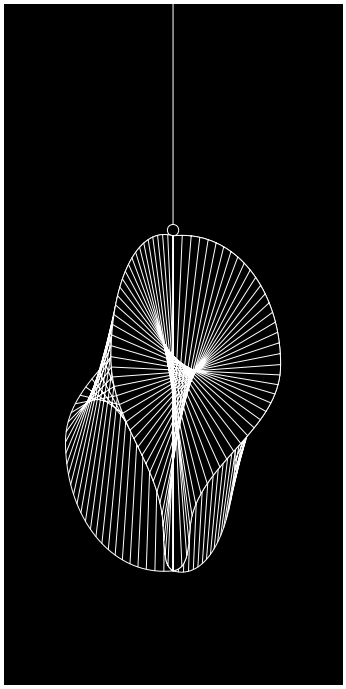
```
%reflect about (origin--(1,-1))
/R{90 rotate -1 1 scale}def
/invR{-1 1 scale -90 rotate}def
%reflect about (origin--(1,1))
/M{-1 1 scale 90 rotate}def
/invM{-90 rotate -1 1 scale}def
%size of element
/scgl 9 def
%
```

[16] Done in TEX, using turtle graphics from BLUe's format.

[17] Not provided here but will be included in my `anthology.mf`.

```
/H{dup 0 gt
   {1 sub
    M H invM scgl 0 rlineto
    H 0 scgl rlineto
    H scgl neg 0 rlineto
    R H invR
    1 add}
   if
}def
```

Linear construction in space II. Thirty years ago I was captivated by Gabo's constructive art, especially his 'Linear construction in space'-like objects. From the METAFONT point of view the following example is all about how to handle a 3-D object — that is, how to describe, project and draw Gabo's 3-D constructive art:



How to do this? How to develop a general technique?

Design First of all, it was necessary to view the object from different angles, which entailed the use of projection techniques.[18]

While programming the object in METAFONT, I had to resolve several issues:

a. how to transform a curve in space
b. how to preserve shape under projection of (a discretisation of) the curve while joining the projected points by METAFONT's splines
c. how to create equidistant points along a curve
d. how to avoid blurring lines
e. how to emulate the used (perspex) material

Coding I solved b and c by first creating the basic shape of a boundary in 2-D, and then replacing the curve by a set of (nearly) equidistant points along the curve. The coding of c reads essentially as follows, where `p10` takes over from `p1`:[19]

```
p10:=for t:=1 upto 19:
     point .05t of p1..endfor origin;
```

The rotation of a boundary curve (for example, from the *yz*-plane into the *xz*-plane) is simply coded as follows, where `p100` takes over from `p10`, etc.:

```
%in yz-plane (the screen)
p100:= for k=0 upto n-1:
  pointtopair(0,xpart(point k of p10),
                ypart(point k of p10))..
    endfor pointtopair(0,0,0);
%in xz-plane
p200:= for k=0 upto n-1:
  pointtopair(xpart(point k of p10), 0,
              ypart(point k of p10))..
    endfor pointtopair(0,0,0);
```

The projection is done via `pointtopair`, which in its simplest version reads as follows:

```
def pointtopair(expr x,y,z)=
%Purpose: The projection of 3D point
%into a pair in the projected plane.
%Arguments: x,y,z coordinates
   (-.6x+.8y,-4/13x-3/13y+12/13z)
enddef;
```

Some of the regular surfaces blur the picture.[20] To clear this up I removed the hidden line parts: to erase what is hidden is determined by the boundary of what is in front:[21]

```
erase fill p100..reverse p200..cycle;
```

The regular surface which is full-blown in sight, has been drawn simply via:

```
for k=0 step 1 until n:
draw point k of p100..point n-k of p200;
endfor
```

To emulate the 'light' caused by the perspex material, I used reverse video, as explained in *The*

[18] See Lauwerier (1987) for an introduction. It contains many examples in ... BASIC!

