The l3color package
Experimental color support

The LaTeX3 Project*

Released 2020-09-24

1 Color models

A color *model* is a way to represent sets of colors. Different models are particularly suitable for different output methods, *e.g.* screen or print. Parameter-based models can describe a very large number of unique colors, and have a varying number of *axes* which define a color space. In contrast, various proprietary models are available which define *spot* colors (more formally separations).

Core models are used to pass color information to output; these are "native" to l3color. Core models use real numbers in the range $[0, 1]$ to represent values. The core models supported here are

- `gray` Grayscale color, with a single axis running from 0 (fully black) to 1 (fully white)
- `rgb` Red-green-blue color, with three axes, one for each of the components
- `cmyk` Cyan-magenta-yellow-black color, with four axes, one for each of the components

There are also interface models: these are convenient for users but have to be manipulated before storing/passing to the backend. Interface models are primarily integer-based: see below for more detail. The supported interface models are

- `Gray` Grayscale color, with a single axis running from 0 (fully black) to 15 (fully white)
- `hsb` Hue-saturation-brightness color, with three axes,all real values in the range $[0, 1]$ for hue saturation and brightness
- `Hsb` Hue-saturation-brightness color, with three axes, integer in the range $[0, 360]$ for hue, real values in the range $[0, 1]$ for saturation and brightness
- `HSB` Hue-saturation-brightness color, with three axes, integers in the range $[0, 240]$ for hue, saturation and brightness
- `HTML` HTML format representation of RGB color given as a single six-digit hexadecimal number

*E-mail: latex-team@latex-project.org

- `RGB` Red-green-blue color, with three axes, one for each of the components, values as integers from 0 to 255
- `wave` Light wavelength, a real number in the range 380 to 780 (nanometres)

All interface models are internally stored as `rgb`.

To allow parsing of data from xcolor, any leading model up the first `:` will be discarded; the approach of selecting an internal form for data is *not* used in l3color.

Additional models may be created to allow mixing of separation colors with each other or with those from other models. See Section 8 for more detail of color support for additional models.

When color is selected by model, the ⟨*values*⟩ given are specified as a comma-separated list. The length of the list will therefore be determined by the detail of the model involved.

Color models (and interconversion) are complex, and more details are given in the manual to the LaTeX 2ε xcolor package and in the *PostScript Language Reference Manual*, published by Addison–Wesley.

2 Color expressions

In addition to allowing specification of color by model and values, l3color also supports color expressions. These are created by combining one or more color names, with the amount of each specified as a percentage. The latter is given between `!` symbols in the expression. Thus for example

```
red!50!green
```

is a mixture of 50 % red and 50 % green. A trailing percentage is interpreted as implicitly followed by `white`, and so

```
red!25
```

specifies 25 % red mixed with 75 % white.

Where the models for the mixed colors are different, the model of the first color is used. Thus

```
red!50!cyan
```

will result in a color specification using the `rgb` model, made up of 50 % red and 50 % of cyan *expressed in* `rgb`. As color model interconversion is not exact.

The one exception to the above is where the first model in an expression is `gray`. In this case, the order of mixing is "swapped" internally, so that for example

```
black!50!red
```

has the same result as

```
red!50!black
```

(the predefined colors `black` and `white` use the `gray` model).

Where more than two colors are mixed in an expression, evaluation takes place in a stepwise fashion. Thus in

```
cyan!50!magenta!10!yellow
```

the sub-expression

```
cyan!50!magenta
```

is first evaluated to give an intermediate color specification, before the second step

```
<intermediate>!10!yellow
```

where `<intermediate>` represents this transitory calculated value.

Within a color expression, `.` may be used to represent the color active for typesetting (the current color). This allows for example

```
.!50
```

to mean a mixture of 50 % of current color with white.

(Color expressions supported here are a subset of those provided by the LaTeX 2$_\varepsilon$ xcolor package. At present, only such features as are clearly useful have been added here.)

3 Named colors

Color names are stored in a single namespace, which makes them accessible as part of color expressions. Whilst they are not reserved in a technical sense, the names `black`, `white`, `red`, `green`, `blue`, `cyan`, `magenta` and `yellow` have special meaning and should not be redefined. Color names should be made up of letters, numbers and spaces only: other characters are reserved for use in color expressions. In particular, `.` represents the current color at the start of a color expression.

`\color_set:nn` `\color_set:nn {⟨name⟩} {⟨color expression⟩}`

Evaluates the ⟨*color expression*⟩ and stores the resulting color specification as the ⟨*name*⟩.

`\color_set:nnn` `\color_set:nnn {⟨name⟩} {⟨model(s)⟩} {⟨value(s)⟩}`

Stores the color specification equivalent to the ⟨*model(s)*⟩ and ⟨*values*⟩ as the ⟨*name*⟩.

`\color_set_eq:nn` `\color_set_eq:nn {⟨name1⟩} {⟨name2⟩}`

Copies the color specification in ⟨*name2*⟩ to ⟨*name1*⟩. The special name `.` may be used to represent the current color, allowing it to be saved to a name.

`\color_show:n` `\color_show:n {⟨name⟩}`

Displays the color specification stored in the ⟨*name*⟩ on the terminal.

4 Selecting colors

General selection of color is safe when split across pages: a stack is used to ensure that the correct color is re-selected on the new page.

`\color_select:n` `\color_select:n {⟨color expression⟩}`

Parses the ⟨*color expression*⟩ and then activates the resulting color specification for typeset material.

\color_select:nn `\color_select:nn {⟨model(s)⟩} {⟨value(s)⟩}`

Activates the color specification equivalent to the ⟨*model(s)*⟩ and ⟨*value(s)*⟩ for typeset material.

