Extending \LaTeX{}’s color facilities: the \texttt{xcolor} package

Dr. Uwe Kern

v3.01 (2023/11/15) *

Abstract

\texttt{xcolor} provides easy driver-independent access to several kinds of colors, tints, shades, tones, and mixes of arbitrary colors by means of color expressions like \texttt{\color{red!50!green!20!blue}}. It allows to select a document-wide target color model and offers tools for automatic color schemes, conversion between twelve color models, alternating table row colors, color blending and masking, color separation, and color wheel calculations.

Contents

1 Introduction \hfill 4
  1.1 Purpose of this package \hfill 4
  1.2 Color tints, shades, tones, and complements \hfill 5
  1.3 Color models \hfill 5
  1.4 Color wheels and color harmony \hfill 5

2 The User Interface \hfill 7
  2.1 Preparation \hfill 7
    2.1.1 Package installation \hfill 7
    2.1.2 Package options \hfill 7
    2.1.3 Executing additional initialisation commands \hfill 8
  2.2 Color models \hfill 8
    2.2.1 Supported color models \hfill 8
    2.2.2 Substituting individual color models \hfill 11
    2.2.3 Changing the target color model within a document \hfill 11
  2.3 Arguments and terminology \hfill 12
    2.3.1 Additional remarks and restrictions on arguments \hfill 12
    2.3.2 Meaning of standard color expressions \hfill 15
    2.3.3 Meaning of extended color expressions \hfill 16
    2.3.4 Color functions \hfill 16
  2.4 Predefined colors \hfill 17
    2.4.1 Colors that are always available \hfill 17

*This package can be downloaded from \texttt{CTAN/macros/latex/contrib/xcolor/}. There is also an \texttt{xcolor} homepage: \texttt{www.ukern.de/tex/xcolor.html}. Please send error reports and suggestions for improvements to: \texttt{https://github.com/latex3/xcolor/issues}.
6.3.2 The cmy model ............................................. 50
6.3.3 The cmyk model .......................................... 51
6.3.4 The hsb model ............................................ 51
6.3.5 The Hsb model ............................................ 53
6.3.6 The tHsb model ........................................... 53
6.3.7 The gray model ............................................ 53
6.3.8 The RGB model ........................................... 54
6.3.9 The HTML model ......................................... 54
6.3.10 The HSB model ........................................... 54
6.3.11 The Gray model .......................................... 54
6.3.12 The wave model ......................................... 55

References .......................... 57

Appendix .................................. 58
Acknowledgement ................................ 58
Trademarks .................................. 58
Known Issues .................................. 58
History ...................................... 58

Index ........................................ 64

List of Tables

1 Package options ............................................. 9
2 Package loading order ..................................... 10
3 Supported color models .................................... 10
4 Arguments and terminology .............................. 13
5 Drivers and color models ................................... 41
6 Driver-dependent internal color representation ....... 43
7 Color constants ............................................ 45
8 Color conversion pairs .................................... 45

List of Figures

1 Color spectrum ............................................. 31
2 Color testing ................................................ 31
3 Progressing from one to another color ................. 32
4 Target color model .......................................... 33
5 Standard color expressions ................................ 33
6 Standard color expressions ................................ 33
7 Current color .............................................. 33
8 Color series ................................................ 34
9 Color masking .............................................. 35
10 Alternating row colors in tables: \rowcolors vs. \rowcolors* . . 35
11 Hsb and tHsb: hue° in 15° steps .......................... 36
12 Color harmony ............................................ 37
1 Introduction

1.1 Purpose of this package

The color package provides a powerful and stable tool for handling colors within (pdf)\LaTeX in a consistent and driver-independent way, supporting several color models (slightly less driver-independent).

Nevertheless, it is sometimes a bit clumsy to use, especially in cases where slight color variations, color mixes or color conversions are involved: this usually implies the usage of another program that calculates the necessary parameters, which are then copied into a \texttt{\definecolor} command in \LaTeX. Quite often, also a pocket calculator is involved in the treatment of issues like the following:

- My company has defined a corporate color, and the printing office tells me how expensive it is to use more than two colors in our new brochure, whereas all kinds of tints (e.g., a 75\% version) of our color can be used at no extra cost. But how to access these color variations in \LaTeX? (Answer: \texttt{\color{CorporateColor!75}} etc.)

- My friend uses a nice color which I would like to apply in my own documents; unfortunately, it is defined in the \texttt{hsb} model which is not supported in my favorite application pdf\LaTeX. What to do now? (Answer: just use the \texttt{hsb} definitions, \texttt{xcolor} will do the necessary calculations)

- How does a mixture of 40\% green and 60\% yellow look like? (Answer: 40\% \includegraphics[width=1cm]{green} + 60\% \includegraphics[width=1cm]{yellow} = \includegraphics[width=1cm]{greenyellow}, e.g., \texttt{\color{green!40!yellow}})

- And how does its complementary color look like? (Answer: \includegraphics[width=1cm]{greenyellow} accessible via \texttt{\color{-green!40!yellow}})

- Now I want to mix three parts of the last color with two parts of its complement and one part of red. How does that look? (Answer: $3 \times \includegraphics[width=1cm]{greenyellow} + 2 \times \includegraphics[width=1cm]{green!40!yellow} + 1 \times \includegraphics[width=1cm]{red} = \includegraphics[width=1cm]{mixcolor}$, the last color being accessible via \texttt{\color{rgb:-green!40!yellow,3;green!40!yellow,2;red,1}})

- I know that light waves of 485nm are within the visible range. But which color do they represent? (Answer: approximately \includegraphics[width=1cm]{greenyellow}, via \texttt{\color{wave}{485}})

- My printing office wants all color definitions in my document to be transformed into the \texttt{cmyk} model. How can I do the calculations efficiently? (Answer: \texttt{\usepackage[cmyk]{xcolor}} or \texttt{\selectcolormodel{cmyk}})

- I have a table with 50 rows. How can I get alternating colors for entire rows without copying 50 \texttt{\rowcolor} commands? The alternating scheme should start in the 3rd row. (Answer: something like \texttt{\rowcolors{3}{OddColor}{EvenColor}})

These are some of the issues solved by the \texttt{xcolor} package. Its purpose can be summarized as to maintain the characteristics of \texttt{color}, while providing additional features and flexibility with (hopefully) easy-to-use interfaces.
1.2 Color tints, shades, tones, and complements

According to [15] we define the terms

- **tint**: a color with white added,
- **shade**: a color with black added,
- **tone**: a color with gray added.

These are special cases of a general function \( \text{mix}(C, C', p) \) which constructs a new color, consisting of \( p \) parts of color \( C \) and \( 1 - p \) parts of color \( C' \), where \( 0 \leq p \leq 1 \). Thus, we set

\[
\begin{align*}
\text{tint}(C, p) & := \text{mix}(C, \text{white}, p) \\
\text{shade}(C, p) & := \text{mix}(C, \text{black}, p) \\
\text{tone}(C, p) & := \text{mix}(C, \text{gray}, p)
\end{align*}
\]

where \text{white}, \text{black}, and \text{gray} are model-specific constants, see table 7 on page 45.

Further we define the term

- **complement**: a color \( C^* \) that yields white if superposed with the original color \( C \),

knowing that there are also different concepts of complementarity in place (e.g., opposite colors on color wheels). See section 6.3 on page 47 for calculation details and section 1.4 for some remarks on color wheels.

1.3 Color models

A color model is a tool to describe or represent a certain set of colors in a way that is suitable for the desired target device, e.g., a screen or a printer. There are proprietary models (like Pantone or HKS) that provide finite sets of colors (often called spot colors), where the user has to choose from without caring about parametrisations; on the other hand, there are parameter-driven models like \text{gray}, \text{rgb}, and \text{cmyk}, that aim to represent large finite or even (theoretically) infinite sets of colors, built on very small subsets of base colors and rules, how to construct other colors from these base colors. For example, a large range of colors can be constructed by linear combinations of the base colors red, green, and blue. On the other hand, usually spot colors can only be approximated by parameter values in models like \text{cmyk} or \text{rgb}: the original colors are being physically mixed even dependent on the targeted kind of paper. Finally, there are certain colors like gold and silver that are hardly reproducible by any parameter-driven color model on standard ink or laser printers.

1.4 Color wheels and color harmony

There has been a long history of placing colors (hues) on wheels in order to discuss theoretical or practical color issues (e.g., Isaac Newton, Johann Wolfgang von Goethe). One reason may be that the circle itself is a natural tool to demonstrate common relationships as well as opposite properties.

Nowadays, there exists some confusion about notions, since both major areas related to color — art and graphics design on one hand, scientific color theory
on the other hand — tend to use the same words to describe color properties, however meaning sometimes very different facts! Thus, the appearance of color wheels differs as well as the meaning of concepts like ‘primary’ or ‘complementary’ colors.

Typical color wheel construction First, three primary colors are placed at $0^\circ$, $120^\circ$, $240^\circ$ (artists often choose red, yellow, blue, whereas color scientists might prefer red, green, blue). Next, three secondary colors are put at $60^\circ$, $180^\circ$, $300^\circ$. Then, six tertiary colors may be placed in between ($30^\circ$, $90^\circ$, ...). This is why color wheels are frequently described to consist of twelve equidistant colors, although the algorithm can be repeated arbitrarily often.

Color harmonies derived from a wheel We start with an arbitrary color wheel:

- **complementary colors** have a distance of $180^\circ$ on the wheel,
- **color triads** consist of three colors in $120^\circ$ steps,
- **color tetrads** consist of four colors in $90^\circ$ steps.

We assume now that the wheel is partitioned into $2n$ sectors of equal size:

- **split complementary colors** of a given color are the two immediate neighbours of its complementary color, characterised by the positions $\frac{n+1}{2n} \cdot 360^\circ$,
- **analogous colors** to a given color are its two or four neighbours, characterised by the positions $\pm \frac{1}{2n} \cdot 360^\circ$ and $\pm \frac{2}{2n} \cdot 360^\circ$.

Given these methods to generate ‘color harmonies’, we conclude that the results depend heavily on how we constructed the wheel! Moreover, the choice of $n$ will affect the visual results as well. Examples are shown in figure 12 on page 37.
2 The User Interface

2.1 Preparation

2.1.1 Package installation

First of all, put \texttt{xcolor.sty} and all the .def files to some place where (pdf)\LaTeX finds them. A typical place according to the \TeX Directory Structure (TDS) would be the directory \texttt{texmf/tex/latex/xcolor}, where \texttt{texmf} denotes the main directory of your \TeX installation. Additionally, put \texttt{xcolor.pro} to a place where \texttt{dvips} finds it, typically \texttt{texmf/dvips/xcolor}. Usually, you will have to run some kind of filename database update in order to make the files known and quickly searchable to the \TeX system. Then simply use \texttt{xcolor} (instead of color) in your document. Thus, the general command is \texttt{\usepackage\[\langle options\rangle\]{xcolor}} in the document preamble. Table 2 on page 10 shows what has to be taken into account with respect to the package loading order.

2.1.2 Package options

In general, there are several types of options:

- options that determine the color driver as explained in \cite{5} and \cite{6}, currently: \texttt{dvips}, \texttt{xdvi}, \texttt{dvipdf}, \texttt{dvipdfm}, \texttt{luatex}, \texttt{pdftex}, \texttt{dvipsone}, \texttt{dviwindo}, \texttt{emtex}, \texttt{dviwin}, \texttt{oztex}, \texttt{textures}, \texttt{pctexps}, \texttt{pctexwin}, \texttt{pctexhp}, \texttt{pctex32}, \texttt{trueutex}, \texttt{tcidvi}, \texttt{vtx}, \texttt{zetex}, \texttt{dvisvgm},

- options that determine the target color model\footnote{Section 2.2.3 on page 11 explains how this setting can be overridden at any point in a document.} (\texttt{natural, rgb, cmy, cmyk, hsb, gray, RGB, HTML, HSB, Gray}) or disable colored output (\texttt{monochrome}),

- options that control whether and how certain sets of predefined colors are being loaded: \texttt{dvipsnames}, \texttt{dvipsnames*}, \texttt{svgnames}, \texttt{svgnames*}, \texttt{x11names}, \texttt{x11names*},

- option that determine which other packages are to be loaded or supported: \texttt{table},

- options that influence the behaviour of other commands: \texttt{prologue}, \texttt{kernelfbox}, \texttt{xcdraw}, \texttt{noxcdraw}, \texttt{fixinclusion}, \texttt{showerrors}, \texttt{hideerrors},

- obsolete options: \texttt{pctex32}, \texttt{trueutex}, \texttt{tcidvi}, \texttt{vtx}, \texttt{zetex}, \texttt{dvisvgm}.

All available package options (except driver selection and obsolete options) are listed in table 1 on page 9. In order to facilitate the co-operation with the \texttt{hyperref} package, there is a command \texttt{\GetGinDriver} that grabs the driver actually used and puts it into the command \texttt{\GinDriver}. The latter can then be used within \texttt{hyperref} (or other packages), see the code example on page 11. If there is no corresponding \texttt{hyperref} option, \texttt{hypertex} will be taken as default.

Warning: there is a substantial difference between \texttt{xcolor} and \texttt{color} regarding how the \texttt{dvips} option is being handled. The \texttt{color} package implicitly invokes \texttt{dvips} with \texttt{dvipsone} driver.

\footnote{This command is executed automatically if the package option \texttt{hyperref} is used.}
the `dvipsnames` option, whenever one of the `dvips`, `oztex`, `xdvi` drivers is selected. This makes documents less portable, since whenever one of these colors is used without explicit `dvipsnames` option, other drivers like `pdftex` will issue error messages because of unknown colors. Therefore, `xcolor` always requires an explicit `dvipsnames` option to use these names — which then works for all drivers.

2.1.3 Executing additional initialisation commands

\texttt{xcolorcmd} Here is a simple interface to pass commands that should be executed at the end of the `xcolor` package (immediately before the initialising `\color{black}` is executed). Just say `\def\xcolorcmd{⟨commands⟩}` at some point before `xcolor` is loaded.

Example: assuming that `a.tex` is a complete \LaTeX{} document, a command like `
\texttt{latex \def\xcolorcmd{⟨colorlet{black}{red}}\input{a}}`

at the console generates a file `a.dvi` with all occurrences of `black` being replaced by `red`, without the necessity to change the source file itself. (The exact spelling of the console command might vary across operating systems and TeX distributions.)

2.2 Color models

2.2.1 Supported color models

The list of supported color models and their parameter ranges is given in table 3 on page 10. We emphasize that this color support is independent of the chosen driver.

‘Color model support’ also means that it is possible to specify colors directly with their parameters, e.g., by saying `\texttt{\textcolor[rgb]{0.7,0.5,0.3}{foo}}` (foo) or `\textcolor[HTML]{AFFE90}{foo}` (foo).

\textbf{rgb, cmyk, hsb, gray} These are the models supported by PostScript directly. We therefore refer to [1] for a description of their properties and relations. There is a special command to fine-tune the mechanisms of undercolor-removal and black-generation during conversion to the `cmyk` model, see section 6.3.2 on page 50 for details.

\textbf{cmy} This is mainly a model for intermediary calculation steps. With respect to that, it is a simple complement of `rgb`. As far as visualisation is concerned, `cmy` is treated as `cmyk` with `k = 0`.

\textbf{HTML} This is a model derived from `rgb` in order to enable input of color parameters from web pages or CSS files. Therefore, it is not really a color model of its own right, but rather a user interface for convenience. It is worth mentioning that `HTML` accepts any combination of the characters 0–9, A–F, a–f, as long as the string has a length of exactly 6 characters. However, outputs of conversions to `HTML` will always consist of numbers and uppercase letters.

