colorspace

Version 1.1.0

Javier Bezos
`http://www.tex-tipografia.com`

2015-06-01

This package is built on the previous attempts to provide spot colors and other additional features by Jens Elstner, Stephan Lehmke and Siep Kroonenberg (with some inspiration from ConTeXt, too). It requires xcolor, which is loaded if it has not been before.

It provides a clean user interface, with a single command for defining a spot color. It seems to work with tikz. Currently only pdftex and luatex are supported.

Mixing spot colors (with process colors or other spots colors) is supported to some extent (see below). You can also assign ICC profiles to the default CMYK, RGB and Gray spaces. Other functions related to the PDF color spaces (indexed, calibrated, Lab spaces) are not yet suported, but they are under study. They apply to text and line art only, not external images. Declarations are global. Using `\selectcolormodel` does not work yet.

Those docs, like the package itself, are still somewhat tentative (as you very likely have noticed), but the basic behaviour described here will be preserved in future versions.

For transparencies, see transparent, by Heiko Oberdiek.

1 Spot colors

Write, for example:

```
\definespotcolor{foo}{BarTone 555 GN}{.3,.4,.5,.6}
```

That's all. Here `foo` is the LaTeX name, as used in `\color` and the like, `BarTone 555 GN` is the PDF name (multiple spaces are collapsed into one) as shown by PDF readers, and the four numbers are the CMYK equivalent. LaTeX knows nothing about the PDF name, which is just a string to be written to the generated file, while the PDF knows nothing about the LaTeX name.

You can use tints as usual in xcolor, like:

```
\color{foo!60}
\colorlet{foo60}{foo!60}
```

and even set tints from other tints. To mix inks, see below.

The special PDF names `All` (for all plates) and `None` work as expected:

```
\definespotcolor{registration}{All}{1,1,1,1}
```

Internally, only CMYK is used, but you can define the equivalent color with another name space, which is then converted:

```
\definespotcolor{foo}{BarTone 555 GN}[rgb]{.5, .4, .3}
```

To change the color space for a page and the subsequent ones, you can set something like:

```
\pagecolorspace{name1,name2,name3}
```

(It can be empty.) To return to the default color space, which contains all the defined spot colors, use `\resetpagecolorspace`. Use this macro with care, because of the asynchronous nature of TEX – remember it affects the whole current page.

2 Mixing spot colors

To mix spot colors you must first declare a color space (or model) including them. This is done with something like:

```
\definecolorspace{name}{mixed}{color1,color2,color3}
```

(The second argument is the type of color space.) For example, if we have two spot colors named `spot1` and `spot2`, and we want in addition yellow:

```
\definecolorspace{spot12y}{mixed}{spot1,spot2,yellow}
```

A typical usage, for shades, would be:

```
\definecolorspace{shaded1}{mixed}{spot1,black}
```

Due to internal limitations of xcolor, no more than four colors are allowed. The alternate color space in the PDF file is that of the spot colors (which means currently it is CMYK).

Then, you can define a color with:

```
\definecolor{mix12y}{spot12y}{.5,.4,.6}
\definecolor{sh1}{shaded1}{.6,.3}
```

or set it with

```
\color[spot12y]{.5,.4,.3}
\color[shaded1]{.6,.3}
```

As in spot colors, the only operation allowed is `!` for tints (ie, `color!num`). But there is an easy trick to mix colors with `!` and `color,num` – just define an ortogonal set of colors based on the new color model:

```
\definecolor{xspot1}{spot12y}{1,0,0}
\definecolor{xspot2}{spot12y}{0,1,0}
\definecolor{xyellow}{spot12y}{0,0,1}
```

and then you can say:

```
\color{xspot1!30!xspot2!40!xyellow}
\color{spot12y:xspot1,3;xspot2,2;xyellow,1}
```

Of course, it is just a trick and a better and direct interface is under study.

Color series are also partially supported. For example:

```
\definecolorseries{test}{spot12y}{grad}[spot12y]{.95,.85,.55}{3,11,17}
\definecolorseries{test}{spot12y}{last}{xyellow!50}{xspotA}
```

(The key is not to mix the new model with other color models.)

3 ICC Based spaces

The starred version `\definecolorspace*` does not define a new color model, but sets the behaviour of the three basic color spaces (`cmyk`, `rgb` and `gray`). When belonging to the same space, the last one for that space takes precedence. It cannot be used to define new colors or set them. Currently, only a type is supported – `iccbased`. For example,

```
\definecolorspace*{sRGB}{iccbased}{sRGB Profile.icc}
```

The space it applies to is read from the ICC profile. The name can be used in `\pagecolorspace` (and must, if you want it to be active). Alternatively, there are 3 reserved names: `*rgb`, `*gray`, `*cmyk`, which stand for the current default spaces. The former are not set by `\resetpagecolorspace`, but the starred named are.

Note those ICC spaces does not go to the output intent dictionary (see the `pdfx` package). The latter, as the PDF reference explains, supplements rather than replaces the ICC profiles in a default color space.

4 Overprinting

This is usually a pre-print task, but by setting it in the document you will get a better idea of how the colors are actually overlapped. However, remember the effect produced is device-dependent, and colorant overprint decisions should be made at output time (according to the PDF reference).

Very often, it is set for the whole document with the package options `knockout` (no overprint), and `overprint`. By default, the overprint mode is 1, but it can be changed with `opm=0`.

Once set the overprint state for the whole document, you can use something like:

```
{\overprintstate{1}text}
\textoverprint[1]{text}
```

(or `0`, or `no`; default in `\textoverprint` is `1`, except with the package option `opm=0`).

Since the color stack is used, pdfTEX $\geq$ 1.40 is required.