Graphics

Graphics and TEX — a reappraisal of METAFONT/METAPOST

Kees van der Laan

Introduction

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

- TEX alone
- TEX and METAFONT
- Use of 'third party' graphics tools.

Everybody uses epsf package in the last two cases as medium to merge the graphics with the TEX script in order to get the results out.[2]

TEX alone. LATEX's picture environment is the common example for this class, though plain TEXies might use the macros from Graham/Knuth/Patashnik,[3] which reflect a subset of LATEX's picture functionality in plain TEX. In BLUe's format system I also introduced the use of 'Turtle graphics.' I have used this in PWT — Publishing with TEX user's guide — for simple fractals.[4]

Interesting too is Gurari's approach.

In scientific circles a problem is how to paste up mathematical graphs electronically. An approach is to calculate the graphs via Pascal or so and let Pascal generate the TEX code for the graph. A few years ago I shuffled and typeset bridge hands via this method.

These methods will not be dealt with in this paper.

1: In the old 'mainframe' days documents were prepared with space left open for graphics and tables, prepared by other tools, to be pasted in.

2: Alas, there is no standard as yet for the use of special-s. I hope Rokicki will succeed with his proposed standard. The inclusion of epsf in (La)TEX documents will not be treated in this paper.

3: Used for typesetting their book Concrete Mathematics.

4: In this note the Hilbert curves of order 1 and 2 have been handled via 'Turtle graphics.'

TEX and METAFONT. John Hobby recognized the power of METAFONT for the design of (systematic) graphics and married PostScript's outlines to METAFONT in his METAPOST, banning the bitmap approach.[5] This is the path I'm on to emulate Naum Gabo's constructive art.

Other approaches are for example

- Jackowski's[6] mftoeps package to transform METAFONT files into PostScript and vice versa
- Leathrum/Tobin's mfpic which applies METAFONT's character handling technique to export graphics in general.

Graphics via METAFONT is the main subject of this paper.

Use of third party tools. Of late more and more sophisticated graphics and multi-media software emerges, which allow *interactive* graphics[7] among other things. Happily, import and export of PostScript files is possible, and therefore the software — 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.

The third party tools will not be dealt with in this paper.

METAFONT

Learning METAFONT was easier for me than to learn TEX. I picked up the flavour from Knuth's (first) book in the field 'TEX and METAFONT – new directions in typesetting.' Next, I read *The* METAFONT*book* to absorb the ideas, possibilities and details. Finally, and inevitably, I exercised graphics examples borrowed from literature.[8]

5: Designed under UNIX but also ported to DOS. AT&T has released METAPOST and add-ons in the public domain. Thank you.

6: And friends.

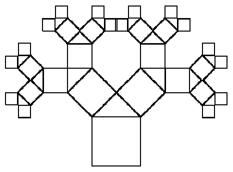
7: As opposed to systematical, reproducible, declarative graphics.

8: This resulted in a file called anthology.mf, which I can easily walk through on my Mac with BLUe Sky's METAfont. It is not (yet) a database to load selectively from.

A superb survey of the language and what you can achieve with it, is given by Hobby in his 'A user manual for METAPOST, AT&T, CSTR 162.'[9]

Next to that there are the introductions: Tobin's 'METAFONT for Beginners' and Jackowski's GUST tutorial. The latter, alas, is in Polish.

During my building up of 'A graphics anthology in METAFONT,' 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, and ilks, which are formulated usually recursively.



However, use of the path data structure in combination with METAFONT's operation on pictures, like addto ... , yields elegant, concise and fast non-recursive programs.[10]

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

Hobby's graph extension has been treated in AT&T CSTR 164. It is all about typesetting scientific graphs, the functionality of troff's grap recasted in METAPOST. The following functionalities are provided[11]

- 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 works of Naum Gabo.[12] When I 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 be used conveniently as a *design* tool for 3D objects too. From a computer science point of view Gabo's sculptures are very interesting, especially those composed of regular surfaces.[13]

> 4D impression can be obtained from 1D information via the use of regular surfaces, projection techniques, and varying viewpoint angles.

I guess, if Gabo would have lived now, he would have exploited the use of computers, because programming a computer — professionally known as software engineering — is a constructivistic activity.

PostScript straightaway. I have no convincing arguments against nor in favour. Perhaps, it is a matter of taste, biased by the stability and high-levelness of METAFONT, next to the reputation of Don Knuth. On the other hand PostScript is a de facto standard, PostScript can be included in (La)TeX documents, and for the moment PostScript is intermediate in the chain dvi→ps→PDF,[14] and therefore a nodding knowledge of PostScript is beneficial anyhow.

One thing for sure, however. Because of METAFONT's path data structure I uncovered a new coding for the Hilbert and similar curves, next to a systematic derecursion technique.

9: Can be obtained by mailing to `netlib@research.att.com` *send 162 from research/cstr.* CSTR 164 can be obtained analogously.

10: Wirth has discussed the trade-off between data structures and algorithms, in his A + DS = Programs. Apparently, he did not think of the path datastructure at the time.

11: Borrowed from the CSTR 164 Introduction.