[19] Note that paths of dynamical length, determined at runtime, are created. I intend to fine-tune this by creating truly equidistant points along a curve via the use of METAFONT's `solve` and Hobby's `arclength`.

[20] In reality Gabo's aim was that from every viewpoint the object could be seen completely. There are no hidden lines in his art. He achieved this by using perspex and nylon.

[21] Knuth uses `overdraw` (1986, p. 243, ex. 13.11), which is very nice. His watchband logo has not been made of perspex apparently.

METAFONT*book*(pp. 115; 118 for the 'dangerous bend').

MF code Too lengthy (≈100 lines) to be included here. The MetaPost program will soon be available. Consult CTAN.

PostScript code As before, the MetaPost code yielded a much larger PostScriptfile; it too is available and can be postprocessed if one wishes.

Macro Facilities

Macro writing in METAFONT is completely different from macro writing in TEX. This paper is not intended as a tutorial on macros; it only provides a few highlights. An appetizer.

FIFO. My favorite FIFO paradigm — first-in-first-out — is implicit in the (var)def parameter handling. For example, the `max` macro (Appendix D of *The* METAFONT*book*, pp. 290–191) allows as argument a list of undetermined length:

```
max(a)   max(a,b,c)
```

Remark. A variable number of arguments is common in METAFONT; for example, `definepixels` and such can be invoked similarly. This is a consequence of (the abstract) `text` as parameter 'type.' MetaPost's `buildcycle` macro makes use of this feature, too, by allowing a list of paths, of undetermined length, as argument. Very useful.

Generic macros. For the `max` macro, for example, the type of the arguments can be numeric, pair or string. This is possible because METAFONT allows for testing for the type of an argument. This generic feature — the same macro for all relevant types — is powerful.

Neat, this abstraction of type and number of arguments, and definitely in agreement with Knuth's aim:

> The rules are intended to work the way you expect them.[22]

Gobbling. Another unusual feature is the infix primary `gobbled`, which not only absorbs the argument *after* but also *before*.[23] Infix operators can be defined with primary, secondary or tertiary levels of precedence.

Clipping boundary. Clipping is not provided as such by METAFONT. However, with `cullit` and `cull`, a picture can be clipped. Below, the current picture is confined to a (scaled) square, and provided with a fret:

```
...%picture so far
%reduce all pixels to 0 or 1
cullit;
%make pixel value of picture 2
%within the square
fill unitsquare scaled 100;
%retain picture within the square
cull currentpicture keeping (2,2);
%draw the boundary
draw unitsquare scaled 100;
```

The clipping by a square boundary is just an example to convey the idea. The approach can be applied to all kinds of shapes: for example, to a ring, as done by Jackowski in his EuroTEX95 paper (1995). Note that PostScript and MetaPost do have the concept of clipping path.

Length of a curve. The macro `length` can be applied to a path, resulting in the maximum 'time' rather than the arc length. Hobby applied Simpson's quadrature rule, which yields the following concise approximation (because we know the formula of the Bezier spline):[24]

```
vardef lengthpath expr p=
save dz; pair dz[];
dz0=point 1 of p - point 0 of p;
dz1=point 2 of p - point 1 of p;
dz2=point 3 of p - point 2 of p;
.5(length(dz0)+length(dz0+2dz1+dz2)+
   length(dz2))
enddef;
```

$$\int_0^3 \|B'\|\,dt \approx .5(\|\Delta z_0\| + \|\Delta z_0 + 2\Delta z_1 + \Delta z_2\| + \|\Delta z_2\|)$$

If the path's name is `p` an invoke might read `lengthpath p`, or for a subpath `lengthpath(subpath (3,6) of p)`.

Selective loading. In BLUe's format system the mechanism of selective loading has been used to build a database of tools, pictures and so on. This functionality can also be implemented in METAFONT. The file to load selectively from consists again of triples: list element tag, a symbolic token,

[22] Some codes in Appendix D are not that intuitive, however.