\textbf{Hsb, tHsb} Firstly, `Hsb` is a ‘user interface’ model, mapping `hue ∈ [0,1]` onto \texttt{hue° ∈ [0,H]}, where `H` is given by \texttt{\def\rangeHsb{⟨H⟩}}. Thus, if `H = 360`, we can think of a circle or wheel to specify the `hue°` parameter. Secondly, `Hsb` is the basis for `tHsb`, also named \texttt{tuned Hsb}, which enables the user
Table 1: Package options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>natural</td>
<td>(Default.) Keep all colors in their model, except RGB (converted to rgb), HSB (converted to hsb), and Gray (converted to gray).</td>
</tr>
<tr>
<td>rgb</td>
<td>Convert all colors to the rgb model.</td>
</tr>
<tr>
<td>cmy</td>
<td>Convert all colors to the cmy model.</td>
</tr>
<tr>
<td>cmyk</td>
<td>Convert all colors to the cmyk model.</td>
</tr>
<tr>
<td>hsb</td>
<td>Convert all colors to the hsb model.</td>
</tr>
<tr>
<td>gray</td>
<td>Convert all colors to the gray model. Especially useful to simulate how a black &amp; white printer will output the document.</td>
</tr>
<tr>
<td>RGB</td>
<td>Convert all colors to the RGB model (and afterwards to rgb).</td>
</tr>
<tr>
<td>HTML</td>
<td>Convert all colors to the HTML model (and afterwards to rgb).</td>
</tr>
<tr>
<td>HSB</td>
<td>Convert all colors to the HSB model (and afterwards to hsb).</td>
</tr>
<tr>
<td>Gray</td>
<td>Convert all colors to the Gray model (and afterwards to gray).</td>
</tr>
<tr>
<td>dvipsnames, dvipsnames*</td>
<td>Load a set of predefined colors.</td>
</tr>
<tr>
<td>svgnames, svgnames*</td>
<td>Load a set of predefined colors according to SVG 1.1.</td>
</tr>
<tr>
<td>x11names, x11names*</td>
<td>Load a set of predefined colors according to Unix/X11.</td>
</tr>
<tr>
<td>table</td>
<td>Load the colortbl package, in order to use the tools for coloring rows, columns, and cells within tables.</td>
</tr>
<tr>
<td>fixpdftex</td>
<td>This option is obsolete and does nothing at all.</td>
</tr>
<tr>
<td>hyperref</td>
<td>Support the hyperref package in terms of color expressions by defining additional keys (see section 2.10 on page 27).</td>
</tr>
<tr>
<td>prologue</td>
<td>Write prologue information to .xcp file for every color definition (as described in section 2.5.1 on page 18).</td>
</tr>
<tr>
<td>kernelfbox</td>
<td>Use LaTeX kernel method to draw (\text{framebox}) frames.</td>
</tr>
<tr>
<td>xcdraw</td>
<td>Use driver-specific commands to draw frames and color boxes.</td>
</tr>
<tr>
<td>noxcdraw</td>
<td>(Default.) Use generic code to draw frames and color boxes.</td>
</tr>
<tr>
<td>fixinclude</td>
<td>Prevent dvips color reset before .eps file inclusion (see section 2.15.2 on page 30).</td>
</tr>
<tr>
<td>showerrors</td>
<td>(Default.) Display an error message if an undefined color is being used (same behaviour as in the original color package).</td>
</tr>
<tr>
<td>hideerrors</td>
<td>Display only a warning if an undefined color is being used, and replace this color by black.</td>
</tr>
</tbody>
</table>

1 See section 2.4.2 on page 17. 2 See section 2.6.2 on page 22.
Table 2: Package loading order

<table>
<thead>
<tr>
<th>Action/Package</th>
<th>colortbl</th>
<th>hyperref</th>
<th>pstricks</th>
<th>color</th>
<th>pstcol</th>
</tr>
</thead>
<tbody>
<tr>
<td>load before xcolor</td>
<td>allowed</td>
<td>allowed</td>
<td>allowed</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>load with xcolor option</td>
<td>table</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>load after xcolor</td>
<td>allowed</td>
<td>allowed</td>
<td>allowed</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

1 pstricks loads xcolor by default

Table 3: Supported color models

<table>
<thead>
<tr>
<th>Name</th>
<th>Base colors/notions</th>
<th>Parameter range</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>rgb</td>
<td>red, green, blue</td>
<td>$[0, 1]^3$</td>
<td></td>
</tr>
<tr>
<td>cmy</td>
<td>cyan, magenta, yellow</td>
<td>$[0, 1]^3$</td>
<td></td>
</tr>
<tr>
<td>cmyk</td>
<td>cyan, magenta, yellow, black</td>
<td>$[0, 1]^4$</td>
<td></td>
</tr>
<tr>
<td>hsb</td>
<td>hue, saturation, brightness</td>
<td>$[0, 1]^3$</td>
<td></td>
</tr>
<tr>
<td>Hsb</td>
<td>hue, saturation, brightness</td>
<td>$[0, H] \times [0, 1]^2$</td>
<td>$H = 360$</td>
</tr>
<tr>
<td>tHsb</td>
<td>hue, saturation, brightness</td>
<td>$[0, H] \times [0, 1]^2$</td>
<td>$H = 360$</td>
</tr>
<tr>
<td>gray</td>
<td>gray</td>
<td>$[0, 1]$</td>
<td></td>
</tr>
<tr>
<td>RGB</td>
<td>Red, Green, Blue</td>
<td>${0, 1, \ldots, L}^3$</td>
<td>$L = 255$</td>
</tr>
<tr>
<td>HTML</td>
<td>RRGGBB</td>
<td>${000000, \ldots, FFFFF}$</td>
<td></td>
</tr>
<tr>
<td>HSB</td>
<td>Hue, Saturation, Brightness</td>
<td>${0, 1, \ldots, M}^3$</td>
<td>$M = 240$</td>
</tr>
<tr>
<td>Gray</td>
<td>Gray</td>
<td>${0, 1, \ldots, N}$</td>
<td>$N = 15$</td>
</tr>
<tr>
<td>wave</td>
<td>lambda (nm)</td>
<td>$[363, 814]$</td>
<td></td>
</tr>
</tbody>
</table>

$L, M, N$ are positive integers; $H$ is a positive real number.

To apply a piecewise linear transformation on \texttt{hue} by shifting selected \texttt{hue} values forward or backward on the circle. This transformation is defined by

\[
\texttt{rangeHsb} \{x_1, y_1; x_2, y_2; \ldots\}
\]

which specifies that \texttt{hue} = \texttt{x1} in \texttt{tHsb} means \texttt{hue} = \texttt{y1} in \texttt{Hsb}, etc. For example, \texttt{yellow} is at 60° in the \texttt{Hsb} circle (red being at 0°), however, in most color wheels used by artists, \texttt{yellow} is at 120°. Therefore, a ‘120,60‘ entry would make sense if we planned to replicate an artist’s color wheel via \texttt{tHsb}. See section 6.3.6 on page 53 for the exact transformation formula and further restrictions, and section 1.4 on page 5 for color wheels and color harmonies. Figure 11 on page 36 may serve for comparison purposes.

Example: \texttt{`\texttt{def rangeHsb}\{60,30;120,60;180,120;210,180;240,240\}`} is in fact \texttt{xcolor’s} default setting.

\textbf{wave} With this model we try to make wavelengths transformable to standard color models, in order to approximate the visual appearance of light waves. While the visible spectrum covers a range of approximately 400–750 nm, the implementation in \texttt{xcolor} generally allows for all real ‘wavelengths’ $\lambda$ that have an absolute value less than 16383.99998 (the biggest number \texttt{TEX} can represent as a dimension). However, the probability of getting any non-black color outside the range
of $\lambda \in [363, 814]$ is exactly zero. Therefore, figure 1 on page 31 illustrates only the mentioned wavelength interval. Note that it is not possible to convert other models to wave in a sensible way, since the latter covers only a limited range of colors.

**RGB, HSB, Gray** These are derived models, turning the continuous $[0, 1]$ parameter ranges of rgb, hsb, and gray to finite sets of integers; we therefore use the term integer models. The constants $L, M, N$ of table 3 are defined via the commands \def{\rangeRGB}{⟨L⟩}, \def{\rangeHSB}{⟨M⟩}, and \def{\rangeGray}{⟨N⟩}.

Changes of these constants can be done before or after the xcolor package is loaded, e.g.,

\documentclass{article}
...
\def{\rangeRGB}{15}
\usepackage[utf8]{xcolor}
...
\GetGinDriver
\usepackage[utf8]{xcolor}
...
\begin{document}
...
\def{\rangeRGB}{63}
...

### 2.2.2 Substituting individual color models

\substitutecolormodel{⟨source model⟩}{⟨target model-list⟩}

Substitute ⟨source model⟩ by the first actually present model that occurs in ⟨target model-list⟩. Only color models of type ⟨num model⟩ are allowed; all changes are local to the current group, but a prepended \xglobal is obeyed.

Example: assume the actual driver has an incorrect implementation of hsb whereas rgb looks well. Then \substitutecolormodel{hsb}{rgb} could be a good choice, since it converts — from that point onwards — all definitions of hsb colors by xcolor’s algorithms into rgb specifications, without touching other models.

### 2.2.3 Changing the target color model within a document

\selectcolormodel{⟨num model⟩}

Sets the target model to ⟨num model⟩, where the latter is one of the model names allowed as package option (i.e., natural, rgb, cmy, cmyk, hsb, gray, RGB, HTML, HSB, Gray), see figure 4 on page 33 for an example. There are two possible hooks, where the conversion to the target model can take place:

- \ifconvertcolorsD: at color definition time\(^3\) (i.e., within \definecolor and friends); this is controlled by the switch \ifconvertcolorsD;
- \ifconvertcolorsU: at time of color usage (immediately before a color is displayed, therefore covering colors that have been defined in other models or that are being

\(^3\)This means that all newly defined colors will be first converted to the target model, then saved.
specified directly like \texttt{\color[rgb]{.1,.2,.3}}); this is controlled by the switch \texttt{ifconvertcolorsU}.

Both switches are set to ‘true’ by selecting any of the models, except \texttt{natural}, which sets them to ‘false’. This applies for selection via a package option as well as via \texttt{\selectcolormodel}. Why don’t we simply convert all colors at time of usage? If many colors are involved, it can save some processing time when all conversions are already done during color definitions. Best performance can be achieved by \texttt{\usepackage[rgb,...]{xcolor}\convertcolorsUfalse}, which is actually the way how \texttt{xcolor} worked up to version 1.07.

### 2.3 Arguments and terminology

Before we describe \texttt{xcolor}’s color-related commands in detail, we define several elements or identifiers that appear repeatedly within arguments of those commands. A general syntax overview is given in table 4 on the next page.

#### 2.3.1 Additional remarks and restrictions on arguments

- **Basic strings and numbers** These arguments do not need much explanation. However, as far as numerical values are concerned, it is noteworthy that real numbers in (La)TEX are — as long as they are to be used in the context of lengths, dimensions, or skips — are restricted to a maximum absolute value $< 16384$. Certainly, in a chain of numerical calculations, this constraint has also to be obeyed for every single interim result, which usually implies further range restrictions. Since \texttt{xcolor} makes extensive use of \texttt{TEX}’s internal dimension registers for most types of calculations, this should be kept in mind whenever \texttt{\langle ext expr\rangle} expressions are to be used.

- **Color names** A \texttt{\langle name\rangle} denotes the declared name (or the name to be declared) of a \texttt{color} or a \texttt{color series}; it may be declared explicitly by one of the following commands: \texttt{\definecolor}, \texttt{\providecolor}, \texttt{\colorlet}, \texttt{\definecolorset}, \texttt{\providecolorset}, \texttt{\definecolorseries}, \texttt{\definecolors}, \texttt{\providecolors}. On the other hand, the reserved color name ‘.’ is declared implicitly and denotes the current color. Actually, besides letters and digits, certain other characters do also work for \texttt{\langle name\rangle} declarations, but the given restriction avoids misunderstandings and ensures compatibility with future extensions of \texttt{xcolor}.


- **Color models** The differentiation between \texttt{core models} (rgb, cmy, cmyk, hsb, gray), \texttt{integer models} (RGB, HTML, HSB, Gray), \texttt{decimal models} (Hsb, tHsb, wave) and \texttt{pseudo models} (currently ‘named’, ‘ps’) has a simple reason: core models with their parameter ranges based on the unit interval $[0,1]$ are best suited for all kinds of calculations, whereas the purpose of the integer models is mainly to facilitate the input of parameters, followed by some transformation into one of the core models. Finally, the decimal models \texttt{Hsb} and \texttt{tHsb} are special-purpose versions of \texttt{hsb}, whereas \texttt{wave} and the pseudo model ‘named’ have a special status, since they are ‘calculation-averse’; it is usually only possible to convert such a color into one of the other models, but not the other way round. Even worse for the pseudo model ‘ps’: since such colors contain PostScript code, they are absolutely intransparent for \texttt{TEX}.
### Table 4: Arguments and terminology

<table>
<thead>
<tr>
<th>Element</th>
<th>Replacement string</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨empty⟩</td>
<td>→ empty string ’ ’</td>
</tr>
<tr>
<td>⟨minus⟩</td>
<td>→ non-empty string consisting of one or more minus signs ‘-’</td>
</tr>
<tr>
<td>⟨plus⟩</td>
<td>→ non-empty string consisting of one or more plus signs ‘+’</td>
</tr>
<tr>
<td>⟨int⟩</td>
<td>→ integer number</td>
</tr>
<tr>
<td>⟨num⟩</td>
<td>→ non-negative integer number</td>
</tr>
<tr>
<td>⟨dec⟩</td>
<td>→ real number</td>
</tr>
<tr>
<td>⟨div⟩</td>
<td>→ non-zero real number</td>
</tr>
<tr>
<td>⟨pt⟩</td>
<td>→ real number from the interval [0, 100]</td>
</tr>
<tr>
<td>⟨id⟩</td>
<td>→ non-empty string consisting of letters and digits (identifier)</td>
</tr>
<tr>
<td>⟨ext id⟩</td>
<td>→ ⟨id⟩₁,…,⟨id⟩₂</td>
</tr>
<tr>
<td>⟨id-list⟩</td>
<td>→ ⟨(ext id)₁,…,(ext id)₂,…⟩</td>
</tr>
<tr>
<td>⟨name⟩</td>
<td>→ ⟨id⟩₁</td>
</tr>
<tr>
<td>⟨model⟩</td>
<td>→ ⟨(num model)⟩</td>
</tr>
<tr>
<td>⟨model-list⟩</td>
<td>→ ⟨(model)₁,…,(model)₂,…⟩</td>
</tr>
<tr>
<td>⟨spec⟩</td>
<td>→ comma-separated list of numerical values (explicit specification)</td>
</tr>
<tr>
<td>⟨spec-list⟩</td>
<td>→ ⟨(spec)₁,…,(spec)₂,…⟩</td>
</tr>
<tr>
<td>⟨type⟩</td>
<td>→ ⟨(empty)⟩</td>
</tr>
<tr>
<td>⟨expr⟩</td>
<td>→ ⟨prefix⟩⟨name⟩⟨mix expr⟩⟨postfix⟩ ⟨standard color expression⟩</td>
</tr>
<tr>
<td>⟨prefix⟩</td>
<td>→ ⟨empty⟩</td>
</tr>
<tr>
<td>⟨mix expr⟩</td>
<td>→ !⟨(pt)₁⟩₁!⟨(name)₁⟩₁!⟨(pt)₂⟩₂!⟨(name)₂⟩₂!⋯!⟨(pt)ₙ⟩ₙ!⟨(name)ₙ⟩ₙ (complete mix expr.)</td>
</tr>
<tr>
<td>⟨postfix⟩</td>
<td>→ !!(⟨num⟩)</td>
</tr>
<tr>
<td>⟨ext expr⟩</td>
<td>→ ⟨core model⟩!⟨(div):⟨(expr)₁⟩₁!⟨(dec):⟨(expr)₂⟩₂!⋯!⟨(expr)ₙ⟩ₙ!⟨(dec):⟨(expr)ₙ⟩ₙ⟩</td>
</tr>
<tr>
<td>⟨func expr⟩</td>
<td>→ !⟨(function):⟨(arg)₁⟩₁,⟨(arg)₂⟩₂,…,⟨(arg)ₙ⟩ₙ⟩ (color function expression)</td>
</tr>
<tr>
<td>⟨function⟩</td>
<td>→ ‘wheel’, ‘twheel’ (color functions)</td>
</tr>
<tr>
<td>⟨color⟩</td>
<td>→ ⟨color expr⟩!⟨(func expr)₁⟩₁!⟨(func expr)₂⟩₂!⋯!⟨(func expr)ₙ⟩ₙ</td>
</tr>
<tr>
<td>⟨color expr⟩</td>
<td>→ ⟨name⟩</td>
</tr>
<tr>
<td></td>
<td>→ ⟨(expr)⟩</td>
</tr>
<tr>
<td></td>
<td>→ ⟨(ext expr)⟩</td>
</tr>
</tbody>
</table>

Remarks: Each → denotes a possible replacement string for the element in the left column; however, further context-dependent restrictions may apply. See main text for details. A string ‘foo’ is always to be understood without the quotes. i, j, k, l, m, n denote non-negative integers, k, l, m, n > 0, m ≤ 8.
The ⟨spec⟩ argument — which specifies the parameters of a color — obviously depends on the underlying color model. We differentiate between explicit and implicit specification, the former referring to numerical parameters as explained in table 3 on page 10, the latter — ideally — referring to driver-provided names.

Examples: ‘.1,.2,.3’, ‘.1 .2 .3’, ‘0.56789’, ‘89ABCD’, ‘ForestGreen’.

Multiple models and specifications These arguments always appear in ⟨model-list⟩ ⟨spec-list⟩ (explicit or implicit) pairs within the following color definition commands: \definecolor, \providecolor, \definecolorset, \providecolorset. First, ⟨model-spec⟩ is being reconciled with the current target model (as set by a package option or the \selectcolormodel command); in case there is no exact match, the first model of the list is chosen. Then, the corresponding color specification will be selected from ⟨spec-list⟩, such that we arrive at a proper ⟨model⟩, ⟨spec⟩ pair. Therefore, in the actual executed color definition there is no ambiguity anymore. The extended form ⟨core model⟩:⟨model⟩₁/⟨model⟩₂/.../⟨model⟩ₘ causes an immediate conversion of the relevant ⟨spec⟩ to ⟨core model⟩; an unknown target model will be silently ignored here.

Examples: ‘rgb/cmyk/named/gray’, ‘0,0,0/0,0,0,1/Black/0’, ‘rgb:cmyk/hsl’.

The type argument This is used only in the context of color defining commands, see the description of \definecolor and friends.

Standard color expressions These expressions serve as a tool to easily specify a certain form of cascaded color mixing which is described in detail in section 2.3.2. The ⟨prefix⟩ argument controls whether the color following thereafter or its complement will be relevant: an odd number of minus signs indicates that the color resulting from the remaining expression has to be converted into its complementary color. An incomplete mix expression is just an abbreviation for a complete mix expression with ⟨name⟩ₙ = white, in order to save some keystrokes in the case of tints. The ⟨postfix⟩ string is usually empty, but it offers some additional functionality in the case of a color series: the non-empty cases require that

- ⟨name⟩ denotes the name of a color series,
- ⟨mix expr⟩ is a complete mix expression.


Extended color expressions These expressions provide another method of color mixing, see section 2.3.3 on page 16 for details. The shorter form

⟨core model⟩:⟨expr⟩₁,⟨dec⟩₁;⟨expr⟩₂,⟨dec⟩₂;...;⟨expr⟩ₖ!⟨dec⟩ₖ

is an abbreviation for the special (and probably most used) case

⟨core model⟩,⟨div⟩:⟨expr⟩₁,⟨dec⟩₁;⟨expr⟩₂,⟨dec⟩₂;...;⟨expr⟩ₖ!⟨dec⟩ₖ
with the following definition (requiring a non-zero sum of all $\langle \text{dec} \rangle_k$ coefficients):

$$\langle \text{div} \rangle := \langle \text{dec} \rangle_1 + \langle \text{dec} \rangle_2 + \cdots + \langle \text{dec} \rangle_k \neq 0.$$  


Function expressions  
These expressions extend the functionality of standard or extended expressions by taking the result of such an expression to perform additional calculations. The number of arguments may vary between different functions, see section 2.3.4 on the following page for details.  
Examples: ‘>wheel,30’, ‘>wheel,30,’ ‘>twheel,1,12’, ‘>twheel,-11,12’.

Colors  
Finally, $\langle \text{color} \rangle$ is the ‘umbrella’ argument, covering the different concepts of specifying colors. This means, whenever there is a $\langle \text{color} \rangle$ argument, the full range of names and expressions, as explained above, may be used.