12: Born Pevsner at Brjansk, 1890.

13: A regular surface is determined by its boundary — 1D — of which points are connected by straight lines. It conveys a picture of a 3D object.

14: Adobe's Portable Document Format, which abstracts from the preparation tools and concentrates on the userside — the consumer — by the concise PDF format and the Acrobat reader, distiller (cross-referencing), exchange and ... multi-media tools.

The question evaporates into thin air if we look at METAPOST as PostScript with METAFONT as user interface. Hobby has enriched METAPOST by the add-on tool graph, which provides troff's grap functionality and a little more.

Examples

In the sequel a few excerpts from my METAFONT anthology.

The included codes work as such on a Mac with BLUe Sky's 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 adapted, such as enclosing it by `beginchar` and `endchar` and providing `beginchar` with the appropriate arguments as treated in *The* METAFONT*book*.

For using METAPOST a few adaptations are needed such as deleting the bitmap operations `cullit`, `screenstrokes`, and ilks, and enclosing the picture by `beginfig` and `endfig`, or `begingraph` and `endgraph` when the graph extension is used. The effect of reverse video via

```
addto blackbackground also -blackpicture
```

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

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

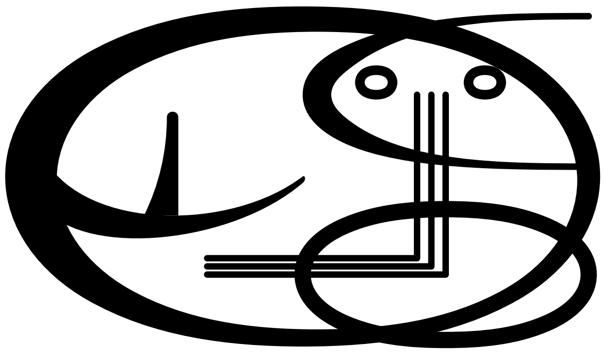


Figure 1: Cat

METAFONT program

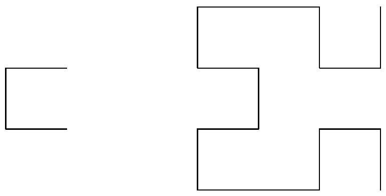
The file is 70 lines. Consult anthology.mf.

Hilbert curve. In Wirth's 'Algorithms + Data structures = Programs' the drawing of Hilbert and Sierpiński curves have been treated as essentially recursive. Via the path data structure and copying and rotating of pictures built so far, it can be nicely derecursified.

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}, $k = 1, 2, \ldots$.

Below H_1 and H_2 have been drawn.[15]



The above picture has been obtained in the spirit of graphics via TEX alone as follows.

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

METAFONT program

Remarks. Order 4 overflows the 'rounding table' limit (300) with the default autorounding. The connecting straight lines are implicit via `--`.

PostScript program

Joseph Romanovsky has transliteraded my (recursive) METAFONT code into the following concise PostScript program.

```
/S{0 R rlineto currentpoint
  stroke moveto}def
/T{90 rotate}def /TM{T 1 -1 scale}def
/H{TM dup
```

15: Done in TEX by means of BLUe's format Turtle Graphics.

```
  0 gt {1 sub H S TM H S H T S
   -1 1 scale H 180 rotate 1 add} if
   TM}def
/R 8 def
100 100 moveto
6 H pop
showpage
```

Linear construction in space II. Thirty years ago I was caught by Gabo, by his constructive art, especially by his 'Linear construction in space'-like objects.

From the METAFONT viewpoint this example is all about how to handle a 3D object, that is how to describe, project and draw Gabo's constructive art.

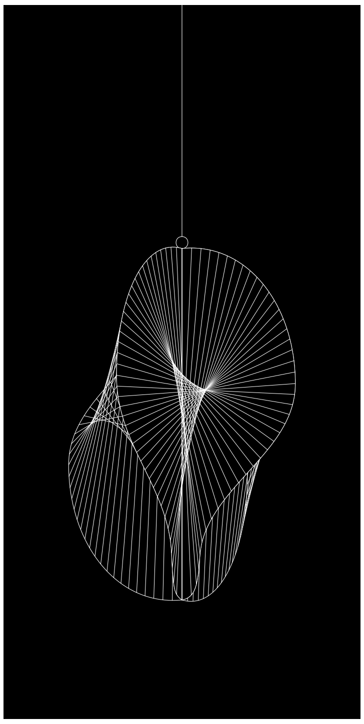


Figure 2: Lin Const in space no.2

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

Design

It is needed to view the object from different angles. This entailed the use of projection techniques.[16]

While programming the object in METAFONT I had to solve

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?

16: See Lauwerier's 'Meetkunde met de micro computer,' for an introduction. It contains many examples in ... BASIC.

Coding

I solved *b*) and *c*) by first creating the basic shape of a boundary in 2D, and then replace 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.[17]

```
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 coded simply 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 a 3D point
%into a pair in the projection plane.
%Arguments: x,y,z coordinates of a point
   (-.6x+.8y,-4/13x-3/13y+12/13z)
enddef;
```

Some of the regular surfaces blur the picture.[18] To clear this up I removed the 'hidden' line parts. To

17: Note that paths of dynamical length — that is determined at runtime — are created. I fine-tuned this by creating really equidistant points along a curve via the use of METAFONT's solve and Hobby's arclength.