[23] Peruse Appendix B of *The* METAFONT*book* for these kinds of features.

[24] Note that the approximation is only good for sufficiently smooth curves. Split up complex curves in simple ones. Another approach is mentioned by Gibbons (1995). The length of a Bezier spline is bounded by its convex hulls; the smaller the piece, the closer the upper and lower bounds. Repeated division of the curve and summing the lengths of the pieces yields the length.

and text enclosed by parentheses and ended by a semicolon. The list element tag macro has 2 arguments: the implicit suffix — which is compared as string with the name of what we want to select — and the text enclosed by parentheses. If the suffix agrees with the required name, a macro of this name is defined, with the text as replacement text. For example:

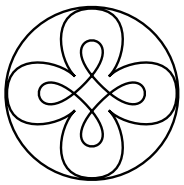
```
vardef lst@#(text t)=
  if str @#= s:%s=selection key
    def @#= t enddef
  fi enddef;
input macro.lst
```

The (toy) file `macro.lst` might read:

```
lst na (draw unitsquare scaled size;);
lst ns (rt);
```

The result with `s:="na";` reads:

```
def na=draw unitsquare scaled size;
enddef;
```



Pitfalls in Compatibility

For my cat, the METAFONT program processed by MetaPost does not yield the correct result. The moustache has disappeared!?! However, if the moustache code is moved to the end, then the correct result is obtained. What has happened? In my opinion this is a consequence of the fact that pixels have values (*The* METAFONT*book*, p. 109).

> Pixels aren't simply "on" or "off" when METAFONT is working on a picture; they can be "doubly on" or "triply off". Each pixel contains a small *integer* value ...

MetaPost's path is apparently just on or off. A `draw <path>` followed by an overlapping `fill` and `unfill` makes that the path disappears. It is better to let the `draw <path>` follow the `fill` and `unfill`.

Example *(Ring with center)*
The compatible way to program this in METAFONT reads essentially as follows:

```
fill fullcircle scaled 10;
unfill fullcircle scaled 7;
drawdot origin;
```

Another incompatibility is in the clipping functionality. In METAFONT this can be done by manipulating pixel values, while MetaPost provides a primitive biased by a clipping contour.

Summarizing my experiences

MetaPost combines the best of both worlds: METAFONT's language features are enriched with contours and `epsf` output.[25] Moreover, it allows access to PostScript's wealth. For creating pictures to be included in (La)TEX or troff documents, MetaPost can be looked upon as PostScript and a METAFONT user interface. Perhaps METAFONT/MetaPost will find their niche in history as convenient tools to describe pictures concisely and at a high level. Via the use of regular surfaces an impression of 4-D can be obtained from 1-D — the contours — information.

What I like about METAFONT**.** First of all, I like very much the quality, the stability and its being for free. Next, it is available for nearly every platform. Finally, there are the following pleasing details:

- the declarative nature of the language
- the meta-ness and the generic aspects
- the generalization of variable, subscripted variable or record field variable into ⟨tag⟩ ⟨suffix⟩
- just 3 kinds of arguments: expr (independent of type), suffix, and text
- the operator definitions with built-in priorities (`primarydef`, etc.)
- pen, path and picture data structures and operations
- filling and erasing operations
- operations for intersection points
- (nonlinear) interpolation between curves
- various handy 'syntactic sugars'
- rich tracing facilities[26]

If Descartes' Analytic Geometry is still taught today, it might benefit from METAFONT for visualizing.

What I miss in MF. In the list below of missing items, those marked with a plus sign are provided in MetaPost:

+ outlines or `epsf` output (it is all about bitmaps of fonts)
+ mixing of text and pictures[27]
+ general file I/O (writing and reading of pictures)
+ a suitable number range (it is restricted to the half-open interval [1/256/256 − 4096])

[25] It is nearly upwards compatible; `cullit` and other bitmap operations have to be replaced.

[26] Agreed, a necessary evil.