2.3.2 Meaning of standard color expressions

We explain now how an expression

$$\langle \text{prefix} \rangle\langle \text{name} \rangle!\langle \text{pct} \rangle_1!\langle \text{name} \rangle_1!\langle \text{pct} \rangle_2!\cdots!\langle \text{pct} \rangle_n!\langle \text{name} \rangle_n\langle \text{postfix} \rangle$$

is being interpreted and processed:

1. First of all, the model and color parameters of $\langle \text{name} \rangle$ are extracted to define a temporary color $\langle \text{temp} \rangle$. If $\langle \text{postfix} \rangle$ has the form ‘!![(\text{num})]’, then $\langle \text{temp} \rangle$ will be the corresponding (direct-accessed) color $\langle \text{num} \rangle$ from the series $\langle \text{name} \rangle$.

2. Then a color mix, consisting of $\langle \text{pct} \rangle_1\%$ of color $\langle \text{temp} \rangle$ and $(100 - \langle \text{pct} \rangle_1)\%$ of color $\langle \text{name} \rangle_1$ is computed; this is the new temporary color $\langle \text{temp} \rangle$.

3. The previous step is being repeated for all remaining parameter pairs ($\langle \text{pct} \rangle_2,\langle \text{name} \rangle_2$), $\cdots$, ($\langle \text{pct} \rangle_n,\langle \text{name} \rangle_n$).

4. If $\langle \text{prefix} \rangle$ consists of an odd number of minus signs ‘-’, then $\langle \text{temp} \rangle$ will be changed into its complementary color.

5. If $\langle \text{postfix} \rangle$ has the form ‘!!+’, ‘!!++’, ‘!!+++’, etc., a number of step commands (= number of ‘+’ signs) are performed on the underlying color series $\langle \text{name} \rangle$. This has no consequences for the color $\langle \text{temp} \rangle$.

6. Now the color $\langle \text{temp} \rangle$ is being displayed or serves as an input for other operations, depending on the invoking command.

Note that in a typical step 2 expression $\langle \text{temp} \rangle!\langle \text{pct} \rangle_u!\langle \text{name} \rangle_u$, if $\langle \text{pct} \rangle_u=100$ resp. $\langle \text{pct} \rangle_u=0$, the color $\langle \text{temp} \rangle$ resp. $\langle \text{name} \rangle_u$ is used without further transformations. In the true mix case, $0 < \langle \text{pct} \rangle_u < 100$, the two involved colors may have been defined in different color models, e.g., $\texttt{\definecolor{foo}{rgb}{...}}$ and $\texttt{\definecolor{bar}{cmyk}{...}}$. In general, the second color, $\langle \text{name} \rangle_u$, is transformed into the model of the first color, $\langle \text{temp} \rangle$, then the mix is calculated within
that model. Thus, \((\text{temp})_\nu \cdot \langle \text{name} \rangle_\nu \) and \((\text{name})_\nu \cdot (100 - \text{pct})_\nu \cdot \langle \text{temp} \rangle_\nu\), which should be equivalent theoretically, will not necessarily yield identical visual results.

Figures 5 to 6 on page 33 show some first applications of colors and expressions. More examples are given in figure 3 on page 32. Over and above that, a large set of color examples can be found in [9].

### 2.3.3 Meaning of extended color expressions

An extended color expression

\[
\langle \text{core model} \rangle_\nu, \langle \text{expr} \rangle_1, \langle \text{dec} \rangle_1, \langle \text{expr} \rangle_2, \langle \text{dec} \rangle_2; \ldots; \langle \text{expr} \rangle_k, \langle \text{dec} \rangle_k
\]

mimes color mixing as painters do it: specify a list of colors, each with a \(\langle \text{dec} \rangle\) factor attached to. For such an \(\langle \text{ext expr} \rangle\), each standard color expression \(\langle \text{expr} \rangle_\nu\) will be converted to \(\langle \text{core model} \rangle_\nu\), then the resulting vector is multiplied by \(\langle \text{dec} \rangle_\nu/\langle \text{div} \rangle\), where

\[
\langle \text{div} \rangle := \langle \text{dec} \rangle_1 + \langle \text{dec} \rangle_2 + \cdots + \langle \text{dec} \rangle_k.
\]

Afterwards the sum of all of these vectors is calculated.

Example: mixing 4 parts of red, 2 parts of green, and 1 part of yellow, we get \(\langle \text{core model} \rangle_\nu, \langle \text{expr} \rangle_1, \langle \text{dec} \rangle_1, \langle \text{expr} \rangle_2, \langle \text{dec} \rangle_2, \langle \text{expr} \rangle_3, \langle \text{dec} \rangle_3\) via \(\text{color}\{\text{rgb: red, 4; green, 2; yellow, 1}\}\). Trying the same with \(-1\) parts of yellow instead, we get \(\langle \text{core model} \rangle_\nu, \langle \text{expr} \rangle_1, \langle \text{dec} \rangle_1, \langle \text{expr} \rangle_2, \langle \text{dec} \rangle_2, \langle \text{expr} \rangle_3, \langle \text{dec} \rangle_3\) via \(\text{color}\{\text{rgb: yellow, 1}\}\). Note that this mechanism can also be used to display an individual color (expression) in a certain color model: \(\text{color}\{\text{rgb: yellow, 1}\}\) results in such a conversion. The general form

\[
\langle \text{core model} \rangle_\nu, \langle \text{div} \rangle_\nu, \langle \text{expr} \rangle_1, \langle \text{dec} \rangle_1, \langle \text{expr} \rangle_2, \langle \text{dec} \rangle_2, \ldots, \langle \text{expr} \rangle_k, \langle \text{dec} \rangle_k
\]

does the same operation with the only difference that the divisor \(\langle \text{div} \rangle_\nu\) is being specified instead of calculated. In the above example, we get a shaded version \(\langle \text{core model} \rangle_\nu, \langle \text{div} \rangle_\nu, \langle \text{expr} \rangle_1, \langle \text{dec} \rangle_1, \langle \text{expr} \rangle_2, \langle \text{dec} \rangle_2, \langle \text{expr} \rangle_3, \langle \text{dec} \rangle_3\) via \(\text{color}\{\text{rgb: red, 4; green, 2; yellow, 1}\}\). Note that it is not forbidden to specify a \(\langle \text{div} \rangle\) argument which is smaller than the sum of all \(\langle \text{dec} \rangle_\nu\), such that one or more of the final color specification parameters could be outside the interval \([0, 1]\). However, the mapping of equation (7) takes care of such cases.

### 2.3.4 Color functions

Color functions take a comma-separated list of arguments, and they serve to transform the given color (i.e., the result of all calculations prior to the function call) into a new color.

**wheel**

**Color wheel calculations** Arguments: \((\text{angle})\) or \((\text{angle}), (\text{full circle})\), the former being an abbreviation of \((\text{angle}), \text{\texttt{rangeHsb}}\). These functions allow to calculate related colors by harmonic relations based on color wheels (cf. section 1.4 on page 5). The second argument \((\text{full circle})\) declares how many units a full circle consists of, the first argument states by how many units the given color has to be translated.

---

\(^4\)Exception: in order to avoid strange results, this rule is being reversed if \((\text{temp})\) origins from the \texttt{gray} model; in this case it is converted into the underlying model of \((\text{name})\).
rotated. To this end, the given color is first converted to $\text{Hsb}$ (in case of $\text{wheel}$), yielding $\text{hue}^\circ$, $\text{saturation}$, and $\text{brightness}$, respectively. Then

$$\text{hue}^\circ := \text{hue}^\circ + \frac{(\text{angle})}{(\text{full circle})} \cdot H, \quad \text{hue} := u\left(\frac{\text{hue}^\circ}{H}\right)$$

where $u$ is the range-reduction function of equation (7) and $H = \text{rangeHsb}$. With $\text{saturation}$ and $\text{brightness}$ left untouched, the final model is $\text{hsb}$. The $\text{tuwheel}$ function works similarly, but its arguments refer to $\text{tHsb}$ instead of $\text{Hsb}$. Examples are shown in figure 12 on page 37.

2.4 Predefined colors

2.4.1 Colors that are always available

Within $\text{xcolor.sty}$, the following color names are defined: \text{red}, \text{green}, \text{blue}, \text{cyan}, \text{magenta}, \text{yellow}, \text{black}, \text{gray}, \text{white}, \text{darkgray}, \text{lightgray}, \text{brown}, \text{lime}, \text{olive}, \text{orange}, \text{pink}, \text{purple}, \text{teal}, \text{violet}.

This base set of colors can be used without restrictions in all kinds of color expressions, as explained in section 2.3 on page 12.

2.4.2 Additional sets of colors

There are also sets of color names that may be loaded by $\text{xcolor}$ via package options, available in two variants: a ‘normal’ version (e.g., $\text{dvipsnames}$) and a ‘starred’ version (e.g., $\text{dvipsnames*}$). The first variant simply defines all the colors immediately, whereas the second applies the mechanism of deferred definition. In the latter case, individual color names have to be activated by $\text{\definecolors}$ or $\text{\providecolors}$ commands, as described in section 2.5.4 on page 20, before they can be applied in a document.

- $\text{dvipsnames/dvipsnames*}$ loads a set of 68 $\text{cmyk}$ colors as defined in the $\text{dvips}$ driver. However, these colors may be used in all supported drivers.

- $\text{svgnames/svgnames*}$ loads a set of 151 $\text{rgb}$ color names\(^5\) according to the SVG 1.1 specification \[17\]\(^6\), enhanced by 4 names taken from the file $\text{rgb.txt}$ which is part of Unix/X11 distributions. Note that HTML4 accepts a subset of 16 color keywords (using identical specifications), see \[16\] and section 4 on page 38.

- $\text{x11names/x11names*}$ loads a set of 317 $\text{rgb}$ color names\(^7\) that are basically variations of a subset of the SVG set mentioned before, according to the file $\text{rgb.txt}$ which is part of Unix/X11 distributions\(^8\). We describe now how to access all 752 color names of $\text{rgb.txt}$ without much effort:

  - Load $\text{x11names}$ as well as $\text{svgnames}$.

\(^5\)In fact, these names represent 141 different colors.

\(^6\)Actually, the cited specification lists only lowercase names, and the original definitions are given in $\text{RGB}$ parameters, converted to $\text{rgb}$ by the author.

\(^7\)These names represent 315 different colors.

\(^8\)Again, the original definitions are given in $\text{RGB}$ parameters, converted to $\text{rgb}$ by the author.
– Capitalise initials and skip blanks: DarkSlateGray instead of dark slate gray.

– X11 names without numbers are identical to the corresponding SVG colors, except in 5 cases: use Gray0, Grey0, Green0, Maroon0, Purple0 instead of Gray, Grey, Green, Maroon, Purple to obtain the original X11 colors.

– For \( N = 0, 1, \ldots, 100 \) use ‘[gray]{N/100}’ or ‘black{100 − N}’ instead of gray\( N \) or grey\( N \).

The color names and corresponding displays are listed in section 4 on page 38. Section 2.15.1 on page 30 describes how to deal with name clashes while using both svgnames and dvipsnames in the same document. See also [9] for a systematic set of color and mix examples.

2.5 Color definition

2.5.1 Ordinary and named colors

In the color package there is a distinction between ‘colors’ (defined by the command \texttt{\definecolor}) and ‘named colors’ (defined by \texttt{\DefineNamedColor}, which is allowed only in the preamble). Whenever an ordinary color is being used in a document, it will be translated into a \texttt{\special} command that contains a — driver-specific — numerical description of the color which is written to the .dvi file. On the other hand, named colors offer the opportunity to store numerical values at a central place whereas during usage, colors may be identified by their names, thus enabling post-processing if required by the output device.

All drivers delivered with the standard graphics package support the formalism of defining and invoking ‘named colors’. However, real support for the concept behind that, i.e. employing names instead of parameters, ranges from ‘none’ to ‘complete’. We demonstrate the current situation for three different drivers:

• \texttt{dvips} has very good support for the ‘named’ concept; the PostScript equivalents to the color names defined by dvipsnames are being loaded — unless switched off — by dvips automatically. However, additional names have to be made known to the PostScript interpreter by some kind of header file. Since version 2.01, xcolor offers an integrated solution for this task: by invoking the package option prologue, a PostScript header file xcolor.pro is loaded by dvips. Additionally, under this option every color definition command\footnote{This is not only true for the document preamble, but for the document body as well.} (\texttt{\definecolor}, \texttt{\colorlet}, etc.) will generate some PostScript code that is written to an auxiliary file with the extension .xcp (shortcut for xcolor prologue). This file is as well loaded by dvips as a prologue, thus making all color names available to the PostScript interpreter. Of course, the .xcp file may be edited before dvips is applied, making it easy to change device-specific color parameters at a central place. Note that the PostScript code is designed similar to color.pro: only \texttt{new} names are defined. This allows to preload other prologue files with color definitions that are not being destroyed by xcolor. On the other hand, it requires the user to take care about redefining color names.

Example: \texttt{\colorlet{foo}{red}\colorlet{foo}{blue}\color{foo}} will
switch to blue in the usual xcolor logic, however the .ps file would display red (unless foo had been defined differently before).

It should be stressed that this mechanism is only employed by the prologue option. Without that, the predefined ‘named’ colors activated by the dvipsnames option (without employing any tints, shades, color expressions, etc.) may be used in this way, all other ‘named’ colors are unknown to PostScript.

• dvipdfm supports only the standard dvipsnames colors since these are hard-coded in the dvipdfm program itself; there seems to be no way to load any user-defined prologue files.

• pdftex does not offer conceptual support, all ‘named’ colors are converted immediately to their numerical representation. It therefore allows unrestricted definition and usage of named colors (although offering no added value through this).

Typically, a .dvi viewer will have difficulties to display user-defined ‘named’ colors. For example, MiKTeX’s viewer Yap currently displays only ‘named’ colors from the dvipsnames set. Thus, whenever the prologue option is invoked together with dvips, all other colors will appear black. However, after employing dvips, a PostScript viewer should display the correct colors.

2.5.2 Color definition in xcolor

\definecolor \{(type)\} {\{(name)\} \{(model-list)\} \{(spec-list)\}} ¹⁰

This is one of the commands that may be used to assign a (name) to a specific color. Afterwards, this color is known to the system (in the current group) and may be used in color expressions, as explained in section 2.3 on page 12. It replaces both color’s \DefineNamedColor and \definecolor. Note that an already existing color’s \DefineNamedColor and \definecolor. Note that an already existing color (name) will be overwritten. The variable \tracingcolors controls whether such an overwriting will be logged or not (see section 2.13 on page 29 for details). The arguments are described in section 2.3 on page 12. Hence, valid expressions for color definitions are

• \definecolor{red}{rgb}{1,0,0},
• \definecolor{red}{rgb/cmyk}{1,0,0/0,1,1,0},
• \definecolor{red}{hsb/rgb/cmyk}{1,0,0/0,1,1,0},
• \definecolor[named]{Black}{cmyk}{0,0,0,1},
• \definecolor{myblack}{named}{Black},

where the last command is equivalent to \colorlet{myblack}{Black} (see below); the second command defines red in the rgb or cmyk model, depending on the current setting of the target model, whereas the third will additionally transform the color to hsb prior to saving. Note that there is a special \pstricks-related version as described in section 2.11 on page 28.

\providecolor \{(type)\} {\{(name)\} \{(model-list)\} \{(spec-list)\}}

¹⁰Prior to version 2.00, this command was called xdefinecolor, the latter name still being available for compatibility reasons.
Similar to \texttt{\definecolor}, but the color \texttt{(name)} is only defined if it does not exist already.

\texttt{\colorlet} \[\langle type \rangle\{\langle name \rangle\}{\langle num model \rangle}\{\langle color \rangle}\]

Copies the actual color which results from \texttt{(color)} to \texttt{(name)}. If \texttt{(num model)} is non-empty, \texttt{(color)} is first transformed to the specified model, before \texttt{(name)} is being defined. The pseudo model ‘named’ is not allowed here, it may, however, be specified in the \texttt{(type)} argument. Note that an already existing color \texttt{(name)} will be overwritten.

Example: we said \texttt{\colorlet{tableheadcolor}{gray!25}} in the preamble of this document. In most of the tables we then formatted the first row by using the command \texttt{\rowcolor{tableheadcolor}}.

\subsection*{2.5.3 Defining sets of colors}

\texttt{\definecolorset} \[\langle type \rangle\{\langle model-list \rangle\}{\langle head \rangle}\{\langle tail \rangle\}{\langle set spec \rangle}\]

This command facilitates the construction of \textit{color sets}, i.e. (possibly large) sets of individual colors with common underlying \texttt{(model-list)} and \texttt{(type)}. Here, \texttt{(set spec)} = \texttt{(name)}\,\texttt{(spec-list)}\,\ldots\,\texttt{(name)}\,\texttt{(spec-list)}\,\ldots \ (l \geq 1 \text{name/specification-list pairs}). Individual colors are being constructed by single\texttt{\definecolor} commands, \(\lambda = 1, \ldots, l\). For example,

- \texttt{\definecolorset{rgb}{}{}{red,1,0,0;green,0,1,0;blue,0,0,1}} could be used to define the basic colors \texttt{red}, \texttt{green}, and \texttt{blue};

- \texttt{\definecolorset{rgb}{x}\{10\}{red,1,0,0;green,0,1,0;blue,0,0,1}} would define the colors \texttt{xred10}, \texttt{xgreen10}, and \texttt{xblue10}.

\texttt{\providecolorset} \[\langle type \rangle\{\langle model-list \rangle\}{\langle head \rangle}\{\langle tail \rangle\}{\langle set spec \rangle}\]

Similar to \texttt{\definecolorset}, but based on \texttt{\providecolor}, thus the individual colors are defined only if they do not exist already.

