Software & Tools

Even more MetaFun with METAPOST: A request for permission

Alexander Berdnikov, Hans Hagen,
Taco Hoekwater and Bogusław Jackowski

Introduction

In May 2000, at the Bachotek conference in Poland, an evening session was organized on extending METAPOST. It's out of love for this program, not out of frustration, that we discussed this topic, and Bachotek provided the right ambiance. For the record: some more people were present at the meeting than the four of us.

1 Accuracy

There are two limitations that sometimes conflict with our requirements. First, there is an upper bound of 4K bp, which can be raised to 32K by disabling a warning, but then some internal intermediate results can fail. The upper bound is not a real limitation in METAFONT, since normally (type 1) fonts are designed on a 1000 times 1000 bp grid. However, in the area where METAPOST is used, this limitation can hurt.

The second limitation concerns the accuracy. There are situations where one expects (for instance) two points to be the same, but when they are calculated in a different way, they slightly differ.

2 Unfill

Instead of implementing an unfill as "fill with white", a real unfill operator should be available. Such a mechanism can probably be combined with the weighted paths mentioned below.

3 Picture operations

In METAPOST a picture is a collection of paths, drawn or filled, using some color and a certain pen. The full range of operations available in METAFONT should also apply to METAPOST: shift, scale, rotate, add, subtract, etc.

4 Weighted paths

Where in METAFONT a path can have a weight, in METAPOST only a color can be applied. In addition to color, a weight should be available, and METAPOST should be capable of dealing with them in a similar way. If needed, a bitmap model, like the one available in METAFONT, should be available in parallel (for calculations, etc.).

5 Intersections

Calculating the intersection points of two paths should be implemented in the core system and not in macros. In a similar way it should be possible to reduce overlapping and self-overlapping contours into elementary ones. Although difficult border cases are very hard to solve, METAPOST should be capable of handling cases of average complexity.

6 Pens

Circular and elliptical pens should be re-implemented in a different way. Currently these pens are reduced to drawing a path with a certain thickness, if needed combined with a PostScript transformation. Instead of drawing, the effective path should be a filled combination of Bezier curves. The user should be able to specify the accuracy as well as methods, such as: try to approximate the resulting contour as well as possible, keep the approximation inside the contour, or keep it outside.

7 Fonts

Although METAPOST provides a way to embed text typeset by TEX, and therefore gives access to the full TEX machinery, it would be nice if the more raw text processing would also honor ligatures and kerning. The code that is needed can be derived from other programs in the TEX suite.

8 Specials

Specials should be re-implemented and become path or point specific in order to achieve special effects.

Conclusion

We are aware of the fact that the schedule of the author of METAPOST, John Hobby, does not permit him to implement these extensions, so we are pleased to hear that he is willing to grant us permission not only to explore these extensions, but also to realize them, and that he is willing to participate in the process of extending METAPOST.

Since we know that stability as well as compatibility are big issues in the TEX community, we are very well aware how delicate the process is of merging extensions as proposed by us and John himself into the existing program. We will do our best to make sure that the burden of merging code will be minimized. And, most of all, we will try not to spoil the beauty of the existing code.

We love METAPOST too much to compete, but we long to complete.

◇ Alexander Berdnikov
Institute of Analytical Instrumentation
St. Petersburg, Russia
berd@ianin.spb.su

◇ Hans Hagen
PRAGMA ADE
Ridderstraat 27
8061 GH Hasselt, The Netherlands
pragma@wxs.nl

◇ Taco Hoekwater
Kluwer Academic Publishers
Achterom 119
3311KB Dordrecht, The Netherlands
taco.hoekwater@wkap.nl

◇ Bogusław Jackowski
BOP sc.
ul. Piastowska 70
80-331 Gdańsk Oliwa, Poland
b.jackowski@gust.org.pl

Extending METAPOST: Response to "Even more MetaFun"

John D. Hobby

Introduction

I agree that METAPOST could use some improvements and bug fixes, but I have been too busy to do any of this work during the last two years or so and I do not expect that situation to improve soon. It is OK with me if others want to make upward-compatible improvements to `mp.web`, but such implementors should know enough about the program to be able to go through the "The command codes" section of `mp.web` and be familiar with everything it refers to. To avoid competing versions, extensions should probably be sent to me (`hobby@research.bell-labs.com`) for inclusion in the master version.