[27] Not to mention Hoenig's typesetting along curved paths (1991).

+ generality of rotating pictures (restricted to a multiple of 90°)
+ various dashed/dotted lines
+ arrows
+ clipping
+ shading and greyscales
+ color
+ invoking TEX
+ making use of PostScript facilities
– triple datatype — point in 3-D as analogue of `pair`
– tree data structure (pointer/handle)

Then there is Hobby's graph extension with its functionalities (as listed earlier), which I have not yet missed, but which I will need for sure when typesetting scientific graphs gracefully.

METAFONT/MetaPost as a production tool. History has it that TEX has been used mainly by scientist with substantial, complex copy, full of mathematics, tables, or graphics, who wish to publish via the electronic networks, via the Internet, not to mention the creative self-publishing world. METAFONT has been used extensively for font production by linguists for non-Latin alphabets.

I don't know of better tools for graphics which are as reliable, portable, ubiquitous, open, completely documented, stable, and cooperative towards other tools. With respect to the last, one can think of the various printer and screen drivers, PostScript and PDF, and the new arrival, HTML — HyperText Markup Language. Moreover, the twins TEX&METAFONT (and descendants) are in the public domain and ported to every platform.

However, one has to learn the systems to know what is under the hood. TEX&METAFONT are not of the push-the-button type tools, like washing machines or cars. Therefore, education is paramount.

The METAFONT/PostScript experts of the Polish TEX User Group GUST have reported that a necessary condition for METAFONT/MetaPost to become a production tool is that `epsf` function as the medium to ease the cooperation with third-party tools. They also have a standing wish for a big METAFONT. As far as I know the company BoP s.c.[28] is the only firm with METAFONT in production.

The future. Maybe we should no longer use paste-up for figures. How about creating hyperlinks to a picture database? If we want to see the picture we can just click and there it is. This is similar to when we like to hear the music when we read about a composer or read the music notation. Whatever these new ways will bring, I for one like the complete document on paper.

[28] Bogusław Jackowski and Piotr Pianowski.

Acknowledgements

Thank you, Joseph Romanovsky, Geoffrey Tobin and John Hobby, for your friendly support. Piet van Oostrum processed 'linear construction no. 2' via MetaPost. The picture came out, for sure, but at the expense of a large PostScript file. Jos Winnink provided the PostScript files via MetaPost and commented on early versions of the paper.

Further references can be found in the file `lit.dat`, on CTAN in `/macros/blu`.

Have fun, and all the best.

References

[1] Gabo, N. *Sixty Years of Constructivism*. Munich: Prestel, 1985.

[2] Gibbons, J. "Dotted and Dashed Lines in METAFONT." *TUGboat* 16,3 (1995), pages 259 – 264.

[3] Graham, R.L., D.E. Knuth, and O. Pastashnik. *Concrete Mathematics*. Reading, Mass.: Addison-Wesley. 1989. [Includes the `gkpmac` macros, available from CTAN.]

[4] Gurari, Eitan M. *TEX and LATEX: Drawing and Literate Programming*. New York: McGraw-Hill, 1994.

[5] Hobby, J.D. *A User's Manual for MetaPost*. Computing Science Technical Report 162. Murray Hill, NJ: AT&T Bell Laboratories, 1992. [Can be obtained from `netlib@research.att.com` via `send 162 from research/cstr`.]

[6] Hobby, J.D. *Drawing Graphs with MetaPost*. Computing Science Technical Report 164. Murray Hill, NJ: AT&T Bell Laboratories, 1993. [Can also be obtained as noted above.]

[7] Hoenig A.J. "When TEX and METAFONT Talk: Typesetting on Curved Paths and Other Special Effects." *TUGboat* 12,3 (1991), pages 554 – 557.

[8] Jackowski, Bogusław. A GUST tutorial on METAFONT. [Currently being translated into English.]