\subsection*{2.5.4 Immediate and deferred definitions}

Traditionally, the definition of a color as described above leads to the immediate construction of a command that holds at least the information needed by the driver to display the desired color. Thus, defining 300 colors, e.g., by loading a huge set of predefined colors, will result in 300 new commands, although most of them — except for the purpose of displaying lists of colors — will hardly ever be used within a document. Along the development of computer memory — increasing in size, decreasing in price — recent \TeX implementations have increased their provisions for internal memory stacks that are available for strings, control sequences, etc. However, as memory continues to be finite, it may still be useful (or occasionally necessary) to have a method at hand that allows to reduce memory requirements a bit. This is the point where \textit{deferred color definition} comes into play. Its principle is simple: for every definition of this type (e.g., via \texttt{\preparecolor}), all necessary

\[\begin{align*}
\texttt{\definecolorset{rgb/hsb/cmyk/gray}\{\{red,1,0,0/0,1,1/0,1,1,0/0.3;green,\ldots\}.}
\end{align*}\]
information is saved on a specific global definition stack, where it can be taken from later (e.g., via \definecolors) in order to construct the actual color command. Note that the following commands are only to be used in the document preamble, since the definition stack of colors for deferred definitions is deleted at the begin of the document body — in order to save memory.

\preparecolor \{(type)\}\{(name)\}\{(model-list)\}\{spec-list\}

Similar to \definecolor, but the color \(name\) is not yet being defined: the arguments \(model-list\) and \(spec-list\) are evaluated immediately, then all necessary parameters (i.e. \(type\), \(name\), \(model\), \(spec\)) are put onto the definition stack for later usage.

\preparecolorset \{(type)\}\{(model-list)\}\{head\}\{tail\}\{set spec\}

\ifdefcolors Similar to \definecolorset, but depending on the \ifdefcolors switch: if set to ‘true’, to each element of the set the command \definecolor (i.e. immediate definition) is applied; if set to ‘false’, \preparecolor (i.e. deferred definition) is applied. For example, the package option svgnames performs something like \definecolorstrue\preparecolorset, whereas svgnames* acts like \definecolorsfalsestrue\preparecolorset. Both options set \definecolorstrue at the end, in order to have a proper starting point for other color sets.

\DefineNamedColor \{(type)\}\{(name)\}\{(model-list)\}\{spec-list\}

is provided mainly for compatibility reasons, especially to support the predefined colors in dvipsnam.def. It is the same as \{cmd\}\{(type)\}\{(name)\}\{(model)\}\{(spec)\}, where \(cmd\) is either \definecolor or \preparecolor, depending on the state of \ifdefcolors. Note that color's restriction to allow \DefineNamedColor only in the document preamble has been abolished in xcolor.

\definecolors \{(id-list)\}

Recall that \(id-list\) has the form \(id_1, \ldots, id_n\) where each \(id_i\) is either an identifier \(id\) or an assignment \(id = \lambda \), describes the general case: the definition stack is searched for the name \(id\) and its corresponding color parameters; if there is no match, nothing happens; if the name \(id\) is on the stack and its color parameters are \(type\), \model\, and \spec\, then the command \definecolor\{(type)\}\{(model)\}\{(spec)\} is executed. Thus, the user may control by which names the prepared colors are to be used in the document. Note that the entry \(id\) is not removed from the stack, such that it can be used several times (even within the same \definecolors command).

\providecolors \{(id-list)\}

Similar to \definecolors, but based on \providecolor, thus the individual colors are defined only if they do not exist already.

2.5.5 Global color definitions

\ifdefglobalcolors By default, definitions via \definecolor, \providecolor, ... are available only within the current group. By setting \globalcolorstrue, all such definitions are being made globally available — until the current group ends.\textsuperscript{12} Another method to specify that an individual color definition is to be made global is to prefix it by \xglobal, e.g., \xglobal\definecolor{foo}....

\textsuperscript{12}The switch may also be set in the preamble in order to control the whole document.
2.6 Color application

2.6.1 Standard color commands

Here is the list of user-level color commands, as known from the color package, but with an extended syntax for the colors, allowing for expressions etc.:

\color {⟨color⟩}
\{⟨model-list⟩\}{⟨spec-list⟩}

Switches to the color given either by name/expression or by model/specification. This color will stay in effect until the end of the current \TeX group.

\textcolor {⟨model-list⟩}{⟨spec-list⟩}{⟨text⟩}
\textcolor {⟨color⟩}{⟨text⟩}

are just alternative syntax for \color, in which the groups are added implicitly. Thus ⟨text⟩ appears in the specified color, but then the color reverts to its previous value. Additionally, it calls \leavevmode to ensure the start of horizontal mode.

\pagecolor {⟨model-list⟩}{⟨spec-list⟩}
\pagecolor {⟨color⟩}
\nopagecolor

Specifies the background color for the current, and all following, pages. It is a global declaration which does not respect \TeX groups. Remark: all of these commands except \color require that the ⟨color⟩ resp. ⟨spec⟩ arguments are put into curly braces \{\}, even if they are buried in macros. For example, after \def\foo{red}, one may say \color\foo, but one should always write \textcolor{\foo}{bar} instead of \textcolor{\foo}{bar} in order to avoid strange results.

Note that color-specific commands from other packages may give unexpected results if directly confronted with color expressions (e.g., soul’s \sethlcolor and friends). However, one can turn the expression into a name via \colorlet and try to use that name instead.

\nopagecolor

Removes any currently specified page color returning to the default transparent background. It is not yet supported by all driver options and so generates a warning if there is no definition in the driver file.

2.6.2 Colored boxes

\colorbox {⟨color⟩}{⟨text⟩}
\colorbox {⟨model-list⟩}{⟨spec-list⟩}{⟨text⟩}

Takes the same argument forms as \textcolor, but the color specifies the background color of the box.

\fcolorbox {⟨frame color⟩}{⟨background color⟩}{⟨text⟩}
\fcolorbox {⟨frame model-list⟩}{⟨frame spec-list⟩}{⟨background model-list⟩}{⟨background spec-list⟩}{⟨text⟩}

Puts a frame of the first color around a box with a background specified by the second color. If only the first optional argument is given, it specifies the color model for both colors. Besides the possibility to specify color expressions as arguments, \fcolorbox now offers more flexibility for its arguments than the color version:

• \texttest \fcolorbox{gray}{yellow}{test},

• \texttest \fcolorbox{cmyk}{0,0,0,0.5}{0,0,1,0}{test},
Additionally, \fcolorbox uses a new approach to frame drawing, which is an extension of Donald Arseneau’s suggestion in bug report latex/3655 [2]. The main difference to \LaTeX’s implementation is that box construction and frame drawing are split into separate operations, such that the frame is drawn after the box contents has been constructed. This ensures that the frame is always on top of the box. Donald Arseneau improved speed as well as memory requirements of this approach. Furthermore, a new macro is introduced:
\begin{verbatim}
\boxframe \{⟨width⟩\} \{⟨height⟩\} \{⟨depth⟩\}
\end{verbatim}

\boxframe \{⟨width⟩\} \{⟨height⟩\} \{⟨depth⟩\}
Draws a frame with a linewidth of \fboxrule. Returns a \hbox with outer dimensions \{⟨width⟩\}, \{⟨height⟩\}, \{⟨depth⟩\}. By this approach, a frame-primitive may also be provided by a driver file, in order to exploit driver-specific drawing facilities (see below). Again, this macro was optimised by Donald Arseneau.

The new frame approach is used for \fcolorbox as well as \LaTeX’s \fbox and \framebox commands, unless the \kernelfbox option is specified, which returns to \LaTeX’s original definitions of \fbox. Option \xcdraw uses PostScript commands to draw frames and color boxes in case of the dvips driver and PDF code to draw frames in case of the pdftex and dvipdfm drivers. This is still experimental code that may confuse .dvi viewers. The opposite option \noxcdraw forces usage of the generic (driver-independent) code.

### 2.6.3 Colors in math

While it is possible to color parts of a formula using \color commands the approach is fairly cumbersome. Therefore the command \mathcolor has been added to the color and xcolor package, which has the same syntax as \textcolor. The full documentation and examples of its use is in mathcolor.pdf in the \LaTeX documentation.

### 2.6.4 Using the current color

Within a color expression, ‘.’ serves as a placeholder for the current color. See figure 7 on page 33 for an example.

It is also possible to save the current color for later use, e.g., via the command \colorlet{foo}{.}.

Note that in some cases the current color is of rather limited use, e.g., the construction of an \fcolorbox implies that at the time when the \{⟨background color⟩\} is evaluated, the current color equals the \{⟨frame color⟩\}; in this case ‘.’ does not refer to the current color outside the box.

### 2.6.5 Color testing

\textcolor{env}{\{⟨num models⟩\}}

This is a simple tabular environment in order to test (display) colors in different models, showing both the visual result and the model-specific parameters. The optional \{⟨num models⟩\} argument is a comma-separated list of numerical color
models (as usual without spaces) which form the table columns; the default list is rgb, cmyk, hsb, HTML.

\testcolor \{\{color\}\} \{\{model-list\}\} \{\{spec-list\}\}

Each \testcolor command generates a table row, containing a display sample plus the respective parameters for each of the models. If the column-model matches the model of the color in question, its parameters are underlined. Note that this command is only available within the testcolors environment. For applications see figure 2 on page 31 and figures 11, 12.

2.7 Color blending

The purpose of color blending is to add some mixing color (expression) to all subsequent explicit color commands. Thus, it is possible to perform such a mix (or blend) operation for many colors without touching the individual commands.

\blendcolors \{\{mix expr\}\}

\blendcolors* \{\{mix expr\}\}

Initialises all necessary parameters for color blending. The actual (completed) color blend expression is stored in \colorblend. In the starred version, the argument will be appended to a previously defined blend expression. An empty \(\text{mix expr}\) argument will switch blending off.

Example: after \blendcolors{!50!yellow}, the colors \(\textcolor{red}{\text{red}}, \text{green}, \text{blue}\) are transformed into \(\textcolor{red}{\text{red}}, \text{green}, \text{blue}\), an additional \blendcolors*{!50} yields \(\textcolor{red}{\text{red}}, \text{green}, \text{blue}\). In order to achieve global scope, \blendcolors may be prefixed by \xglobal.

Remark: color blending is applied only to explicit color commands, i.e. \color, \fcolorbox and the like. In the previous example the frames are not being blended because their color is set by an driver-internal command (switching back to the ‘current color’). Thus, to influence these implicit colors as well, we have to set the current color after the blending: \blendcolors{!50!yellow}\color{black} results in \(\textcolor{red}{\text{red}}, \text{green}, \text{blue}\), an additional \blendcolors*{!50}\color{black} yields \(\textcolor{red}{\text{red}}, \text{green}, \text{blue}\).

2.8 Color masks and separation

The purpose of color separation is to represent all colors that appear in the document as a combination of a finite subset of base colors and their tints. Most prominent is cmyk separation, where the base colors are cyan, magenta, yellow, and black, as required by the printers. This can be done by choosing the package option cmyk, such that all colors will be converted in this model, and post-processing the output file. We describe now another — and more general — solution: color masking. How does it work? Color masking is based on a specified color model (\textit{m-model}) and a parameter vector (\textit{m-spec}). Whenever a color is to be displayed in the document, it will first be converted to (\textit{m-model}), afterwards each component of the resulting color vector will be multiplied by the corresponding component of (\textit{m-spec}). For example, let’s assume that (\textit{m-model}) equals cmyk, and (\textit{m-spec}) equals \((\mu_c, \mu_m, \mu_y, \mu_k)\). Then an arbitrary color \textit{foo} will be transformed according to

\[
\textit{foo} \mapsto (c, m, y, k) \mapsto (\mu_c \cdot c, \mu_m \cdot m, \mu_y \cdot y, \mu_k \cdot k)
\]
Obviously, color separation is a special case of masking by the vectors $(1, 0, 0, 0)$, $(0, 1, 0, 0)$, etc. An interesting application is to shade or tint all colors by masking them with $(x, x, x)$ in the rgb or cmy model, see the last two rows in figure 9 on page 35.

\begin{verbatim}
\makecolors{(num model)}{(color)}
\end{verbatim}

Initialises all necessary parameters for color masking: if \texttt{(num model)} is not specified (or empty), \texttt{(m-model)} will be set to the natural model of \texttt{(color)}, otherwise to \texttt{(num model)}; the color specification of \texttt{(color)} is extracted to define \texttt{(m-spec)}. Additionally, \texttt{\maskcolorstrue} is performed. Color masking can be switched off temporarily by \texttt{\maskcolorsttrue}, or — in a more radical way — by \texttt{\maskcolors{}}, which in addition clears the initialisation parameters. In general, the scope of \texttt{\maskcolors} is the current group (unless it is prefixed by the \texttt{\xglobal} command), but it may be used in the document preamble as well. The final remark of the color blending section applies here similarly.

Now it is easy to separate a complete document without touching the source code: \texttt{latex \def\xcolorcmd{\maskcolors[cmyk]{cyan}}\input{a}} will do the cyan part of the job for \texttt{a.tex}.

\textbf{Caution:} \texttt{xcolor} has no idea about colors in files that are included via the command \texttt{\includegraphics}, e.g., images of type .eps, .pdf, .jpg, or .png. Such files have to be separated separately. Nevertheless, \texttt{xcolor} offers some basic support by storing the mask color in \texttt{\colormask}, which can be used to decide which file is to be included:

\begin{verbatim}
\def\temp{cyan}\ifx\colormask\temp \includegraphics{foo_c}\else \def\temp{magenta}\ifx\colormask\temp \includegraphics{foo_m}\else ...
\fi\fi
\end{verbatim}

\section{Color series}

Automatic coloring may be useful in graphics or chart applications, where a — potentially large and unspecified — number of colors are needed, and the user does not want or is not able to specify each individual color. Therefore, we introduce the term \textit{color series}, which consists of a base color and a scheme, how the next color is being constructed from the current color.

The practical application consists of three parts: definition of a color series (usually once in the document), initialisation of the series (potentially several times), and application — with or without stepping — of the current color of the series (potentially many times).

\subsection{Definition of a color series}

\begin{verbatim}
\deffcolorseries\texttt{\{\{name\}\}\{\{core model\}\}\{\{method\}\}\{\{b-model\}\}\{\{b-spec\}\}\{\{s-model\}\}\{\{s-spec\}\}}
\end{verbatim}

Defines a color series called \texttt{(name)}, whose calculations are performed within the color model \texttt{(core model)}, where \texttt{(method)} selects the algorithm (one of \texttt{step}, \texttt{grad}, \texttt{last}, see below). The method details are determined by the remaining arguments:

- \texttt{\{\{b-model\}\}\{\{b-spec\}\}} specifies the base (= first) color in the algorithm, either directly, e.g., \texttt{\{rgb\}{1,0.5,0.5}}, or as a \texttt{(color)}, e.g., \texttt{\{-yellow!50\}}, if the optional argument is missing.
• \{(s-model)\}\{(s-spec)\} specifies how the step vector is calculated in the algorithm, according to the chosen \{(method)\}:

- \textbf{step, grad}: the optional argument is meaningless, and \{(s-spec)\} is a parameter vector whose dimension is determined by \{(core model)\}, e.g., \{0.1,-0.2,0.3\} in case of \texttt{rgb}, \texttt{cmy}, or \texttt{hsh}.

- \textbf{last}: the last color is specified either directly, e.g., \texttt{[rgb]{1,0.5,0.5}}, or as a \{(color)\}, e.g., \texttt{{-yellow!50}}, if the optional argument is missing.

This is the general scheme:

\[
\text{color}_1 := \text{base}, \quad \text{color}_{n+1} := U(\text{color}_n + \text{step})
\]  

(6)

for \(n = 1, 2, \ldots\), where \(U\) maps arbitrary real \(m\)-vectors into the unit \(m\)-cube:

\[
U(x_1, \ldots, x_m) = (u(x_1), \ldots, u(x_m)), \quad u(x) = \begin{cases} 
1 & \text{if } x = 1 \\
[x] & \text{if } x \neq 1 
\end{cases}
\]  

(7)

Thus, every step of the algorithm yields a valid color with parameters from the interval \([0, 1]\).

Now, the different methods use different schemes to calculate the step vector:

• \textbf{step, grad}: the last argument, \{(s-spec)\}, defines the directional vector \texttt{grad}.

• \textbf{last}: \{(s-spec)\} resp. \{(s-model)\}\{(s-spec)\} defines the color parameter vector \texttt{last}.

Then, during \\texttt{\resetcolorseries}, the actual step vector is calculated:

\[
\text{step} := \begin{cases} 
\text{grad} & \text{if } (\text{method}) = \text{step} \\
\frac{1}{(\text{div})} \cdot \text{grad} & \text{if } (\text{method}) = \text{grad} \\
\frac{1}{(\text{div})} \cdot (\text{last} - \text{base}) & \text{if } (\text{method}) = \text{last}
\end{cases}
\]  

(8)

Please note that it is also possible to use the current color placeholder \texttt{.} within the definition of color series. Thus, \texttt{\definecolorseries{foo}{rgb}{last}{.}{-.}} will set up a series that starts with the current color and ends with its complement. Of course, similar to TeX’s \texttt{\let} primitive, the current definition of the current color at the time of execution is used, there is no relation to current colors in any later stage of the document.

### 2.9.2 Initialisation of a color series

\texttt{\resetcolorseries\{(div)\}\{(name)\}}

This command has to be applied at least once, in order to make use of the color series \{(name)\}. It resets the current color of the series to the base color and calculates the actual step vector according to the chosen \{(div)\}, a non-zero real number, for the methods \texttt{grad} and \texttt{last}, see equation (8). If the optional argument is empty, the value stored in the macro \texttt{\colorseriescycle} is applied. Its default value is 16, which can be changed by \texttt{\def\colorseriescycle{(div)}}, applied before the \texttt{xcolor} package is loaded (similar to \texttt{\rangeRGB} and friends). The optional argument is ignored in case of the \texttt{step} method.
2.9.3 Application of a color series

There are two ways to display the current color of a color series: any of the color expressions in section 2.3 on page 12 used within a \color, \textcolor, \... command will display this color according to the usual syntax of such expressions. However, in the cases when \postfix equals ‘!!+’, \color{\langle name\rangle!!+} etc., will not only display the color, but it will also perform a step operation. Thus, the current color of the series will be changed in that case. An expression \color{\langle name\rangle!![\langle num\rangle]} enables direct access to an element of a series, where \langle num\rangle = 0, 1, 2, ..., starting with 0 for the base color. See figure 8 on page 34 for a demonstration of different methods.