\l_color_fixed_model_tl When this is set to a non-empty value, colors will be converted to the specified model when they are selected. Note that included images and similar are not influenced by this setting.

5 Colors for drawing: fills and strokes

Colors for drawing operations and so forth are split into strokes and fills (the latter may also be referred to as non-stroke color). These operations *may* apply to text, but this is backend-depended and the general `\color_select:n(n)` should be used for typeset text. In contrast to general color, these operations are *not* stacked and thus must *not* split across pages. Also, an appropriate scope must be applied to the color change, for example `\draw_begin:`/`\draw_end:`.

\color_fill:n
\color_stroke:n `\color_fill:n {⟨color expression⟩}`

Parses the ⟨*color expression*⟩ and then activates the resulting color specification for filling or stroking.

\color_fill:nn
\color_stroke:nn `\color_fill:nn {⟨model(s)⟩} {⟨value(s)⟩}`

Activates the color specification equivalent to the ⟨*model(s)*⟩ and ⟨*value(s)*⟩ for filling or stroking.

color.fc
color.sc When using `dvips`, these PostScript variables hold the fill and stroke color, respectively.

6 Multiple color models

When selecting or setting a color with an explicit model, it is possible to give values for more than one model at one time. This is particularly useful where automated conversion between models does not give the desired outcome. To do this, the list of models and list of values are both subdivided using `/` characters (as for the similar function in xcolor). For example, to save a color with explicit `cmyk` and `rgb` values, one could use

```
\color_set:nnn { foo } { cmyk / rgb }
  { 0.1 , 0.2 , 0.3 , 0.4 / 0.1, 0.2 , 0.3 }
```

The manually-specified conversion will be used in preference to automated calculation whenever the model(s) listed are used: both in expressions and when a fixed model is active.

Similarly, the same syntax can be applied to directly selecting a color.

```
\color_select:nn { cmyk / rgb }
  { 0.1 , 0.2 , 0.3 , 0.4 / 0.1, 0.2 , 0.3 }
```

Again, this list is used when a fixed model is active: the first entry is used unless there is a fixed model matching one of the other entries.

7 Exporting color specifications

The major use of color expressions is in setting typesetting output, but there are other places in which some form of color information is required. These may need data in a different format or using a different model to the internal representation. Thus a set of functions are available to export colors in different formats.

Valid export targets are

- `backend` Two brace groups: the first containing the model, the second containing space-separated values appropriate for the model; this is the format required by backend functions of expl3
- `HTML` Uppercase two-digit hexadecimal values, expressing a red-green-blue color; the digits are *not* separated
- `space-sep-cmyk` Space-separated cyan-magenta-yellow-black values
- `space-sep-rgb` Space-separated red-green-blue values suitable for use as a PDF annotation color

`\color_export:nnN` `\color_export:nnN {⟨color expression⟩} {⟨format⟩} {⟨tl⟩}`

Parses the ⟨*color expression*⟩ as described earlier, then converts to the ⟨*format*⟩ specified and assigns the data to the ⟨*tl*⟩.

`\color_export:nnnN` `\color_export:nnnN {⟨model⟩} {⟨value(s)⟩} {⟨format⟩} {⟨tl⟩}`

Expresses the combination of ⟨*model*⟩ and ⟨*value(s)*⟩ in an internal representation, then converts to the ⟨*format*⟩ specified and assigns the data to the ⟨*tl*⟩.

8 Creating new color models

Additional color models are required to support specialist workflows, for example those involving separations (see https://helpx.adobe.com/indesign/using/spot-process-colors.html for details of the use of separations in print). Color models may be split into families; for the standard device-based color models (`DeviceCMYK`, `DeviceRGB`, `DeviceGray`), these are synonymous. This is not generally the case: see the PDF reference for more details. (Note that l3color uses the shorter names `cmyk`, etc.)

`\color_model_new:nnn` `\color_model_new:nnn {⟨model⟩} {⟨family⟩} {⟨params⟩}`

Creates a new ⟨*model*⟩ which is derived from the color model ⟨*family*⟩. The latter should be one of

- `DeviceN`
- `Separation`

(The ⟨*family*⟩ may be given in mixed case as-in the PDF reference: internally, case of these strings is folded.) Depending on the ⟨*family*⟩, one or more ⟨*params*⟩ are mandatory or optional.

For a `Separation` space, there are three *compulsory* keys.

- `name` The name of the Separation, for example the formal name of a spot color ink. Such a ⟨*name*⟩ may contain spaces, etc., which are not permitted in the ⟨*model*⟩.
- `alternative-model` An alternative device colorspace, one of `cmyk`, `rgb`, `gray` or `CIELAB`. The three parameter-based models work as described above; see below for details of CIELAB colors.
- `alternative-values` A comma-separated list of values appropriate to the `alternative-model`. This information is used by the PDF application if the `Separation` is not available.

CIELAB color separations are created using the `alternative-model = CIELAB` setting. These colors must also have an `illuminant` key, one of `a`, `c`, `e`, `d50`, `d55`, `d65` or `d75`. The `alternative-values` in this case are the three parameters $L*$, $a*$ and $b*$ of the CIELAB model. Full details of this device-independent color approach are given in the documentation to the colorspace package.

CIELAB colors *cannot* be converted into other device-dependent color spaces, and as such, mixing can only occur if colors set up using the CIELAB model are also given with an alternative parameter-based model. If that is not the case, l3color will fallback to using black as the colorant in any mixing.

For a `DeviceN` space, there is one *compulsory* key.

- `names` The names of the components of the `DeviceN` space. Each should be either the ⟨*name*⟩ of a `Separation` model, a process color name (`cyan`, etc.) or the special name `none`.

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.