18: 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.

erase what is hidden is determined by the boundary of what is in front.[19]

```
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 the material perspex I used reverse video as explained in *The* METAFONT*book* 115, 118 for the 'dangerous bend.'

METAFONT program

The file is 120 odd lines. Consult anthology.mf.

Macro facilities

Macro writing in METAFONT is completely different from macro writing in TEX. This note is not aimed as a tutorial on macros, it 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 macro `max`[20] 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 ilks 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 in allowing a list of paths, of undetermined length, as argument. Useful it is.

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

19: Knuth uses `overdraw`. (see for example ex13.11) which is very nice. His watchband logo has not been made of perspex apparently.
20: Of Appendix D.

Neat, that abstraction of type and number of arguments, and definitely in agreement with Knuth's aim[21]

'The rules are intended to work the way you expect them.'

Gobbling. Another unusual feature is the infix primary gobbled, which not only absorbs the argument *after* but also *before.*[22] Infix operators can be defined with primary, secondary or tertiary level 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 EuroTEX *96* paper.

Length of a curve. The macro length can be applied to a path with as result the maximum 'time,' not the arc length. Hobby applied Simpson's quadrature rule, which results in the following concise approximation, because we know the formula of the Bezier spline.[23]

21: Some codes in Appendix D are not that intuitive, however.
22: Peruse Appendix B for these kinds of features.
23: Note that the approximation is only good for sufficient smooth curves. Split up complex curves in simple ones. Another approach is mentioned by Gibbons in his TUG *95* paper. The length of a Bezier spline is bounded by its convex hulls. The smaller the piece the closer the upper and lower bound. Repeated division of the curve and summing the lengths of the pieces yields the length.

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

```
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;
```

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 ilks. This functionality can be implemented in METAFONT too. The file to load selectively from consists again of triples: list element tag, a symbolic token, 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 the definition of the list element tag might read as follows.

```
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 as follows

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

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

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

Pitfall in compatibility

The METAFONT program for my cat processed by METAPOST does not yield the correct result. The moustache has disappeared!?! If we move the moustache code to the end then the correct result is obtained.

What has happened?

In my opinion this is a consequence of that pixels have values, to quote *The* METAFONT*book* 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.

The compatible way to program this in METAFONT reads essentially as follows.

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

Summarizing my experience

METAPOST combines the best of both worlds: METAFONT's language features are enriched with contours and epsf output.[24] Moreover, it allows access to PostScript's wealth.

METAPOST can be looked upon as

PostScript with a METAFONT user interface

to create pictures to be included in (La)TEX or troff documents.

Maybe METAFONT/METAPOST will find their niche in history as convenient tools to describe pictures concisely and at a high level.

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

- 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

24: It is nearly upwards compatible. `cullit` and other bitmap operations have to replaced.

– (nonlinear) interpolation between curves
– various handy 'syntactic sugars,' and
– rich tracing facilities.[25]

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

What I missed in METAFONT. In the list of missing items below those denoted by + are provided in METAPOST.

+ outlines or epsf output (it is all about bitmaps of fonts)
+ mixing of text and pictures[26]
+ general file I/O (writing and reading of pictures)
+ a suitable number range (restricted to the half-open interval [1/256/256 – 4096))
+ generality of rotating pictures (restricted to a multiple of 90°)
+ dashed/dotted kind of lines
+ arrows
+ clipping
+ shading and greyscales
+ colour
+ to invoke TEX
+ to make use of PostScript facilities
– triple datatype — point in 3D as analogon of pair
– tree datastructure (pointer/handle).

Then there is Hobby's 'graph' extension with functionalities[27] I did not miss yet, but which I do need for sure when typesetting scientific graphs, gracefully.

METAFONT/METAPOST as 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 internet, not in the least in the creative off-off self-publishing world. METAFONT has been used mainly for font production by off-off linguists for non-Latin alphabets.

I don't know of better tools for document production which are so reliable, portable, ubiquitous, open, completely documented, stable, and cooperative towards other tools. With respect to the latter one can think of the various printer and screen drivers, PostScript and PDF, and the new hype 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 the necessary condition for METAFONT/METAPOST to become a production tool is that epsf can function as medium to ease the cooperation with third party tools. As far as I know 'BoP s.c.' (B. Jackowski and P. Pianowski) is the only company with METAFONT in production.

The future. Maybe we should no longer paste up for figures. What 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.

Acknowledgments

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, at the expense of a large PostScript file.

Jos Winnink transcripted the METAFONT pictures into METAPOST. He also lend the usual helping hand in transforming the BLUe scripts into MAPS submissions.

For the details of the references search the file lit.dat which comes with BLUe's format system.

◇ Kees van der Laan
cgl@rc.service.rug.nl

25: Agreed, a necessary evil.
26: Not to mention Hoenig's typesetting along curved paths.
27: Enumerated earlier.