2.9.4 Differences between colors and color series

Although they behave similar if applied within color expressions, the objects defined by \definecolor and \definecolorseries are fundamentally different with respect to their scope/availability: like color’s original \definecolor command, \definecolor generates local colors, whereas \definecolorseries generates global objects (otherwise it would not be possible to use the stepping mechanism within tables or graphics conveniently). E.g., if we assume that bar is an undefined color, then after saying

```latex
\begingroup
\definecolorseries{foo}{rgb}{last}{red}{blue}
\resetcolorseries[10]{foo}
\definecolor{bar}{rgb}{.6,.5,.4}
\endgroup
```

commands like \color{foo} or \color{foo!!+} may be used without restrictions, whereas \color{bar} will give an error message. However, it is possible to say \colorlet{bar}{foo} or \colorlet{bar}{foo!!+} in order to save the current color of a series locally — with or without stepping.

2.10 Border colors for hyperlinks

Change 2022: The options described in this subsection have been removed from the code as no longer relevant. hyperref can handle color names in the border color keys.

This affects the package options hyperref, and the Keys \citetbordercolor, \filebordercolor, \linkbordercolor, \menubordercolor, \pagebordercolor, \runcolor, \urlbordercolor and \xpdfborder. The hyperref package offers all kinds of support for hyperlinks, pdfmarks etc. There are two standard ways to make hyperlinks visible (see the package documentation[14] for additional information on how to set up these features):

- print hyperlinks in a different color than normal text, using the keys citecolor, filecolor, linkcolor, menucolor, pagecolor, runcolor, uricolor with color expressions, e.g., \hypersetup{uricolor=green!50};
- display a colored border around hyperlinks, using the keys citebordercolor, filebordercolor, linkbordercolor, menubordercolor, pagebordercolor, runcolor, urilbordercolor with explicit numerical rgb parameter specification, e.g., \hypersetup{urlbordercolor={1 0.5 0.25}}.
Obviously, the second method is somewhat inconvenient since it does not allow for color names or even color expressions. Therefore, \texttt{xcolor} provides — via the package option \texttt{hyperref} — a set of extended keys \texttt{xcitebordercolor}, \texttt{xfilebordercolor}, \texttt{xlinkbordercolor}, \texttt{xmenubordercolor}, \texttt{xpagebordercolor}, \texttt{xrunbordercolor}, \texttt{xurlbordercolor} which are being used in conjunction with color expressions, e.g., \texttt{\hypersetup{xurlbordercolor=-green!50}}.

Another new key, \texttt{xpdfborder}, provides a way to deal with a \texttt{dvips}-related problem: for most of the drivers, a setting like \texttt{pdfborder={0 0 1}} will determine the width of the border that is drawn around hyperlinks in points. However, in the \texttt{dvips} case, the numerical parameters are interpreted in relation to the chosen output resolution for processing the .dvi file into a .ps file. Unfortunately, at the time when the .dvi is constructed, nobody knows if and at which resolution a transformation into .ps will take place afterwards. Consequently, any default value for \texttt{pdfborder} may be useful or not. Within \texttt{hyperref}, the default for \texttt{dvips} is \texttt{pdfborder={0 0 12}}, which works fine for a resolution of 600 or 1200 dpi, but which produces an invisible border for a resolution of 8000 dpi, as determined by the command-line switch \texttt{-Ppdf}. On the other hand, setting \texttt{pdfborder={0 0 80}} works fine for \texttt{dvips} at 8000 dpi, but makes a document unportable, since other drivers (or even \texttt{dvips} in a low resolution) will draw very thick boxes in that case. This is were the \texttt{xpdfborder} key comes in handy: it rescales its arguments for the \texttt{dvips} case by a factor 80 (ready for 8000 dpi) and leaves everything unchanged for other drivers. Thus one can say \texttt{xpdfborder={0 0 1}} in a driver-independent way.

### 2.11 Additional color specification in the \texttt{pstricks} world

For \texttt{pstricks} users, there are different ways of invoking colors within command option keys:

- \texttt{\psset{linecolor=green!50}}
- \texttt{\psset{linecolor=[rgb]{0.5,1,0.5}}}
- \texttt{\psframebox[linecolor={[rgb]{0.5,1,0.5}}]{foo}}

Note the additional curly braces in the last case; without them, the optional argument of \texttt{\psframebox} would be terminated too early.

\texttt{\definecolor[ps]{(name)}{(core model-list)}{(code)}}

Stores PostScript \texttt{(code)} — that should not contain slash \texttt{/} characters — within a color. Example: after \texttt{\definecolor[ps]{foo}{rgb}{bar}}, the \texttt{pstricks} command \texttt{\psline[linecolor=foo]}... inserts \texttt{`bar setrgbcolor'} where the linecolor information is required — at least in case of the \texttt{dvips} driver. See also \texttt{xcolor2.tex} for an illustrative application.

### 2.12 Color in tables

The commands in this subsection were in previous versions provided by the \texttt{xcolor} package as an extension of the \texttt{colortbl} commands. They have now been merged into the \texttt{colortbl} package properly. The package can be loaded either with the \texttt{table} option or independently with \texttt{\usepackage}. The description of the commands is kept only for historical reasons. If in doubt consult the documentation of \texttt{colortbl}.
\rowcolors \{(commands)\}\{(row)\}\{(odd-row color)\}\{(even-row color)\}
\rowcolors* \{(commands)\}\{(row)\}\{(odd-row color)\}\{(even-row color)\}

One of these commands has to be executed before a table starts. \{(row)\} tells the number of the first row which should be colored according to the \{(odd-row color)\} and \{(even-row color)\} scheme. Each of the color arguments may also be left empty (= no color). In the starred version, \{(commands)\} are ignored in rows with inactive \rowcolors status (see below), whereas in the non-starred version, \{(commands)\} are applied to every row of the table. Such optional commands may be \hline or \noalign{\langle stuff\rangle}.

\rownum
The \rowcolors status is activated (i.e., use coloring scheme) by default and/or \showrowcolors; it is inactivated (i.e., ignore coloring scheme) by the command \hiderowcolors. The counter \rownum may be used within such a table to access the current row number. An example is given in figure 10 on page 35.

Note that table coloring may be combined with color series. This method was used to construct the examples in figure 8 on page 34.

### 2.13 Color information

\extractcolors \{(color)\}\{(cmd)\}
Extracts the color specification of \{(color)\} and puts it into \{(cmd)\}; equivalent to \def\cmd{\{(model)\}\{(spec)\}}.

\extractcolors* \{(color)\}\{(model-cmd)\}\{(color-cmd)\}
Extracts the color specification of \{(color)\} and puts it into \{(model-cmd)\} and \{(color-cmd)\}, respectively.

\tracingcolors =\{int\}
Controls the amount of information that is written into the log file:

- \{int\} ≤ 0: no specific color logging.
- \{int\} ≥ 1: ignored color definitions due to \providecolor are logged.
- \{int\} ≥ 2: multiple (i.e. overwritten) color definitions are logged.
- \{int\} ≥ 3: every command that defines a color will be logged.
- \{int\} ≥ 4: every command that sets a color will be logged.

Like \TeX’s \tracing... commands, this command may be used globally (in the document preamble) or locally/block-wise. The package sets \tracingcolors=0 as default. Remark: since registers are limited and valuable, no counter is wasted for this issue.

Note that whenever a color is used that has been defined via \{color\}’s \definecolor command rather than \xcolor’s new \definecolor and friends, a warning message ‘Incompatible color definition’ will be issued.¹³

### 2.14 Color conversion

\convertcolors \{(model)\}\{(spec)\}\{(target model)\}\{(cmd)\}
Converts a color, given by the \{(spec)\} in model \{(model)\}, into \{(target model)\} and stores the new color specification in \{(cmd)\}. \{(target model)\} must be of type \{num

¹³This should not happen since usually there is no reason to load \{color\} in parallel to \xcolor.
model), whereas \langle model \rangle may also be ‘named’, in which case \langle spec \rangle is simply the name of the color.
Example: \texttt{\convertcolorspec{cmyk}\{0.81,1,0,0.07\}{HTML}\{html\}} acts like \texttt{\def \tmp{1F00ED}}.

2.15 Problems and solutions

2.15.1 Name clashs between dvipsnames and svgnames

Due to the fixed option processing order (which does not depend on the order how the options were specified in the \texttt{\usepackage} command), the svgnames colors will always overrule dvipsnames colors with identical names. This can lead to undesired results if both options are used together. For instance, Fuchsia yields \texttt{\definecolor{Fuchsia}} under the regime of dvipsnames and \texttt{\definecolor{Fuchsia}} with respect to svgnames. However, there is a simple trick — based on deferred color definition — that allows us to use colors from both sets in the desired way:

\texttt{\usepackage[dvipsnames*,svgnames]{xcolor}}
\texttt{\definecolor{Fuchsia}}

Now all colors from the SVG set are available (except Fuchsia) plus Fuchsia from the other set.

2.15.2 Change color of included .eps file

In general, xcolor cannot change colors of an image that is being included via the \texttt{\includegraphics} command from the graphics or graphicx package. There is, however, a limited opportunity to influence the current color of included PostScript files. Consider the following file foo.eps which draws a framed gray box:

\texttt{%!PS-Adobe-3.0 EPSF-3.0}
\texttt{%%BoundingBox: 0 0 60 12}
\texttt{0 0 60 12 rectfill}
\texttt{0.75 setgray}
\texttt{2 2 56 8 rectfill}

Now run the following code through \LaTeX{} and dvips:

\texttt{\documentclass{minimal}}
\texttt{\usepackage[fixinclude]{xcolor}}
\texttt{\usepackage{graphics}}
\texttt{\begin{document}}
\texttt{\includegraphics{foo}} \texttt{\textcolor{red}{\includegraphics{foo}}}
\texttt{\end{document}}

The resulting .ps file will display two gray boxes: the first with a black frame, the second with a red frame. If we had omitted the fixinclude option, the second box would also display a black frame. This is because dvips usually resets the current color to black immediately before including an .eps file.
3 Examples

Figure 1: Color spectrum

\newcount\WL \unitlength.75pt
\begin{picture}(460,60)(355,-10)
  \sffamily \tiny \linethickness{1.25\unitlength} \WL=360
  \multiput(360,0)(1,0){456}{{\color[wave]{\the\WL}\line(0,1){50}}\global\advance\WL1}
  \linethickness{0.25\unitlength}\WL=360
  \multiput(360,0)(20,0){23}{{\picture(0,0)
    \line(0,-1){5} \multiput(5,0)(5,0){3}{\line(0,-1){2.5}}
    \put(0,-10){\makebox(0,0){\the\WL}}\global\advance\WL20
    \endpicture}
  \end{picture}

Figure 2: Color testing

\begin{testcolors}[rgb,cmyk,hsb,HTML,gray]
\testcolor{olive}
\testcolor{red!50!green}
\testcolor{-cyan!50!magenta}
\testcolor[cmyk]{0,0,1,0.5}
\testcolor[cmyk]{0,0,.5,.5}
\testcolor[rgb:cmyk]{0,0,.5,.5}
\end{testcolors}
Figure 3: Progressing from one to another color

<table>
<thead>
<tr>
<th>Color</th>
<th>Definition/representation (pdftex driver)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyGreen</td>
<td>{0.92 0.87 0.87 k 0.92 0 0.87 0.09 K}{cmyk}{0.92,0.87,0.09}</td>
</tr>
<tr>
<td>MyGreen-rgb</td>
<td>{0 0.91 0.04001 rg 0 0.91 0.04001 RG}{rgb}{0,0.91,0.04001}</td>
</tr>
<tr>
<td>MyGreen-cmy</td>
<td>{1 0.09 0.95999 0 k 1 0.09 0.95999 0 K}{cmy}{1,0.09,0.95999}</td>
</tr>
<tr>
<td>MyGreen-hsb</td>
<td>{0 0.91 0.03995 rg 0 0.91 0.03995 RG}{hsb}{0.34065,1,0.91}</td>
</tr>
<tr>
<td>MyGreen-gray</td>
<td>{0.5383 g 0.5383 Q}{gray}{0.5383}</td>
</tr>
</tbody>
</table>
Figure 4: Target color model

\selectcolormodel ...{natural} ...
...{rgb} ...
...{cmy} ...
...{cmyk} ...
...{hsb} ...
...{gray}

Figure 5: Standard color expressions

\fboxrule6pt
\fcolorbox {red!70!green}% outer frame
{yellow!30!blue}% outer background
{\fcolorbox{-yellow!30!blue}% inner frame
{-red!70!green}% inner background
{Test\textcolor{red!72.75}{Test}\color{-green}Test}

Figure 6: Standard color expressions

\fboxrule6pt
\fcolorbox
{red!70!green}% outer frame
{yellow!30!blue}% outer background
{\fcolorbox{-yellow!30!blue}% inner frame
{-red!70!green}% inner background
{Test\textcolor{red!72.75}{Test}\{Test}\color{-green}Test}}

Figure 7: Current color

\def\test{current, \textcolor{!50}{50\%}, \textcolor{-.}{complement}, \textcolor{yellow!50!}{mix}}
\def\Test{Test\Test\Test\Test\Test}
\textcolor{blue}{\{Test\textcolor{red}{\Test}\textcolor{red}{\Test}\textcolor{red}{\Test}\textcolor{red}{\Test}\textcolor{red}{\Test}}

current, 50%, complement, mix
and current, 50%, complement, mix
TestTestTestTestTest
and current, 50%, complement, mix
TestTestTestTestTest
TestTestTestTestTest
Figure 8: Color series

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>$S_2$</td>
<td>$G_1$</td>
<td>$G_2$</td>
<td>$L_1$</td>
<td>$L_2$</td>
<td>$L_3$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

**Individual definitions**

$S_1$ \definecolorseries{test}{rgb}{step}{rgb}[.95,.85,.55]{.17,.47,.37}

$S_2$ \definecolorseries{test}{hsb}{step}{hsb}[.575,1,1]{.11,-.05,0}

$G_1$ \definecolorseries{test}{rgb}{grad}{rgb}[.95,.85,.55]{3,11,17}

$G_2$ \definecolorseries{test}{hsb}{grad}{hsb}[.575,1,1]{.987,-.234,0}

$L_1$ \definecolorseries{test}{rgb}{last}{rgb}[.95,.85,.55]{.95,.85,.55}

$L_2$ \definecolorseries{test}{hsb}{last}{hsb}[.575,1,1]{-.425,.15,1}

$L_3$ \definecolorseries{test}{cmy}{last}{yellow!50}{blue}

$L_4$ \definecolorseries{test}{hsb}{last}{yellow!50}{blue}

$L_5$ \definecolorseries{test}{cmy}{last}{yellow!50}{blue}

**Common definitions**

\resetcolorseries[12]{test}

\rowcolors[\hline]{1}{test!!+}{test!!+}

\begin{tabular}{c}
\number\rownum\ | \number\rownum\ | \number\rownum\ | \number\rownum\ | \number\rownum\ | \number\rownum\ | \number\rownum\ | \number\rownum\ | \number\rownum\ | \number\rownum\ | \number\rownum\ | \number\rownum\ | \number\rownum\ | \number\rownum\ | \number\rownum\
\end{tabular}
Figure 9: Color masking

