MetaPost: PNG Output

Abstract
The latest version of Metapost (1.80x) has a third output backend: it is now possible to generate PNG bitmaps directly from within Metapost.

Introduction

For one of my presentations at EuroTEX2012 in Breskens, I wanted to create an animation in order to demonstrate a Metapost macro that uses timer variables to progress through a scene.

While working on that presentation, it quickly became obvious that the 'traditional' method of creating an animation with Metapost by using ImageMagick's `convert` to turn EPS images into PNG images was very time consuming. So much so, that I actually managed to write a new backend for Metapost while waiting for ImageMagick to complete the conversion.

Simple usage

Metapost will create a PNG image (instead of Encapsulated PostScript or Scalable Vector Graphics) by setting `outputformat` to the string `png`:

```
outputformat := "png";
outputtemplate := "%j-%c.%o";
beginfig(1);
fill fullcircle scaled 100 withcolor red;
endfig;
end.
```

This input generates a bitmap file with dimensions 100 x 100 pixels, with 8-bit/color RGBA. It shows a red dot on a transparent background.

Adjusting the bitmap size

In the simple example given above, Metapost has used the default conversion ratio where one point equals one pixel. This is not always desired, and it is tedious to have to scale the picture whenever a different output size is required.

To allow easy modification of the bitmap size independent of the actual graphic, two new internal parameters have been added: `hppp` and `vppp` (the names come from Metafont, but the actual meaning is specific to Metapost).

In Metapost, 'hppp' stands for 'horizontal points per pixel', and similarly for 'vppp'. Adding

```
hppp := 2.0;
```

to the example above changes the bitmap into 50 x 100 pixels. Specifying values less than 1.0 (but above zero!) makes the bitmap larger.

Adjusting the output options

Metapost creates a 32-bit RGBA bitmap image, unless the user alters the value of another new internal parameter: `outputformatoptions`.

The syntax for `outputformatoptions` is a space-separated list of settings. Individual settings are *keyword* + = + *value*. The only currently allowed ones are:

```
format=[rgba|rgb|graya|gray]
antialias=[none|fast|good|best]
```

No spaces are allowed on the left, nor on the right, of the equals sign inside a setting.

The assignment that would match the compiled-in default setup is:

```
outputformatoptions := "format=rgba antialias=fast";
```

however, `outputformatoptions` is initially the empty string, because that makes it easier to test whether a user-driven change has already been made.

Some notes on the different PNG output formats:

- The `rgb` and `gray` subformats have a white background. The `rgba` and `graya` subformats have a transparent background.
- The bitdepth is always 8 bits per pixel component.
- In all cases, the current picture is initially created in 8-bit RGB mode. For the `gray` and `graya` subformats, the RGB colors are reduced just before the actual PNG file is written, using a standard rule:

$$g = 0.2126 * r + 0.7152 * g + 0.0722 * b$$

- CMYK colors are always converted to RGB during generation of the output image using:

$$r = 1 - (c + k > 1?1 : c + k);$$
$$g = 1 - (m + k > 1?1 : m + k);$$
$$b = 1 - (y + k > 1?1 : y + k);$$

If you care about color conversions, you should be doing a `within <pic>` loop inside `extra_endfig`. The built-in conversions are more of a fallback.

What you should also know

Metapost uses cairo (http://cairographics.org) to do the bitmap creation, and then uses libpng (http://www.libpng.org) to create the actual file.

Any `prologues` setting is always ignored: the internal equivalent of the `glyph of` operator is used to draw characters onto the bitmap directly.

If there are points in the current picture with negative coordinates, then the whole picture is shifted upwards to prevent things from falling outside the generated bitmap.

Taco Hoekwater