MetaPost developments — autumn 2006

TACO HOEKWATER
Elvenkind, Dordrecht
taco (at) elvenkind dot com

Abstract

The new release of MetaPost includes some new features as well as a number of bug fixes. The new functionality includes: the possibility of using a template for the naming of output files; support for CMYK and greyscale color models; per-object PostScript specials; the option to generate Encapsulated PostScript files adhering to Adobe's Document Structuring Conventions; the ability to embed re-encoded and/or subsetted fonts; and support for the GNU implementation of troff (groff).

Introduction

Version 0.901 of MetaPost was released at BachoTEX 2005. It featured an updated manual and the new `mpversion` primitive, but was mostly a bug-fix release.

At that time, a new version was promised for the autumn. In hindsight, that was overly optimistic. It is now autumn 2006, and version 1.0 will finally be released by the time this article is published.

Bug fixes in version 1.0

Stability issues

In previous versions of MetaPost, the size of the memory array was not stored in the mem file, because it was assumed to be a fixed quantity. But in Web2c-based systems, the memory sizes are dynamic: the actual size that should be used by the executable can change depending on the command-line invocation and `texmf.cnf` settings.

This caused a number of bugs, for example

- disappearing specials from the output;
- incorrect error messages;
- unexplained crashes.

The required minimum memory size is now included in the memory dump file. If a mismatch occurs, an error message will be issued.

turningnumber

The previous (0.9) MetaPost executable used a very simple algorithm to implement the `turningnumber` operation: connect the dots, add up all the angles, and then divide by 360. This was a temporary patch that was added as a stop-gap measure to make the erroneous cases easier to predict. The current version includes a final fix for `turningnumber` by including new code that calculates the true curvature for path segments.

This new algorithm is based on a mailing list discussion between members of the group. It is slower, but (finally) 100% accurate.

Adobe Document Structuring Conventions

Thanks to detailed comments from Michail Vidiassov, the output will now strictly adhere to the Adobe Document Structuring Conventions for Encapsulated PostScript files — when the internal quantity `prologues` is set to 2 or higher. The special setting of `prologues` is needed for compatibility with existing MetaPost post-processing tools.

Ignored withcolor

All previous versions of MetaPost failed to recognize that the user supplied a color specification when the color consisted of three zero-valued parts, as in this example:

```
draw fullcircle withcolor black;
```

In this case, no PostScript color selection command was output at all. If any surrounding command specified a different drawing color, that color would be used instead of black.

Current color trashed after clip

Previously, a `clip` command would completely destroy the internal graphic state. As a side effect, it would force the default color of the following operation to be black even if the surrounding document specified a different text color. This is now fixed.

Table 1: Escape sequences for `filenametemplate`

%%	A percent sign
%j	The current jobname
%⟨0-9⟩c	The charcode value
%⟨0-9⟩y	The current year
%⟨0-9⟩m	The numeric month
%⟨0-9⟩d	The day of the month
%⟨0-9⟩H	The hour
%⟨0-9⟩M	The minute

New features in version 1.0

File-name templates

The first of the new features is support for output filename templates. These templates use `printf`-style escape sequences (listed above) and are re-evaluated before each `shipout`. Numeric fields may be left-padded with zeroes.

The new primitive is `filenametemplate`, and it is string-valued. Here's an example:

```
filenametemplate "%j-%3c.eps";
beginfig(1);
  draw p;
endfig;
```

If this input file is saved as `test.mp`, then the output file will be named `test-001.eps`, instead of `test.1` as in previous versions.

Here are the escape sequences:

To ensure compatibility with older files, the default value of `filenametemplate` is `%j.%c`. If it is assigned an empty string, it will revert to that default.

CMYK color model

Support is now provided for the industry-standard CMYK color model. Following is a simple example.

```
beginfig(1);
   draw fullcircle
      withcmykcolor (1,0,0,0);
endfig;
```

To make more flexible use possible, a new type of expression is introduced. A `cmykcolor` is a quartet of `numeric` values that behaves similarly to the already existing type `color`. This example is equivalent to the previous one:

```
beginfig(1);
   cmykcolor cyan;
   cyan := (1,0,0,0);
   draw fullcircle withcmykcolor cyan;
endfig;
```

The new `cyanpart`, `magentapart`, `yellowpart` and `blackpart` primitives allow access to the various pieces of a `cmykcolor` or the CMYK component of an image object.

Greyscale color model

Only two new primitives are needed for greyscale support: `withgreyscale` and `greypart`. This is because greyscale values are simple `numeric` values.

```
beginfig(1);
   faded := 0.5;
   draw fullcircle withgreyscale faded;
endfig;
```

Mark-only color model

There is also a new primitive for 'mark-only' support: `withoutcolor`. This command is convenient in cases where MetaPost is not supposed to output any explicit color commands to the PostScript file at all, as in the generation of font outlines. Example:

```
beginfig(1);
   draw fullcircle withoutcolor;
endfig;
```

Other color handling changes

A new primitive `defaultcolormodel` is introduced. This specifies the assumed color model for objects that are drawn without any color specification. In all three models that actually specify a color, the default color is black.