\begin{figure}
\centering
\begin{tabular}{|c|}
\hline
\end{tabular}
\end{figure}

Figure 10: Alternating row colors in tables: \rowcolors vs. \rowcolors*

\begin{verbatim}
\rowcolors[\hline]{3}{green!25}{yellow!50} \arrayrulecolor{red!75!gray}
\begin{tabular}{ll}
test & row \number\rownum\ \\
test & row \number\rownum\ \\
test & row \number\rownum\ \\
\arrayrulecolor{black}
test & row \number\rownum\ \\
test & row \number\rownum\ \\
\rowcolor{blue!25}
test & row \number\rownum\ \\
\hiderowcolors
test & row \number\rownum\ \\
\showrowcolors
test & row \number\rownum\ \\
\multicolumn{1}{>{\columncolor{red!12}}l}{test} & row \number\rownum\ \\
\end{tabular}
\end{verbatim}
<table>
<thead>
<tr>
<th>color</th>
<th>rgb</th>
<th>cmyk</th>
<th>hsb</th>
<th>Hsb</th>
<th>tHsb</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Hsb]0,1,1</td>
<td>1 0 0</td>
<td>0 1 1 0</td>
<td>0 1 1</td>
<td>0 1 1</td>
<td>0 1 1</td>
</tr>
<tr>
<td>[Hsb]15,1,1</td>
<td>1 0.25000 0</td>
<td>0 0.74998 1 0</td>
<td>0.04167 1 1</td>
<td>15.00128 1 1</td>
<td>30.00256 1 1</td>
</tr>
<tr>
<td>[Hsb]30,1,1</td>
<td>1 0.49998 0</td>
<td>0 0.50002 1 0</td>
<td>0.08333 1 1</td>
<td>29.99872 1 1</td>
<td>59.99744 1 1</td>
</tr>
<tr>
<td>[Hsb]45,1,1</td>
<td>1 0.75 0</td>
<td>0 0.25 1 0</td>
<td>0.125 1 1</td>
<td>45 1 1</td>
<td>90 1 1</td>
</tr>
<tr>
<td>[Hsb]60,1,1</td>
<td>1 0.99998 1 0</td>
<td>0 0.00020 1 0</td>
<td>0.16667 1 1</td>
<td>60.00128 1 1</td>
<td>120.00128 1 1</td>
</tr>
<tr>
<td>[Hsb]75,1,1</td>
<td>0.75002 1 0</td>
<td>0.24998 0 1 0</td>
<td>0.20833 1 1</td>
<td>74.99872 1 1</td>
<td>134.99872 1 1</td>
</tr>
<tr>
<td>[Hsb]90,1,1</td>
<td>0.5 1 0</td>
<td>0.5 0 1 0</td>
<td>0.25 1 1</td>
<td>90 1 1</td>
<td>150 1 1</td>
</tr>
<tr>
<td>[Hsb]105,1,1</td>
<td>0.24998 1 0</td>
<td>0.75002 0 1 0</td>
<td>0.29167 1 1</td>
<td>105.00128 1 1</td>
<td>165.00128 1 1</td>
</tr>
<tr>
<td>[Hsb]120,1,1</td>
<td>0.00002 0 1 0</td>
<td>0.99998 1 0 0</td>
<td>0.33333 1 1</td>
<td>119.99872 1 1</td>
<td>179.99872 1 1</td>
</tr>
<tr>
<td>[Hsb]135,1,1</td>
<td>0 1 0.25 0 1</td>
<td>1 0 0.75 0 0</td>
<td>0.375 1 1</td>
<td>135 1 1</td>
<td>187.5 1 1</td>
</tr>
<tr>
<td>[Hsb]150,1,1</td>
<td>0 1 0.50002 1 0</td>
<td>1 0 0.49998 0 0</td>
<td>0.41667 1 1</td>
<td>150.00128 1 1</td>
<td>195.00064 1 1</td>
</tr>
<tr>
<td>[Hsb]165,1,1</td>
<td>0 1 0.75 0 1 0</td>
<td>1 0 0.25002 0 0</td>
<td>0.45833 1 1</td>
<td>164.99872 1 1</td>
<td>202.99936 1 1</td>
</tr>
<tr>
<td>[Hsb]180,1,1</td>
<td>0 1 1 1 0 0</td>
<td>1 0 0 0</td>
<td>0.5 1 1</td>
<td>180 1 1</td>
<td>210 1 1</td>
</tr>
<tr>
<td>[Hsb]195,1,1</td>
<td>0 0.74998 1 0</td>
<td>1 0.25002 0 0</td>
<td>0.54167 1 1</td>
<td>195.00128 1 1</td>
<td>217.50064 1 1</td>
</tr>
<tr>
<td>[Hsb]210,1,1</td>
<td>0 0.50002 1 0</td>
<td>1 0.49998 0 0</td>
<td>0.58333 1 1</td>
<td>209.99872 1 1</td>
<td>224.99936 1 1</td>
</tr>
<tr>
<td>[Hsb]225,1,1</td>
<td>0 0.25 1 0</td>
<td>1 0.75 0 0</td>
<td>0.625 1 1</td>
<td>225 1 1</td>
<td>232.5 1 1</td>
</tr>
<tr>
<td>[Hsb]240,1,1</td>
<td>0 0.00002 0 1 0</td>
<td>0.99998 1 0 0</td>
<td>0.66667 1 1</td>
<td>240.00128 1 1</td>
<td>240.00128 1 1</td>
</tr>
<tr>
<td>[Hsb]255,1,1</td>
<td>0 0.24996 0 1 0</td>
<td>0.75002 1 0 0</td>
<td>0.70833 1 1</td>
<td>254.99872 1 1</td>
<td>254.99872 1 1</td>
</tr>
<tr>
<td>[Hsb]270,1,1</td>
<td>0 0.5 1 0</td>
<td>0.5 1 0 0</td>
<td>0.75 1 1</td>
<td>270 1 1</td>
<td>270 1 1</td>
</tr>
<tr>
<td>[Hsb]285,1,1</td>
<td>0 1 0 0</td>
<td>0 1 0.25 0 0</td>
<td>0.79167 1 1</td>
<td>285.00128 1 1</td>
<td>285.00128 1 1</td>
</tr>
<tr>
<td>[Hsb]300,1,1</td>
<td>0.99998 0 1 0</td>
<td>0.00002 1 0 0</td>
<td>0.83333 1 1</td>
<td>299.99872 1 1</td>
<td>299.99872 1 1</td>
</tr>
<tr>
<td>[Hsb]315,1,1</td>
<td>0 1 0.75 0</td>
<td>0 1 0.25 0 0</td>
<td>0.875 1 1</td>
<td>315 1 1</td>
<td>315 1 1</td>
</tr>
<tr>
<td>[Hsb]330,1,1</td>
<td>1 0 0.49998 0 1 0</td>
<td>1 0.50002 0 0</td>
<td>0.91667 1 1</td>
<td>330.00128 1 1</td>
<td>330.00128 1 1</td>
</tr>
<tr>
<td>[Hsb]345,1,1</td>
<td>1 0 0 0 0</td>
<td>1 0 0.74998 0 0</td>
<td>0.95833 1 1</td>
<td>344.99872 1 1</td>
<td>344.99872 1 1</td>
</tr>
<tr>
<td>[Hsb]360,1,1</td>
<td>1 0 0</td>
<td>0 1 1 0 0</td>
<td>1 1 1</td>
<td>360 1 1</td>
<td>360 1 1</td>
</tr>
</tbody>
</table>

Figure 11: Hsb and tHsb: hue in 15° steps
Figure 12: Color harmony

<table>
<thead>
<tr>
<th>Color</th>
<th>RGB</th>
<th>CMYK</th>
<th>HSV</th>
<th>tHSV</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Complementary colors</strong> (two-color harmony):</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yellow &gt; wheel,1,2</td>
<td>0.00002 0 1</td>
<td>0.99998 1 0 0</td>
<td>240.00128 1 1</td>
<td>240.00128 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,1,2</td>
<td>1 1 0</td>
<td>0 0 1 0</td>
<td>60.00128 1 1</td>
<td>120.00128 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,1,2</td>
<td>0 1 0.99995</td>
<td>0 1 0.00005 0</td>
<td>300.00256 1 1</td>
<td>300.00256 1 1</td>
</tr>
<tr>
<td><strong>Color triad</strong> (three-color harmony):</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yellow &gt; wheel,1,3</td>
<td>0 0 1 1</td>
<td>1 0 0 0</td>
<td>180 1 1</td>
<td>210 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,2,4</td>
<td>0 0 0 0 0</td>
<td>1 0 0 0 0</td>
<td>300.00256 1 1</td>
<td>300.00256 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,3,4</td>
<td>0 0 1 0 1</td>
<td>0 0 1 0 1</td>
<td>60.00128 1 1</td>
<td>120.00128 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,2,3</td>
<td>1 0.99995 0</td>
<td>0 1 0 0 0</td>
<td>240.00128 1 1</td>
<td>240.00128 1 1</td>
</tr>
<tr>
<td><strong>Color tetrad</strong> (four-color harmony):</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yellow &gt; wheel,3,4</td>
<td>0 0 1 0 1</td>
<td>0 0 1 0 1</td>
<td>60.00128 1 1</td>
<td>120.00128 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,2,3</td>
<td>0 1 0.99995</td>
<td>0 1 0.00005 0</td>
<td>300.00256 1 1</td>
<td>300.00256 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,3,4</td>
<td>0.25002 0 1</td>
<td>0 0 0.74998 1 0</td>
<td>15.00128 1 1</td>
<td>30.00256 1 1</td>
</tr>
<tr>
<td><strong>Split complementary colors</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yellow &gt; wheel,7,12</td>
<td>0.5 0 1</td>
<td>0.5 0 1</td>
<td>270 1 1</td>
<td>270 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,5,12</td>
<td>0 0.99995 1</td>
<td>0.5 0.50005 0 0</td>
<td>210.00256 1 1</td>
<td>225.00128 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,2,3</td>
<td>0 1 0</td>
<td>1 0 0 0 0</td>
<td>60.00128 1 1</td>
<td>120.00128 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,7,12</td>
<td>1 0 0.49998</td>
<td>0 1 0.50002 0 0</td>
<td>330.00128 1 1</td>
<td>330.00256 1 1</td>
</tr>
<tr>
<td><strong>Analogous (adjacent) colors</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yellow &gt; wheel,11,12</td>
<td>1 0.50005 0</td>
<td>0 0.49995 1 0</td>
<td>30.00256 1 1</td>
<td>60.00513 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,10,12</td>
<td>1 0 0</td>
<td>0 1 0 0</td>
<td>36 1 1</td>
<td>36 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,2,12</td>
<td>0 1 0.00005</td>
<td>1 0 0.99995 0</td>
<td>120.00256 1 1</td>
<td>180.00128 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,1,12</td>
<td>0.5 1 0</td>
<td>0.5 0 1 0</td>
<td>90 1 1</td>
<td>150 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,7,12</td>
<td>0 0 0.49998</td>
<td>0 1 0.50002 0 0</td>
<td>330.00128 1 1</td>
<td>330.00256 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,2,12</td>
<td>0 0.99995 0</td>
<td>1 0 0.99995 0</td>
<td>120.00256 1 1</td>
<td>180.00128 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,1,12</td>
<td>0.5 1 0</td>
<td>0.5 0 1 0</td>
<td>90 1 1</td>
<td>150 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,7,12</td>
<td>0 0 0.49998</td>
<td>0 1 0.50002 0 0</td>
<td>330.00128 1 1</td>
<td>330.00256 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,2,12</td>
<td>0 1 0.00001</td>
<td>1 0 0.99999 0</td>
<td>120.013 1 1</td>
<td>180.0065 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,10,12</td>
<td>1 0.50005 0</td>
<td>0 0.49995 1 0</td>
<td>30.00256 1 1</td>
<td>60.00513 1 1</td>
</tr>
<tr>
<td>yellow &gt; wheel,11,12</td>
<td>0 0.75012 0</td>
<td>0 0.24998 1 0</td>
<td>45.00714 1 1</td>
<td>90.01428 1 1</td>
</tr>
</tbody>
</table>
4 Colors by Name

4.1 Base colors (always available)

- black
- blue
- brown
- cyan
- darkgray
- gray
- green
- lightgray
- lime
- magenta
- olive
- orange
- pink
- purple
- red
- teal
- violet
- white
- yellow

4.2 Colors via dvipsnames option

- Apricot
- Aquamarine
- Bittersweet
- Black
- Blue
- BlueGreen
- BlueViolet
- BrickRed
- Brown
- BurntOrange
- CadetBlue
- CarnationPink
- Cerulean
- CornflowerBlue
- Cyan
- Dandelion
- DarkOrchid
- Emerald
- ForestGreen
- Fuchsia
- Goldenrod
- Gray
- Green
- GreenYellow
- JungleGreen
- Lavender
- LimeGreen
- Magenta
- Mahogany
- Maroon
- Melon
- MidnightBlue
- Mulberry
- NavyBlue
- OliveGreen
- Orange
- OrangeRed
- Orchid
- Peach
- Periwinkle
- Plum
- ProcessBlue
- Purple
- RawSienna
- Red
- RedOrange
- RedViolet
- Rhodamine
- RoyalBlue
- RoyalPurple
- RubinRed
- Salmon
- SeaGreen
- Sepia
- SkyBlue
- SpringGreen
- Tan
- TealBlue
- Thistle
- Turquoise
- Violet
- VioletRed
- White
- WildStrawberry
- Yellow
- YellowGreen
- YellowOrange

4.3 Colors via svgnames option

- AliceBlue
- AntiqueWhite
- Aqua
- Azure
- Beige
- Bisque
- Black
- BlanchedAlmond
- Blue
- BlueViolet
- Brown
- BurlyWood
- CadetBlue
- Chartreuse
- Chocolate
- Coral
- CornflowerBlue
- Cornsilk
- Crimson
- Cyan
- DarkBlue
- DarkCyan
- DarkGoldenrod
- DarkGray
- DarkGreen
- DarkGrey
- DarkKhaki
- DarkMagenta
- DarkOliveGreen
- DarkOrange
- DarkOrchid
- DarkRed
- DarkSalmon
- DarkSeaGreen
- DarkSlateBlue
- DarkSlateGray
- DarkTurquoise
- DarkViolet
- DeepPink
- DeepSkyBlue
- DimGray
- DimGrey
- DodgerBlue
- FireBrick
- FloralWhite
- ForestGreen
- Fuchsia
- Gainsboro
- GhostWhite
- Gold
- Goldenrod
- Gray
- Green
- GreenYellow
- Grey
- Honeydew
- HotPink
- IndianRed
- Indigo
- Ivory
- Khaki
- Lavender
- LavenderBlush
- LawnGreen
- LemonChiffon
- LightBlue
- LightCoral
- LightCyan
- LightGoldenrod
- LightGoldenrodYellow
- LightGray
- LightGreen
- LightGrey
- LightPink
- LightSalmon
- LightSeaGreen
- LightSkyBlue
- LightSlateBlue
- LightSlateGray
- LightSteelBlue
- LightYellow
- Lime
- LimeGreen
- Linen
- Magenta
- Maroon
- MediumAquamarine
- MediumBlue
- MediumOrchid
- MediumPurple
- MediumSeaGreen
- MediumSlateBlue
- MediumSpringGreen
- MediumTurquoise
- MediumVioletRed
- MidnightBlue
- MintCream
- MistyRose
- Moccasin
- NavajoWhite
- Navy
- NavyBlue
- OldLace
- Olive
- OliveDrab

HTML4 color keyword subset: Aqua, Black, Blue, Fuchsia, Gray, Green, Lime, Maroon, Navy, Olive, Purple, Red, Silver, Teal, White, Yellow.

Colors taken from Unix/X11: LightGoldenrod, LightSlateBlue, NavyBlue, VioletRed.

### 4.4 Colors via x11names option
Duplicate colors: Gray0 = Grey0, Green0 = Green1.
5 Technical Supplement

5.1 Color models supported by drivers

Since some of the drivers only pretend to support the hsb model, we included some code to bypass this behaviour. The models actually added by xcolor are shown in the log file. Table 5 lists mainly the drivers that are part of current MiKTeX distributions and their color model support. Probably, other distributions behave similarly.

<table>
<thead>
<tr>
<th>Driver</th>
<th>Version</th>
<th>rgb</th>
<th>cmy</th>
<th>cmyk</th>
<th>hsb</th>
<th>gray</th>
<th>RGB</th>
<th>HTML</th>
<th>HSB</th>
<th>Gray</th>
</tr>
</thead>
<tbody>
<tr>
<td>dvipdf</td>
<td>2015/12/30 v3.0k</td>
<td>d</td>
<td>n</td>
<td>d</td>
<td>d</td>
<td>i</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>dvips</td>
<td>2015/12/30 v3.0k</td>
<td>d</td>
<td>n</td>
<td>d</td>
<td>d</td>
<td>i</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>dvipsone</td>
<td>2015/12/30 v3.0k</td>
<td>d</td>
<td>n</td>
<td>d</td>
<td>d</td>
<td>i</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>pctex32</td>
<td>2015/12/30 v3.0k</td>
<td>d</td>
<td>n</td>
<td>d</td>
<td>d</td>
<td>i</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>pctexps</td>
<td>2015/12/30 v3.0k</td>
<td>d</td>
<td>n</td>
<td>d</td>
<td>d</td>
<td>i</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>pdftex</td>
<td>2011/05/27 v0.06d</td>
<td>d</td>
<td>n</td>
<td>d</td>
<td>n</td>
<td>i</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>luatex</td>
<td>2016/01/23 v0.01b</td>
<td>d</td>
<td>n</td>
<td>d</td>
<td>n</td>
<td>i</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>dvipdfm</td>
<td>1999/9/6 vx.x</td>
<td>d</td>
<td>n</td>
<td>d</td>
<td>n</td>
<td>i</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>dvipdfmx</td>
<td>2016/04/06 v4.08</td>
<td>d</td>
<td>n</td>
<td>d</td>
<td>?</td>
<td>d</td>
<td>i</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>textures</td>
<td>1997/5/28 v0.3</td>
<td>d</td>
<td>n</td>
<td>d</td>
<td>?</td>
<td>i</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>vtex</td>
<td>1999/01/14 v6.3</td>
<td>d</td>
<td>n</td>
<td>d</td>
<td>n</td>
<td>i</td>
<td>i</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>xetex</td>
<td>2016/04/06 v4.08</td>
<td>d</td>
<td>n</td>
<td>d</td>
<td>d</td>
<td>i</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>tcidvi</td>
<td>2015/12/30 v3.0k</td>
<td>i</td>
<td>n</td>
<td>n</td>
<td>i</td>
<td>d</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>truetex</td>
<td>2015/12/30 v3.0k</td>
<td>i</td>
<td>n</td>
<td>i</td>
<td>n</td>
<td>i</td>
<td>d</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>dvivin</td>
<td>2015/12/30 v3.0k</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>emtex</td>
<td>2015/12/30 v3.0k</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>pctexhp</td>
<td>2015/12/30 v3.0k</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>pctexwin</td>
<td>2015/12/30 v3.0k</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td></td>
</tr>
</tbody>
</table>

Driver’s color model support: d = direct, i = indirect, n = none

5.2 How xcolor handles driver-specific color models

Although there is a variety of drivers that implement different approaches to color visualisation, they all have some features in common, as defined by the original color package. One of these features is that any color model ‘foo’ requires a \color@foo\{\cmd\}\{\spec\} command in order to translate the ‘foo’-dependent color \spec into some driver-specific code that is stored in \cmd. Therefore, xcolor in general detects driver-support for the ‘foo’ model via the existence of \color@foo.

By this mechanism, xcolor can also change the behaviour of certain models without touching the driver file itself. A good example is the \substitutecolormodel command which is used during the package initialisation process to provide support for models that are not covered by the actual driver (like hsb for pdftex) or that have incorrect implementations (like hsb for dvipdfm).
5.3 Behind the scenes: internal color representation

Every definition of a color in order to access it by its name requires an internal representation of the color, i.e. a macro that contains some bits of information required by the driver to display the color properly.\color\texttt{\definecolor{foo}{...}{...}} generates a command \texttt{\color@foo}\footnote{The double backslash is intentional.} which contains the color definition in a driver-dependent way; therefore it is possible but non-trivial to access the color model and parameters afterwards (see the \texttt{colorinfo} package \cite{12} for a solution).

\color\texttt{\DefineNamedColor{named}{foo}{...}{...}} generates \texttt{\col@foo}\footnote{The single backslash is intentional.} which again contains some driver-dependent information. In this case, an additional \texttt{\color@foo} will only be defined if the package option \texttt{usecolors} is active.

\texttt{xcolor\definecolor{foo}{...}{...}} generates a command \texttt{\color@foo} as well, which combines the features of the former commands and contains both the driver-dependent and driver-independent information, thus making it possible to access the relevant parameters in a standardised way. Although it has now a different syntax, \texttt{\color@foo} expands to the same expression as the original command. On the other hand, \texttt{\col@foo} commands are no longer needed and therefore not generated in the ‘named’ case: \texttt{xcolor} works with a single color data structure (as described).

Table 6 on the following page shows some examples for the two most prominent drivers. See also figure 3 on page 32 which displays the definitions with respect to the driver that was used to process this document.

