MetaPost: PNG output

Taco Hoekwater

Abstract

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

1 Introduction

For one of my presentations at EuroTEX 2012 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 managed to write a new backend for MetaPost while waiting for ImageMagick to complete the conversion.

2 Simple usage

MetaPost will create a PNG image (instead of EPS or SVG) 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 RGBA color. It shows a red dot on a transparent background.

3 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 meaning is specific to MetaPost).

In MetaPost, 'hppp' stands for 'horizontal points per pixel'; similarly for 'vppp'. Adding '`hppp:=2.0;`' to the example above changes the bitmap to be 50 x 100 pixels. Specifying values less than 1.0 (but above zero!) makes the bitmap larger.

4 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 use ⟨*keyword*⟩=⟨*value*⟩ syntax. Currently supported are:

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

No spaces are allowed on either side of the equals sign inside a setting.

The compiled-in default could be given as:

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

However, the `outputformatoptions` variable value itself 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 bit depth 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:
  $$gray = 0.2126 * r + 0.7152 * g + 0.0722 * b$$
- 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 conversion, you should do a `within` ⟨*pic*⟩ loop inside `extra_endfig`. The built-in conversions are intended as a fallback.

5 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
`http://tug.org/metapost`