The primitives for the already existing RGB color model are now also available under new names: the draw option `withcolor` becomes `withrgbcolor`, and the variable type `rgbcolor` is an alias for `color`.

The existing primitive `withcolor` has been extended so that it accepts any of the five possible input syntaxes:

Actual input	*Remapped meaning*
withcolor ⟨rgbcolor⟩	withrgbcolor ⟨rgbcolor⟩
withcolor ⟨cmykcolor⟩	withcmykcolor ⟨cmykcolor⟩
withcolor ⟨numeric⟩	withgreyscale ⟨numeric⟩
withcolor ⟨false⟩	withoutcolor
withcolor ⟨true⟩	⟨nothing⟩

An image object can have only one color model. The last specification of `withcolor`, `withcmykcolor` or `withgreyscale` controls the color model for a particular object.

Object specials

The new MetaPost supports specials that can be attached to main drawing objects. These specials are output on their own lines, immediately before and after the object they are attached to.

The names of the two new drawing options are `withprescript` and `withpostscript`. Their arguments are simple strings that are output as-is. It is up to the macro writer to make sure that the generated PostScript code is correct.

```
beginfig(1);
  draw fullcircle
     withprescript "gsave"
       withpostscript "grestore";
endfig;
```

Multiple prescripts and postscripts for a single object are possible; simply repeat the command. They are placed in the output file in the same order in which they are specified.

Standalone EPS

If `prologues` is set to the value 2, MetaPost now generates a proper Encapsulated PostScript level 2 image that does not depend on `dvips` post-processing. A private PostScript dictionary will be created to reduce the output size for large images.

In this output mode, fonts are not actually embedded, but their definition will be handled correctly.

Font re-encoding

If `prologues` is set larger than 1, any used fonts are automatically re-encoded, and the encoding vector file specified in the fontmap entry will be embedded in the output file.

This code is based on the font library used by dvips and pdfTEX. Also following in the footsteps of pdfTEX, there are two new associated primitives: `fontmapfile` and `fontmapline`. Their string-valued arguments use the same optional flag as pdfTEX:

	replace the current font list completely
+	extend the font list, but ignore duplicates
=	extend the font list, replacing duplicates
−	remove all matching fonts from the font list

Here is an example:

```
prologues := 2;
fontmapfile "+ec-public-lm.map";
beginfig(1);
  draw "Helló, világ" infont "ec-lmr10";
endfig;
```

Font inclusion

Font inclusion is triggered by `prologues` being equal to 3. Whether or not actual inclusion and/or subsetting takes place is controlled by the map files. These can be controlled using the syntax explained in the previous section.

GNU groff support

The new version of MetaPost has native support for GNU groff, thanks to a set of patches from Werner Lemberg and Michail Vidiassov.

Future plans

With version 1.0 out the door, plans for the next version are being made. The next release after this one is 1.1, and it will likely have the following set of new features:

MetaPost dynamic library

It will become possible to build MetaPost as a thread-safe dynamic library as well as a static executable. This will allow easy embedding inside other programs, as well as facilitating the creation of a system-level rendering service.

The two main pieces needed for this are the creation of a MetaPost instance structure to replace the current set of global variables, and the addition of an extra layer of I/O indirection.

Better numerical precision

MetaPost currently uses fixed-point arithmetic based on 32-bit integers, with the decimal dot varying between the 16^{th} bit (for numeric values), the 20^{th} bit (for angles), and the 28^{th} bit (for Bézier fractions). The precision as well as the range of these calculations leaves something to be desired.

There is a web change file by Giuseppe Bilotta that uses 64-bit internal calculations instead, with the decimal dot at the 32^{nd}, 52^{nd}, or 60^{th} bit. While this change file does not deal with the problems MetaPost has with ranged input data, it does solve the most obvious acute precision problems.

Storable objects

We want to add the possibility of storing and retrieving named drawing objects. These objects will be stored in memory only once, and written to the output file only once.

This will greatly reduce the memory requirements and output size for certain types of figures.

Multiple linear equation systems
All linear equations in MetaPost are part of the same equation system. With every new equation this entire system has to be updated, at a significant cost in terms of running time.

Certain macro programming styles would benefit enormously from the possibility to use disjoint sets of equations simultaneously.

Some 3D support
The ability to create and use 12-part transformations and perhaps some other operations on triplets should make it easier for macro packages to implement three-dimensional drawings.

It seems unlikely at this point that there will be true three-dimensional paths. We certainly do not object to such an extension, but both the expertise and available time to do this are sorely lacking within the current MetaPost development team.

Where to find MetaPost

- Web home page and portal:
`http://www.tug.org/metapost`
- User mailing list:
`http://www.tug.org/mailman/listinfo/metapost`
- Development & sources:
`https://foundry.supelec.fr/projects/metapost`