5.4 A remark on accuracy

Since the macros presented here require some computation, special efforts were made to ensure a maximum of accuracy for conversion and mixing formulas — all within \TeX’s limited numerical capabilities.\footnote{For example, applying the ‘transformation’ \texttt{\dimen0=0.\langle int\rangle pt \the\dimen0} to all 5-digit numbers \texttt{\langle int\rangle} of the range 00000…99999, exactly 34464 of these 100000 numbers don’t survive unchanged. We are not talking about gobbled final zeros here…} We decided to develop and include a small set of commands to improve the quality of division and multiplication results, instead of loading one of the packages that provide multi-digit arithmetic and a lot more, like \texttt{realcalc} or \texttt{fp}. The marginal contribution of the latter packages seems not to justify their usage for our purposes. Thus, we stay within a sort of fixed-point arithmetic framework, providing at most 5 decimal digits via \TeX’s dimension registers.
Table 6: Driver-dependent internal color representation

<table>
<thead>
<tr>
<th>dvips driver</th>
<th>pdftex driver</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\color@Plum=macro:</code></td>
<td><code>\color@Plum=macro:</code></td>
</tr>
<tr>
<td><code>-&gt;rgb .5 0 1.</code></td>
<td><code>-&gt;rgb .5 0 1.</code></td>
</tr>
<tr>
<td><code>{\definecolor{Plum}{rgb}{.5,0,1}}</code></td>
<td><code>{\definecolor{Plum}{rgb}{.5,0,1}}</code></td>
</tr>
<tr>
<td><code>xcolor</code></td>
<td><code>xcolor</code></td>
</tr>
<tr>
<td><code>-&gt;\xcolor@ {}{rgb .5 0 1 rg .5 0 1 RG}.</code></td>
<td><code>-&gt;\xcolor@ {}{rgb .5 0 1 rg .5 0 1 RG}.</code></td>
</tr>
<tr>
<td><code>(with option usenames)</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6 The Formulas

6.1 Color mixing

In general, we use linear interpolation for color mixing:

\[ \text{mix}(C, C', p) = p \cdot C + (1 - p) \cdot C' \]  

(9)

Note that there is a special situation in the \texttt{hsb} case: if \texttt{saturation} = 0 then the color equals a gray color of level \texttt{brightness}, independently of the \texttt{hue} value. Therefore, to achieve smooth transitions of an arbitrary color to a specific gray (like white or black), we actually use the formulas

\[
\begin{align*}
\text{tint}_{\texttt{hsb}}(C, p) &= p \cdot C + (1 - p) \cdot (\text{hue}, 0, 1) \\
\text{shade}_{\texttt{hsb}}(C, p) &= p \cdot C + (1 - p) \cdot (\text{hue}, 0, 0) \\
\text{tone}_{\texttt{hsb}}(C, p) &= p \cdot C + (1 - p) \cdot (\text{hue}, 0, \frac{1}{2})
\end{align*}
\]  

(10) (11) (12)

where \( C = (\text{hue}, \text{saturation}, \text{brightness}) \).

From equation (9) and the way how color expressions are being interpreted, as described in section 2.3 on page 12, it is an easy proof by induction to verify that a color expression

\[ C_0!P_1!C_1!P_2! \ldots !P_n!C_n \]  

(13)

with \( n \in \{0, 1, 2, \ldots \} \), colors \( C_0, C_1, \ldots, C_n \), and percentages \( P_1, \ldots, P_n \in [0, 100] \) will result in a parameter vector

\[
C = \sum_{\nu=0}^{n} \left( \prod_{\mu=\nu+1}^{n} p_\mu \right) (1 - p_\nu) \cdot C_\nu \\
= p_0 \cdot \cdots \cdot p_1 \cdot C_0 \\
+ p_0 \cdot \cdots \cdot p_2 (1 - p_1) \cdot C_1 \\
+ p_0 \cdot \cdots \cdot p_3 (1 - p_2) \cdot C_2 \\
+ \ldots \\
+ p_0 (1 - p_{n-1}) \cdot C_{n-1} \\
+ (1 - p_n) \cdot C_n
\]  

(14)

where \( p_0 := 0 \) and \( p_\nu := P_\nu/100 \) for \( \nu = 1, \ldots, n \). We note also a split formula:

\[
C_0!P_1!C_1!\ldots!P_{n+k}!C_{n+k} = p_{n+k} \cdot \cdots \cdot p_{n+1} \cdot C_0!P_1!C_1!\ldots!P_n!C_n \\
- p_{n+k} \cdot \cdots \cdot p_{n+1} \cdot C_n \\
+ C_n!P_{n+1}!C_{n+1}!\ldots!P_{n+k}!C_{n+k}
\]  

(15)

6.2 Conversion between integer and real models

We fix a positive integer \( n \) and define the sets \( \mathcal{I}_n := \{0, 1, \ldots, n\} \) and \( \mathcal{R} := [0, 1] \).

The complement of \( \nu \in \mathcal{I}_n \) is \( n - \nu \), the complement of \( x \in \mathcal{R} \) is \( 1 - x \).
Table 7: Color constants

<table>
<thead>
<tr>
<th>model/constant</th>
<th>white</th>
<th>black</th>
<th>gray</th>
</tr>
</thead>
<tbody>
<tr>
<td>rgb</td>
<td>(1, 1, 1)</td>
<td>(0, 0, 0)</td>
<td>(1/2, 1/2, 1/2)</td>
</tr>
<tr>
<td>cmy</td>
<td>(0, 0, 0)</td>
<td>(0, 0, 0)</td>
<td>(1/2, 1/2, 1/2)</td>
</tr>
<tr>
<td>cmyk</td>
<td>(0, 0, 0)</td>
<td>(0, 0, 0)</td>
<td>(0, 0, 1)</td>
</tr>
<tr>
<td>hsb</td>
<td>(h, 0, 1)</td>
<td>(h, 0, 0)</td>
<td>(h, 0, 1/2)</td>
</tr>
<tr>
<td>Hsb</td>
<td>(h°, 0, 1)</td>
<td>(h°, 0, 0)</td>
<td>(h°, 0, 1/2)</td>
</tr>
<tr>
<td>tHsb</td>
<td>(h°, 0, 1)</td>
<td>(h°, 0, 0)</td>
<td>(h°, 0, 1/2)</td>
</tr>
<tr>
<td>gray</td>
<td>1</td>
<td>0</td>
<td>1/2</td>
</tr>
<tr>
<td>RGB</td>
<td>(L, L, L)</td>
<td>(0, 0, 0)</td>
<td>([L + 1]/2, [L + 1]/2, [L + 1]/2)</td>
</tr>
<tr>
<td>HTML</td>
<td>FFFFFF</td>
<td>000000</td>
<td>808080</td>
</tr>
<tr>
<td>HSB</td>
<td>(H, 0, M)</td>
<td>(H, 0, 0)</td>
<td>(H, 0, [M + 1]/2)</td>
</tr>
<tr>
<td>Gray</td>
<td>N</td>
<td>0</td>
<td>[N + 1]/2</td>
</tr>
</tbody>
</table>

Table 8: Color conversion pairs

<table>
<thead>
<tr>
<th>from/to</th>
<th>rgb</th>
<th>cmy</th>
<th>cmyk</th>
<th>hsb</th>
<th>Hsb</th>
<th>tHsb</th>
<th>gray</th>
<th>RGB</th>
<th>HTML</th>
<th>HSB</th>
<th>Gray</th>
</tr>
</thead>
<tbody>
<tr>
<td>rgb</td>
<td>id</td>
<td>*</td>
<td>(cmy)</td>
<td>*</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>(hsb)</td>
<td>(gray)</td>
</tr>
<tr>
<td>cmy</td>
<td>*</td>
<td>id</td>
<td>*</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>*</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>(rgb)</td>
</tr>
<tr>
<td>cmyk</td>
<td>(cmy)</td>
<td>*</td>
<td>id</td>
<td>(cmy)</td>
<td>(cmy)</td>
<td>(cmy)</td>
<td>*</td>
<td>(cmy)</td>
<td>(cmy)</td>
<td>(cmy)</td>
<td>(gray)</td>
</tr>
<tr>
<td>hsb</td>
<td>*</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>id</td>
<td>*</td>
<td>(Hsb)</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>*</td>
<td>(rgb)</td>
</tr>
<tr>
<td>Hsb</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>*</td>
<td>id</td>
<td>*</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>(hsb)</td>
</tr>
<tr>
<td>tHsb</td>
<td>(Hsb)</td>
<td>(Hsb)</td>
<td>(Hsb)</td>
<td>(Hsb)</td>
<td>*</td>
<td>id</td>
<td>(Hsb)</td>
<td>(Hsb)</td>
<td>(Hsb)</td>
<td>(Hsb)</td>
<td>(Hsb)</td>
</tr>
<tr>
<td>gray</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>id</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>RGB</td>
<td>*</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>id</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>(rgb)</td>
</tr>
<tr>
<td>HTML</td>
<td>*</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>id</td>
<td>(rgb)</td>
<td>(rgb)</td>
<td>(rgb)</td>
</tr>
<tr>
<td>HSB</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>*</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>id</td>
<td>(hsb)</td>
</tr>
<tr>
<td>Gray</td>
<td>(gray)</td>
<td>(gray)</td>
<td>(gray)</td>
<td>(gray)</td>
<td>(gray)</td>
<td>(gray)</td>
<td>*</td>
<td>(gray)</td>
<td>(gray)</td>
<td>(gray)</td>
<td>(gray)</td>
</tr>
<tr>
<td>wave</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>*</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>(hsb)</td>
<td>(hsb)</td>
</tr>
</tbody>
</table>

id = identity function; * = specific conversion function;
(model) = conversion via specified model
6.2.1 Real to integer conversion

The straightforward mapping for this case is
\[ \Gamma_n : \mathcal{R} \to \mathcal{I}_n , \; x \mapsto \text{round}(n \cdot x, 0) = \left\lfloor \frac{1}{2} + n \cdot x \right\rfloor \] (16)
where \( \text{round}(r, d) \) rounds the real number \( r \) to \( d \geq 0 \) decimal digits. This mapping nearly always preserves complements, as shown in the next lemma.

**Lemma 1** (Preservation of complements). For \( x \in \mathcal{R} \),
\[ \Gamma_n(x) + \Gamma_n(1 - x) = n \iff x \notin \mathcal{R}_n^c := \{ \frac{\nu}{n} \mid \nu = 1, 2, \ldots, n \} . \] (17)

**Proof.** Let \( \nu := \Gamma_n(x) \), then from \( -\frac{1}{2} \leq \eta := n \cdot x - \nu < \frac{1}{2} \) we conclude
\[ \Gamma_n(1 - x) = \text{round}(n(1 - x), 0) = \text{round}(n - \nu - \eta, 0) = \begin{cases} n - \nu & \text{if } \eta \neq -\frac{1}{2} \\ n - \nu + 1 & \text{if } \eta = -\frac{1}{2} \end{cases} \]
Now, \( \eta = -\frac{1}{2} \iff x = \frac{1}{n}(\nu - \frac{1}{2}) \iff x \in \mathcal{I}_n' \). \( \square \)

Remark: the set \( \mathcal{R}_n^c \) is obviously identical to the set of points where \( \Gamma_n \) is not continuous.

6.2.2 Integer to real conversion

The straightforward way in this case is the function
\[ \Delta_n^* : \mathcal{I}_n \to \mathcal{R} , \; \nu \mapsto \frac{\nu}{n} . \] (18)
This is, however, only one out of a variety of solutions: every function \( \Delta_n : \mathcal{I}_n \to \mathcal{R} \) that obeys the condition
\[ \nu \in \mathcal{I}_n \Rightarrow \Gamma_n(\Delta_n(\nu)) = \nu \] (19)
which is equivalent to
\[ \nu \in \mathcal{I}_n \Rightarrow \nu + \frac{1}{2} > n \cdot \Delta_n(\nu) \geq \nu - \frac{1}{2} \] (20)
does at least guarantee that all integers \( \nu \) may be reconstructed from \( \Delta_n(\nu) \) via multiplication by \( n \) and rounding to the nearest integer. Preservation of complements means now
\[ \nu \in \mathcal{I}_n \Rightarrow \Delta_n(\nu) + \Delta_n(n - \nu) = 1 \] (21)
which is obviously the case for \( \Delta_n = \Delta_n^* \). If we consider, more generally, a transformation
\[ \Delta_n(\nu) = \frac{\nu + \alpha}{n + \beta} \] (22)
with \( \beta \neq -n \), then the magic inequality (20) is equivalent to
\[ \frac{1}{2} > \frac{\alpha n - \beta \nu}{n + \beta} \geq -\frac{1}{2} \] (23)
which is obeyed by the function

$$\Delta'_n : \mathcal{I}_n \to \mathbb{R}, \, \nu \mapsto \begin{cases} \frac{\nu}{n+1} & \text{if } \nu \leq \frac{n+1}{2} \\ \frac{n+1}{n+1} & \text{if } \nu > \frac{n+1}{2} \end{cases}$$

that has the nice feature $\Delta'_n\left(\frac{n+1}{2}\right) = \frac{1}{2}$ for odd $n$.

**Lemma 2** (Preservation of complements). For odd $n$ and each $\nu \in \mathcal{I}_n$,\[\Delta'_n(\nu) + \Delta'_n(n - \nu) = 1 \iff \nu \notin \mathcal{T}_n^0 := \left\{ \frac{n-1}{2}, \frac{n+1}{2} \right\}.\]

**Proof.** The assertion is a consequence of the following arguments:
- $\nu < \frac{n-1}{2} \iff n - \nu > \frac{n+1}{2}$ and $\frac{n-1}{2} + \frac{n+1}{2} = n$;
- $\nu < \frac{n-1}{2} \Rightarrow \Delta'_n(\nu) + \Delta'_n(n - \nu) = \frac{\nu}{n+1} + \frac{n-\nu+1}{n+1} = 1$;
- $\nu = \frac{n-1}{2} \Rightarrow \Delta'_n(\nu) + \Delta'_n(n - \nu) = \frac{n}{2(n+1)} + \frac{1}{2} = \frac{n}{n+1} \neq 1$.

For the time being, we choose $\Delta_n := \Delta'_n$ as default transformation function.

Another variant — which is probably too slow for large-scale on-the-fly calculations — may be used for constructing sets of predefined colors. The basic idea is to minimize the number of decimal digits in the representation while keeping some invariance with respect to the original resolution:

$$\Delta''_n : \mathcal{I}_n \to \mathbb{R}, \, \nu \mapsto \text{round} \left( \nu \left( \frac{n}{n+1} \right) \right)$$

where

$$d_n : [0, 1] \to \mathbb{N}, \, x \mapsto \min \{ d \in \mathbb{N} \mid \Gamma_n \left( \text{round}(\Delta''_n(\Gamma_n(x))), d \right) = \Gamma_n(x) \} \quad (27)$$

In the most common case $n = 255$ it turns out that we end up with at most 3 decimal digits; preservation of complements is only violated for $\nu \in \{25, 26, 76, 77, 127, 128, 178, 179, 229, 230\}$ where the corresponding set of decimal numbers is $\{0.098, 0.1, 0.298, 0.3, 0.498, 0.5, 0.698, 0.7, 0.898, 0.9\}$.

### 6.3 Color conversion and complements

We collect here the specific conversion formulas between the supported color models. Table 8 on page 45 gives an overview of how each conversion pair is handled. In general, PostScript (as described in [1]) is used as a basis for most of the calculations, since it supports the color models rgb, cmyk, hsv, and gray natively.


First, we define a constant which is being used throughout the conversion formulas:

$$E := (1, 1, 1) \quad (28)$$

### 6.3.1 The rgb model

**Conversion rgb to cmy** Source: [1], p. 475.

$$(\text{cyan, magenta, yellow}) := E - (\text{red, green, blue}) \quad (29)$$
Conversion rgb to hsb (1)  We set

\[ x := \max\{\text{red}, \text{green}, \text{blue}\} \quad (30) \]

\[ y := \text{med}\{\text{red}, \text{green}, \text{blue}\} \quad (31) \]

\[ z := \min\{\text{red}, \text{green}, \text{blue}\} \quad (32) \]

where ‘med’ denotes the median of the values. Then,

\[ \text{brightness} := x \quad (34) \]

Case \( x = z \):

\[ \text{saturation} := 0 \quad (35) \]
\[ \text{hue} := 0 \quad (36) \]

Case \( x \neq z \):

\[ \text{saturation} := \frac{x - z}{x} \quad (37) \]

\[ f := \frac{x - y}{x - z} \quad (38) \]

\[ \text{hue} := \frac{1}{6} \cdot \begin{cases} 
1 - f & \text{if } x = \text{red} \geq \text{green} \geq \text{blue} = z \\
1 + f & \text{if } x = \text{green} \geq \text{red} \geq \text{blue} = z \\
3 - f & \text{if } x = \text{green} \geq \text{blue} \geq \text{red} = z \\
3 + f & \text{if } x = \text{blue} \geq \text{green} \geq \text{red} = z \\
5 - f & \text{if } x = \text{blue} \geq \text{red} \geq \text{green} = z \\
5 + f & \text{if } x = \text{red} \geq \text{blue} > \text{green} = z 
\end{cases} \quad (39) \]

This is based on [15], RGB to HSV Algorithm (Hexcone Model), which reads (slightly reformulated):

\[ r := \frac{x - \text{red}}{x - z}, \quad g := \frac{x - \text{green}}{x - z}, \quad b := \frac{x - \text{blue}}{x - z} \quad (40) \]

\[ \text{hue} := \frac{1}{6} \cdot \begin{cases} 
5 + b & \text{if } \text{red} = x \text{ and } \text{green} = z \\
1 - g & \text{if } \text{red} = x \text{ and } \text{green} > z \\
1 + r & \text{if } \text{green} = x \text{ and } \text{blue} = z \\
3 - b & \text{if } \text{green} = x \text{ and } \text{blue} > z \\
3 + g & \text{if } \text{blue} = x \text{ and } \text{red} = z \\
5 - r & \text{if } \text{blue} = x \text{ and } \text{red} > z 
\end{cases} \quad (41) \]

Note that the singular case \( x = z \) is not covered completely in Smith’s original algorithm; we stick here to PostScript’s behaviour in real life.