9 Accuracy

It would improve the METAPOST language to switch from 32-bit scaled integers to 64-bit floating point but this would be a lot of work and it would have to be done very carefully. The scale factors of 2^{16} and 2^{28} could be retained, but it would still be necessary to go through the entire program changing numerical constants and looking for places where the new arithmetic violates hidden assumptions. The code for solving linear equations is especially likely to need attention.

Another concern is how to ensure that METAPOST continues to give the same numerical results in all implementations. It is may be safe to assume 64-bit IEEE standard floating point is now available on all systems, but some compilers use a higher precision for intermediate results and it would be necessary to defeat that somehow. There is also the concern that `mp.web` is technically a Pascal program and not all installations are based on Pascal-to-C translation.

I considered all these issues when I implemented the graph package, and I decided that it was a lot easier to design the graph macros around METAPOST's `mlog` and `mexp` operators.

10 Unfill and weighted paths

The current implementation keeps track of the painting order for the basic components of a picture but does not try determine exactly which parts overlap. Unfill may be easier than allowing weighted paths with add and subtract, but implementing all these features together would require an ability to keep track of a picture as a spline-bounded planar subdivision. This would require sophisticated algorithms and data structures, and geometric algorithms of this type are famous for being extremely difficult to implement robustly. Also, it is not clear how to make fonts fit into this model.

It would be very tricky to make the new features upward compatible. In "Drawing Graphs with METAPOST" (CSTR 164), I describe a `for ... within` iteration that allows one to pick apart a picture and see all its basic components in the painting order. This is fundamental to the `graph.mp` macros and would be very hard to reconcile with new features such as add, subtract and cull.

11 Bitmaps

The request for a bitmap model like the one available in METAFONT is quite natural, provided that it is somehow generalized to handle color. A basic design would not be too hard to implement unless there is to be an operator that takes a picture and converts it into a bitmap as the PostScript interpreter would.

12 Intersections

There could certainly be a primitive that attempts to find all the intersections between two paths instead of just a single intersection point. If the operator is to take two paths, intersect their interiors, and return a path that describes this region, paths would have to be generalized to allow multiple noncontiguous outlines. One way to do this would involve a new type of path join operator that behaves syntactically like `--` or `...`, takes one unit of "time" along the path, and means that there is a gap in the path. It would also be necessary to generalize `fill` to accept non-closed paths.

13 Pens

The main advantages of the current treatment of circular and elliptical pens are that it is robust and generates simple PostScript. The main disadvantage I am aware of is that elliptical pens interact poorly with dashed lines. The present scheme also makes it hard to convert METAPOST output into a Type 1 font, but that is not METAPOST's intended application.

I think it would be a mistake to make all pen operations outline-based. Outline-based pens could be provided as an alternative, but this would not be easy to implement. Polygonal pens are easier to express in terms of outlines, but the implementation was a lot of work and it still has known bugs.

14 Specials

The METAPOST language already has a very simple `special` mechanism and any improvements will have to be upward compatible. Perhaps a different key word is needed.

15 Fonts

I agree that it would be nice to have METAPOST's primitives handle ligatures and kerning.

16 Other features

There has been some demand for other features that would not be too hard to implement.

- Make dash patterns work with polygonal pens.
- Add operators to determine whether a point is "inside" a path or what its winding number is with respect to the path. Is some point "painted" by a given picture, and if so what color?
- Add an operator for the area "inside" a path and the total area affected by a picture.
- Allow `shipout` to an arbitrary file, not just the job name with a numeric extension.
- Have a `settex` command that takes a string expression and uses it to decide what version of TEXto use for `btex` ... `etex`. It would have to occur before the first `btex` and it should be restrictive enough not to be a security risk.

Conclusion

Some of the new features I have discussed would be easy to implement, and many of the others would be nice to have if someone can find the time to implement them properly. However, I am inclined to think that it would be a mistake to do outline-based versions of elliptical pens or to add weighted paths with add and subtract. Care should be taken to make any additional features work reliably and fit in with the rest of the language.

◇ John D. Hobby
Bell Laboratories
Room 2C-458
700 Mountain Ave.
Murray Hill, NJ 07974-0636
hobby@research.bell-labs.com