[9] Jackowski, Bogusław. "A METAFONT-EPS Interface." *Proceedings of the Ninth European TEX Conference* (Sept. 4 – 8, 1995, Arnhem, The Netherlands), 257 – 271. [`mftoeps` is available via ftp from `ftp.pg.dga.pl` in the `/pub/TeX/GUST/contrib/MF-PS` subdirectory.]

[10] Knuth D.E. *TEX and* METAFONT*: New Directions in Typesetting*. Part 1: *Mathematical Typography*; Part 2: *TEX: A System for Technical Text*; Part 3: METAFONT*: A System for Alphabet Design*. AMS. Digital Press. (1979).

[11] Knuth, D.E. *The* METAFONT*book*. Reading, Mass.: Addison-Wesley, 1986.

[12] Lauwerier H.A. *Meetkunde met de microcomputer*. Epsilon 8, 1987. [Basic programs are included.]

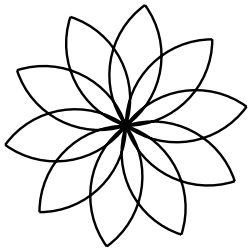
[13] Leathrum, T., and G. Tobin. `mfpic`. 1994. [Available from CTAN in the `graphics` subdirectory.]

[14] Rokicki, T. "A Proposed Standard for Specials." *TUGboat* 16,4 (1995), pages 395–401.

[15] van der Laan, C.G. "Typesetting Bridge via TEX." *TUGboat* 11,2 (1990), pages 265–276.

[16] Tobin, G. METAFONT for Beginners. 1993. [Available from CTAN in the `info` subdirectory.]

[17] Wirth, N. *Algorithms + Data Structures = Programs*. Englewood Cliffs, NJ: Prentice-Hall, 1976.

