colorspace

Version 1.1.1

Javier Bezos
http://www.texnia.com

2016-09-12

The aim of this package is, as the name implies, to provide tools for PDF color spaces. It requires xcolor, which is loaded if it has not been before. It seems to work with tikz.

Currently it supports what I think are the most common tools:

- Spot colors, with a clean user interface, and including tints (with the `!` notation).
- Proper switching of color spaces.
- Mixed spot and process colors (up to 4), like shades (ie, a spot color with black).
- ICC based default CMYK, RGB and Gray spaces.
- Overprinting (across pages, using the color stack).

Currently only pdftex and luatex are supported. Support for xetex is on the 'to do' list, but due to the limitations of this engine this task is somewhat challenging and I'm not sure all features will be implemented.

Other functions related to the PDF color spaces (indexed, calibrated, Lab spaces) are not yet suported, but they are under study. Calibrated colors, although not directly supported, can be defined with an ICC profile created with LPROF[1] and then assigned to a default space as described below.

They apply to text and line art only, not external images. For the latter, graphicx provides a plea of (undocumented) transformations: `interpolate`, `decodearray`, `maskarray`, `intent`, `ocobjnum`, and `ocobjref`. For transparencies, see transparent, by Heiko Oberdiek.

This package is still evolving, but the basic behaviour described here will be preserved in future versions. However, some functions from xcolor might not work yet (for example `\selectcolormodel`).

Declarations are global and should go in the preamble.

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).

[1] `http://lprof.sourceforge.net/`

It does not provide a list of Pantone, TrueMatch, Folcoltone, Toyo, etc., colors. Currently you can find quite easily CMYK equivalents on the web, and after all they are intended to be used with a "physical" palette (and not, as sometimes done, just picking a color just because looks nice on screen).

1 Spot colors

`\definespotcolor{`⟨*latex-name*⟩`}{`⟨*PDF-name*⟩`}{`⟨*CMYK-equivalent*⟩`}`

Write, for example:

```
\definespotcolor{foo}{BarTone 555 GN}{.8,.2,.5,.3}
```

for .

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!50}
\colorlet{foo60}{foo!50}
```

which would produce , 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 for the equivalent color, but you can define the latter with another name space, which is then converted:

```
\definespotcolor{foob}{BarTone 666 GN}[rgb]{.8, .2, .4}
```

which yields . Remember as far PDF is concerned a spot color is a color space on its own.

2 Mixing spot colors

`\definecolorspace{`⟨*latex-name*⟩`}{mixed}{`⟨*color-list*⟩`}`

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{fooshaded}{mixed}{foo,black}
```

Then, you can define a color with:

```
\definecolor{mix12y}{spot12y}{.5,.4,.6}
\definecolor{darkfoo}{fooshaded}{.6,.3}
```

or set it with

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

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).

As with 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 (none of those provided by xcolor fits really with the new `mixed` models).

Color series (see the xcolor documentation) are also partially supported. For example:

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

Here is a example with some of the described techniques, based on the `fooshaded` space defined above (figure 1 shows the `foo` plate):

```
\definecolor{sfoo}{fooshaded}{1,0}
\definecolor{sblack}{fooshaded}{0,1}
\definecolorseries{shseries}{fooshaded}{last}{sfoo!40}{sblack}
```

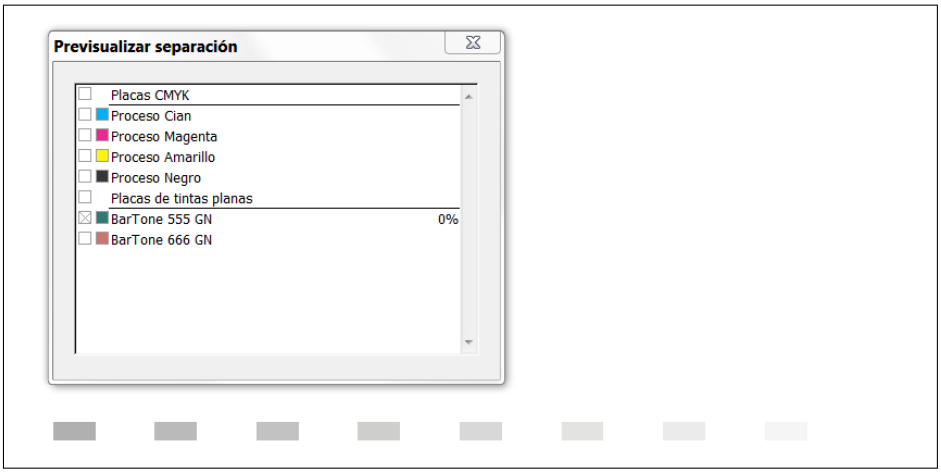

Figure 1: Plate for the `foo` spot color as shown by Adobe Acrobat. Note the `foob` spot color defined in this document is listed, too.

```
\def\testclr#1{{\fboxsep0pt\fbox{\colorbox{#1}{\phantom{XX}}}}}
\resetcolorseries[8]{shseries}
\begin{tabular}{cccccccc}
0 \testclr{shseries!!+} &
1 \testclr{shseries!!+} &
2 \testclr{shseries!!+} &
3 \testclr{shseries!!+} &
4 \testclr{shseries!!+} &
5 \testclr{shseries!!+} &
6 \testclr{shseries!!+} &
7 \testclr{shseries!!+}
\end{tabular}
```

0 1 2 3 4 5 6 7

The key is not to mix a `mixed` model with other color models.

3 Page color spaces

Each PDF page must know which colors will be used (other than the predefined CMYK, RGB and Gray). By default, colorspace turn on for every page all newly defined colors, and that will be fine in most cases. However, you may want to set explicitly the list. Use this feature with care, because (1) the asynchronous nature of TEX (remember it affects the whole current page), and (2) each distinct color list creates a PDF resource.

`\pagecolorspace{`⟨*color-list*⟩`}`
`\resetpagecolorspace`

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`.

4 ICC Based spaces

`\definecolorspace*{`⟨*latex-name*⟩`}{iccbased}{`⟨*icc-file*⟩`}`

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 definition for that space takes precedence and its considered the default one. 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.

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.

The name can be used in `\pagecolorspace`. Alternatively, there are 3 reserved names: `*rgb`, `*gray`, `*cmyk`, which stand for the current default spaces. Named ICC based spaces are not set by `\resetpagecolorspace`, but the starred named are. On the other hand, the starred names are not set automatically by `\pagecolorspace`, and you must set them explicitly if you want them to be active.

5 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 in soft proofing. 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.

6 Version

1.1.1. No new features. Just internal changes related to LUATEX and new manual.