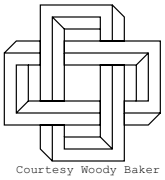


Appendix ToC METAFONT Anthology

I hope that the codings mentioned below will contribute to the METAFONT 'literature.'

```
%Anthology of METAFONT endeavours,
%Started Nov 1995, revision March 1996
%Author/Compositor: Kees van der Laan,
%Hunzeweg 57, 9893PB, Garnwerd, Holland
%                 cgl@rc.service.rug.nl.
%The anthology files work under
%BLUe sky's METAFONT (Public Domain)
%on the Macintosh.
%(How to scroll through these examples on
% other systems I don't know.
% Just copy what is of interest for you
% and emulate...
% towards a discipline of coding in
% METAFONT/METAPOST.)
%For other systems copy the example(s)
%you are interested in build a character
%from it and follow the usual procedures
%to include the font in your (La)TeX,
%or use for example the Jackowski/Rycko
%mftoepsf package to produce epsf.
%For use in METAPOST enclose the elements
%by beginfig ...endfig.
%
%Disclaimer borrowed from
%Preface METAFONT book:
%'That's the way how it is with any
% powerful tool:
%    There is always more to learn, and
%    there are always better ways to do
%    what you have done before.'
%
%Table of Contents
%%%Preliminary, file ant.mac
%-Macros used
% (default) openit
% pointtopair projection
% dash
% turtle movements:
%    east, south, west, north
%-Defaults
%%%ToC proper%%%
%
%%%Tiles, file: ant.til
%-Roos and variation via interpath
%-Variations via interpath
%-Mondriaan
%-Alhambra tiles (Courtesy M C Escher)
%-Escher tiles
%-Escher reptiles I
%-Escher reptiles II
%-Escher Buddhas
%-Escher square limit
%-Meeting and meeting apart
%-Moriscs ornament (Courtesy
%                   J V Romanovsky)
%-Chinese porcelain ornament
%       (Courtesy J V Romanovsky)
%-Chinese porcelain ornament II
%-Horak's tiles
%%%2D figures, file: ant.two
%-Cat (varying pen width, non-linear
%      interpolation)
%-Cat II (portable?)
%-Cat III(essential picture)
%-Whirlpool (Courtesy H A Lauwerier)
%-Generalized polygon
% (sides as flexes)
%-Flexes II (Courtesy B Jackowski)
%-Flexes III(Interpath)
%-2D polygon regular patterns
%-Nails (Courtesy J V Romanovsky)
%-Nails II
%-Nails III
%-Yin-Yang
%-Removing Overlap and
% Expanded Stroke
%-Turtle graphics: square spiral
%-Spiral of squares
% (Courtesy H A Lauwerier)
%-Recycle logo (Courtesy P Mackay)
%-Recycle logo II
%
%%%3D, file: ant.thr
%-2.5D simplest example:
% twisted plane
```

```
%-Cube and impossible cube
% (MB 13.7, 2.5D variant)
%-Tetraeder
%-Tetraeder and
% Gabo's torsions inside
%-Octaeder
%-Ikasoeder
%-Triangular edge Moebius band
% (Courtesy Tuckerman)
%-Hobby's pyramid
% (without text etc) in 2.5D
%-Hypercube
%-Spiral 2.5D
%-Emulating Gabo:
% doll's
% Hyperboloid
% Linear construction I
% Linear construction II
% Speric theme
% Speric theme;
%   interpolating surfaces
% Vertical construction 0
%-Toroid of rotating circles
%-Toroid of rotating circles
% (2.5D variant)
%-Cube example,
% varying viewing angles
%-Moebius band,
% varying viewing angles
%
%%%OP art, file ant.opt
%-Optical illusion
% (jiggling squares)
%-Optical illusion (parallel?)
%-Op/kinetical art (Courtesy Soto)
%-Knuth meets Vasarely
%-Vasarely II
%-Vasarely III
%-Vasarely IV
%-Vasarely V
%-Vasarely VI
%-Op art pulsing I
% (Courtesy Jean Larcher)
%-Op art pulsing II
% (Courtesy Jean Larcher)
%-Op art (Profile)
%-Op art (circles and ellipses)
% (Courtesy Schrofer)
%
%%%Fractals, file: ant.frc
%-Cantor dust fractal
%-Hilbert curve
% (Courtesey N Wirth)
%-Sierpinski curve
% (Courtesey N Wirth)
%-W-curve (Courtesey N Wirth)
%-H-fractal
%-Pythagorean tree
%-Pythagorean tree, non-recursive
%-Sierpinski square
% (Courtesy B Jackowski)
%-Sierpinski triangle
% (recursive with use of save)
%-Sierpinski triangle (without save)
%-Sierpinski carpet (via gambling)
%
%%%Math curves
%-Conic sections
% Ellips, hyperbole, parabole
%-Lemniscate
%-Astroid
%-Cardoid
%-(Hypo)cycloid
%-Deltoid
%-Nefroid
%-Spirals: Square, Archimedes,
% Logarithmic, Spheric
%
%%%Various, file ant.var
%-Haralambous' deformations
% of a 'circle'
%-Calculation of (arc)length of a
%  Bezier segment via quadrature
%-Exercise from MB: p.176 (solve use)
%-Exercise from MB: 13.8  (star)
%-Exercise from MB: 13.10 (S-figure)
%-Exercise from MB: 13.10 (S-figure
%     contour via 'expanded stroke')
%-Exercise from MB: 13.11
% (Moebius band)
%-Exercise from MB: 13.11
% (variant 2.5D Moebius band)
%-Example from MB 14 p134
% (Interpath use: heart interpolation)
%-Exercise from MB: 15.6
% (variant, via paths)
%-Exercise from MB: 20.5
% (variant, parameter sep TeXnique?)
%-n faculty (exercise recursion,
% number range)
%-n asterisks(exercise recursion)
%-quicksort  (exercise recursion)
%%%end ToC
```



Courtesy Woody Baker

◇ Kees van der Laan
Hunzeweg 57
9893 PB Garnwerd
The Netherlands
Email: cgl@rc.service.rug.nl