Because we need to sort three numbers in order to calculate \( x, y, z \), several comparisons are involved in the algorithm. We present now a second method which is more suited for \( \TeX \).
Conversion rgb to hsb (2) Let $\beta$ be a function that takes a Boolean expression as argument and returns 1 if the expression is true, 0 otherwise; set

$$i := 4 \cdot \beta(red \geq green) + 2 \cdot \beta(green \geq blue) + \beta(blue \geq red),$$

(42)

and

$$(hue, saturation, brightness) := \begin{cases} 
\Phi(blue, green, red, 3, 1) & \text{if } i = 1 \\
\Phi(green, red, blue, 1, 1) & \text{if } i = 2 \\
\Phi(green, blue, red, 3, -1) & \text{if } i = 3 \\
\Phi(red, green, blue, 5, 1) & \text{if } i = 4 \\
\Phi(blue, red, green, 5, -1) & \text{if } i = 5 \\
\Phi(red, green, blue, 1, -1) & \text{if } i = 6 \\
(0, 0, blue) & \text{if } i = 7 
\end{cases}$$

(43)

where

$$\Phi(x, y, z, u, v) := \frac{u \cdot (x - z) + v \cdot (x - y)}{6(x - z)}, \frac{x - z}{x}.$$ 

(44)

The singular case $x = z$, which is equivalent to $red = green = blue$, is covered here by $i = 7$.

It is not difficult to see that this algorithm is a reformulation of the previous method. The following table explains how the transition from equation (39) to equation (43) works:

<table>
<thead>
<tr>
<th>Condition</th>
<th>red $\geq$ green</th>
<th>green $\geq$ blue</th>
<th>blue $\geq$ red</th>
<th>$i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 - f$</td>
<td>1</td>
<td>1</td>
<td>*</td>
<td>6/7</td>
</tr>
<tr>
<td>$1 + f$</td>
<td>*</td>
<td>1</td>
<td>*</td>
<td>2/3/6/7</td>
</tr>
<tr>
<td>$3 - f$</td>
<td>*</td>
<td>1</td>
<td>1</td>
<td>3/7</td>
</tr>
<tr>
<td>$3 + f$</td>
<td>*</td>
<td>*</td>
<td>1</td>
<td>1/3/5/7</td>
</tr>
<tr>
<td>$5 - f$</td>
<td>1</td>
<td>*</td>
<td>1</td>
<td>5/7</td>
</tr>
<tr>
<td>$5 + f$</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>4/5/6/7</td>
</tr>
</tbody>
</table>

Here, * denotes possible 0 or 1 values. Bold $i$ values mark the main cases where all * values of a row are zero. The slight difference to equation (39) in the last inequality is intentional and does no harm.

Conversion rgb to gray Source: [1], p. 474.

$$gray := 0.3 \cdot red + 0.59 \cdot green + 0.11 \cdot blue$$

(45)

Conversion rgb to RGB As described in section 6.2.1 on page 46.

$$(Red, Green, Blue) := (\Gamma_L(red), \Gamma_L(green), \Gamma_L(blue))$$

(46)

Conversion rgb to HTML As described in section 6.2.1 on page 46. Convert to 6-digit hexadecimal afterwards. Certainly, multiplication and summation can be replaced by simple text concatenation of 2-digit hexadecimals.

$$RRGGBB := (65536 \cdot \Gamma_L(red) + 256 \cdot \Gamma_L(green) + \Gamma_L(blue))_{hex}$$

(47)
Complement of rgb color  We simply take the complementary vector:

\[(\text{red}^*, \text{green}^*, \text{blue}^*) := E - (\text{red}, \text{green}, \text{blue})\]  \hspace{1cm} (48)

6.3.2 The cmy model

Conversion cmy to rgb  This is simply a reversion of the rgb $\rightarrow$ cmy case, cf. section 6.3.1 on page 47.

\[(\text{red}, \text{green}, \text{blue}) := E - (\text{cyan}, \text{magenta}, \text{yellow})\]  \hspace{1cm} (49)

Conversion cmy to cmyk  This is probably the hardest of our conversion tasks: many sources emphasize that there does not exist any universal conversion algorithm for this case because of device-dependence. The following algorithm is an extended version of the one given in [1], p. 476.

\[
k := \min\{\text{cyan}, \text{magenta}, \text{yellow}\} \hspace{1cm} (50)
\]

\[
cyan := \min\{1, \max\{0, \text{cyan} - \text{UCR}_c(k)\}\} \hspace{1cm} (51)
\]

\[
magenta := \min\{1, \max\{0, \text{magenta} - \text{UCR}_m(k)\}\} \hspace{1cm} (52)
\]

\[
yellow := \min\{1, \max\{0, \text{yellow} - \text{UCR}_y(k)\}\} \hspace{1cm} (53)
\]

\[
\text{black} := \text{BG}(k) \hspace{1cm} (54)
\]

Here, four additional functions are required:

\[\text{UCR}_c, \text{UCR}_m, \text{UCR}_y : [0, 1] \rightarrow [-1, 1]\]  \hspace{1cm} undercolor-removal

\[\text{BG} : [0, 1] \rightarrow [0, 1]\]  \hspace{1cm} black-generation

These functions are device-dependent, see the remarks in [1]. Although there are some indications that they should be chosen as nonlinear functions, as long as we have no further knowledge about the target device we define them linearly:

\[
\text{UCR}_c(k) := \beta_c \cdot k \hspace{1cm} (55)
\]

\[
\text{UCR}_m(k) := \beta_m \cdot k \hspace{1cm} (56)
\]

\[
\text{UCR}_y(k) := \beta_y \cdot k \hspace{1cm} (57)
\]

\[
\text{BG}(k) := \beta_k \cdot k \hspace{1cm} (58)
\]

\adjustUcrBg  where the parameters are given by \adjustUcrBg\{\beta_\text{c}, \beta_\text{m}, \beta_\text{y}, \beta_\text{k}\} at any point in a document, defaulting to \{1, 1, 1, 1\}.

Conversion cmy to gray  This is derived from the conversion chain cmy $\rightarrow$ rgb $\rightarrow$ gray.

\[\text{gray} := 1 - (0.3 \cdot \text{cyan} + 0.59 \cdot \text{magenta} + 0.11 \cdot \text{yellow})\]  \hspace{1cm} (59)

Complement of cmy color  We simply take the complementary vector:

\[(\text{cyan}^*, \text{magenta}^*, \text{yellow}^*) := E - (\text{cyan}, \text{magenta}, \text{yellow})\]  \hspace{1cm} (60)
6.3.3 The cmyk model

**Conversion cmyk to cmy** Based on [1], p. 477, in connection with rgb → cmy conversion.

\[
\begin{align*}
cyan & := \min\{1, \text{cyan + black}\} \\
magenta & := \min\{1, \text{magenta + black}\} \\
yellow & := \min\{1, \text{yellow + black}\}
\end{align*}
\] (61-63)

**Conversion cmyk to gray** Source: [1], p. 475.

\[
\text{gray} := 1 - \min\{1, 0.3 \cdot \text{cyan + 0.59 \cdot magenta + 0.11 \cdot yellow + black}\}
\] (64)

**Complement of cmyk color** The simple vector complement does not yield useful results. Therefore, we first convert \( C = (\text{cyan, magenta, yellow, black}) \) to the cmy model, calculate the complement there, and convert back to cmyk.

6.3.4 The hsb model

**Conversion hsb to rgb**

\[
(red, green, blue) := \text{brightness} \cdot (E - \text{saturation} \cdot F)
\] (65)

with

\[
i := \lfloor 6 \cdot \text{hue} \rfloor, \quad f := 6 \cdot \text{hue} - i
\] (66)

and

\[
F := \begin{cases} 
(0, 1 - f, 1) & \text{if } i = 0 \\
(f, 0, 1) & \text{if } i = 1 \\
(1, 0, 1 - f) & \text{if } i = 2 \\
(1, f, 0) & \text{if } i = 3 \\
(1 - f, 1, 0) & \text{if } i = 4 \\
(0, 1, f) & \text{if } i = 5 \\
(0, 1, 1) & \text{if } i = 6 
\end{cases}
\] (67)

This is based on [15], HSV to RGB Algorithm (Hexcone Model), which reads (slightly reformulated):

\[
m := 1 - \text{saturation}
\] (68)

\[
n := 1 - f \cdot \text{saturation}
\] (69)

\[
k := 1 - (1 - f) \cdot \text{saturation}
\] (70)

\[
(red, green, blue) := \text{brightness} \cdot \begin{cases} 
(1, k, m) & \text{if } i = 0, 0 \\
(n, 1, m) & \text{if } i = 1 \\
(m, 1, k) & \text{if } i = 2 \\
(m, n, 1) & \text{if } i = 3 \\
(k, m, 1) & \text{if } i = 4 \\
(1, m, n) & \text{if } i = 5 
\end{cases}
\] (71)
Note that the case $i = 6$ (which results from $\text{hue} = 1$) is missing in Smith’s algorithm. Because of
\[
\lim_{f \to 1} (0, 1, f) = (0, 1, 1) = \lim_{f \to 0} (0, 1 - f, 1)
\] (72)
it is clear that there is only one way to define $F$ for $i = 6$ in order to get a continuous function, as shown in equation (67). This has been transformed back to equation (71). A similar argument shows that $F$ indeed is a continuous function of $\text{hue}$ over the whole range $[0, 1]$.

**Conversion hsb to Hsb** Only the first component has to be changed.
\[
(hue^*, \text{saturation}, \text{brightness}) := (H \cdot hue, \text{saturation}, \text{brightness})
\] (73)

**Conversion hsb to HSB** As described in section 6.2.1 on page 46.
\[
(Hue, \text{Saturation}, \text{Brightness}) := (\Gamma_M(hue), \Gamma_M(\text{saturation}), \Gamma_M(\text{brightness}))
\] (74)

**Complement of hsb color** We have not found a formula in the literature, therefore we give a short proof afterwards.

**Lemma 3.** The hsb-complement can be calculated by the following formulas:
\[
hue^* := \begin{cases} 
\text{hue} + \frac{1}{2} & \text{if } \text{hue} < \frac{1}{2} \\
\text{hue} - \frac{1}{2} & \text{if } \text{hue} \geq \frac{1}{2}
\end{cases}
\] (75)

\[
\text{brightness}^* := 1 - \text{brightness} \cdot (1 - \text{saturation})
\] (76)

\[
\text{saturation}^* := \begin{cases} 
0 & \text{if } \text{brightness}^* = 0 \\
\frac{\text{brightness} \cdot \text{saturation}}{\text{brightness}^*} & \text{if } \text{brightness}^* \neq 0
\end{cases}
\] (77)

**Proof.** Starting with the original color $C = (h, s, b)$, we define color $C^* = (h^*, s^*, b^*)$ by the given formulas, convert both $C$ and $C^*$ to the rgb model and show that
\[
C_{rgb} + C^*_{rgb} = b \cdot (E - s \cdot F) + b^* \cdot (E - s^* \cdot F^*) = E,
\] (78)
which means that $C_{rgb}$ is the complement of $C^*_{rgb}$. First we note that the parameters of $C^*$ are in the legal range $[0, 1]$. This is obvious for $h^*, b^*$. From $b^* = 1 - b \cdot (1 - s) = 1 - b + b \cdot s$ we derive $b \cdot s = b^* - (1 - b) \leq b^*$, therefore $s^* \in [0, 1]$, and
\[
b^* = 0 \Leftrightarrow s = 0 \text{ and } b = 1.
\]

Thus, equation (78) holds in the case $b^* = 0$. Now we assume $b^* \neq 0$, hence
\[
C_{rgb} + C^*_{rgb} = b \cdot (E - s \cdot F) + b^* \cdot \left(E - \frac{b \cdot s}{b^*} \cdot F^*\right)
= b \cdot E - b \cdot s \cdot F + b^* \cdot E - b \cdot s \cdot F^*
= E - b \cdot s \cdot (F + F^* - E)
\]
since $b^* = 1 - b + bs$. Therefore, it is sufficient to show that
\[ F + F^* = E. \] (79)

From
\[ h < \frac{1}{2} \Rightarrow h^* = h + \frac{1}{2} \Rightarrow 6h^* = 6h + 3 \Rightarrow i^* = i + 3 \quad \text{and} \quad f^* = f \]
it is easy to see from (67) that equation (79) holds for the cases $i = 0, 1, 2$. Similarly,
\[ h \geq \frac{1}{2} \Rightarrow h^* = h - \frac{1}{2} \Rightarrow 6h^* = 6h - 3 \Rightarrow i^* = i - 3 \quad \text{and} \quad f^* = f \]
and again from (67) we derive (79) for the cases $i = 3, 4, 5$. Finally, if $i = 6$ then $f = 0$ and $F + F^* = (0, 1, 1) + (1, 0, 0) = E$.

6.3.5 The Hsb model
Conversion Hsb to hsb
Only the first component has to be changed.
\[ (\text{hue}, \text{saturation}, \text{brightness}) := (\text{hue}^c / H, \text{saturation}, \text{brightness}) \] (80)

Conversion Hsb to tHsb
Under the settings of (82)–(84) we simply have to exchange the letters $x$ and $y$ in equation (85) to get the inverse transformation:
\[ \text{hue}^c \in [y_{n-1}, y_n] \Rightarrow \text{hue}^c := x_{n-1} + \frac{x_n - x_{n-1}}{y_n - y_{n-1}} \cdot (\text{hue}^c - y_{n-1}) \] (81)
while saturation and brightness are left unchanged.

6.3.6 The tHsb model
\(\text{rangeHsb}\) Conversion tHsb to Hsb
We assume that \(\text{rangeHsb} = H\) and \(\text{rangetHsb}\) expands to
\[ x_1, y_1; x_2, y_2; \ldots; x_{n-1}, y_{n-1} \] (82)
where
\[ x_0 := 0 < x_1 < x_2 < \cdots < x_{n-1} < x_h := H \] (83)
\[ y_0 := 0 < y_1 < y_2 < \cdots < y_{n-1} < y_h := H \] (84)
with an integer $h > 0$. Now the $x$ and $y$ values determine a piecewise linear transformation:
\[ \text{hue}^c \in [x_{n-1}, x_n] \Rightarrow \text{hue}^c := y_{n-1} + \frac{y_n - y_{n-1}}{x_n - x_{n-1}} \cdot (\text{hue}^c - x_{n-1}) \] (85)
while saturation and brightness are left unchanged.

6.3.7 The gray model
Conversion gray to rgb
Source: [1], p. 474.
\[ (\text{red}, \text{green}, \text{blue}) := \text{gray} \cdot E \] (86)
Conversion gray to cmy  This is derived from the conversion chain gray → rgb → cmy.

\[(cyan, magenta, yellow) := (1 - \text{gray}) \cdot E\]  \hspace{1cm} (87)

Conversion gray to cmyk  Source: [1], p. 475.

\[(cyan, magenta, yellow, black) := (0, 0, 1 - \text{gray})\]  \hspace{1cm} (88)

Conversion gray to hsb  This is derived from the conversion chain gray → rgb → hsb.

\[(\text{hue}, \text{saturation}, \text{brightness}) := (0, 0, \text{gray})\]  \hspace{1cm} (89)

Conversion gray to Hsb/tHsb  This is derived from the conversion chain gray → hsb → Hsb, followed by Hsb → tHsb if applicable.

\[(\text{hue}^\circ, \text{saturation}, \text{brightness}) := (0, 0, \text{gray})\]  \hspace{1cm} (90)

Conversion gray to Gray  As described in section 6.2.1 on page 46.

\[\text{Gray} := \Gamma_N(\text{gray})\]  \hspace{1cm} (91)

Complement of gray color  This is similar to the rgb case:

\[\text{gray}^* := 1 - \text{gray}\]  \hspace{1cm} (92)

6.3.8  The RGB model

Conversion RGB to rgb  As described in section 6.2.2 on page 46.

\[(\text{red}, \text{green}, \text{blue}) := (\Delta_L(\text{Red}), \Delta_L(\text{Green}), \Delta_L(\text{Blue}))\]  \hspace{1cm} (93)

6.3.9  The HTML model

Conversion HTML to rgb  As described in section 6.2.2 on page 46: starting with RRGGBB set

\[(\text{red}, \text{green}, \text{blue}) := (\Delta_{255}(RR_{dec}), \Delta_{255}(GG_{dec}), \Delta_{255}(BB_{dec}))\]  \hspace{1cm} (94)

6.3.10  The HSB model

Conversion HSB to hsb  As described in section 6.2.2 on page 46.

\[(\text{hue}, \text{saturation}, \text{brightness}) := (\Delta_M(Hue), \Delta_M(Saturation), \Delta_M(Brightness))\]  \hspace{1cm} (95)

6.3.11  The Gray model

Conversion Gray to gray  As described in section 6.2.2 on page 46.

\[\text{gray} := \Delta_N(\text{Gray})\]  \hspace{1cm} (96)
6.3.12 The wave model

Conversion wave to rgb  Source: based on Dan Bruton’s algorithm [4]. Let \( \lambda \) be a visible wavelength, given in nanometers (nm), i.e., \( \lambda \in [380, 780] \). We assume further that \( \gamma > 0 \) is a fixed number (\( \gamma = 0.8 \) in [4]). First set

\[
(r, g, b) := \begin{cases} 
\left( \frac{440 - \lambda}{440 - 380}, 0, 1 \right) & \text{if } \lambda \in [380, 440[ \\
(0, \frac{\lambda - 440}{490 - 440}, 1) & \text{if } \lambda \in [440, 490[ \\
(0, 1, \frac{510 - \lambda}{510 - 490}) & \text{if } \lambda \in [490, 510[ \\
\left( \frac{\lambda - 510}{580 - 510}, 1, 0 \right) & \text{if } \lambda \in [510, 580[ \\
(1, \frac{645 - \lambda}{645 - 580}, 0) & \text{if } \lambda \in [580, 645[ \\
(1, 0, 0) & \text{if } \lambda \in [645, 780] 
\end{cases} 
\tag{97}
\]

then, in order to let the intensity fall off near the vision limits,

\[
f := \begin{cases} 
0.3 + 0.7 \cdot \frac{\lambda - 380}{420 - 380} & \text{if } \lambda \in [380, 420[ \\
1 & \text{if } \lambda \in [420, 700[ \\
0.3 + 0.7 \cdot \frac{780 - \lambda}{780 - 700} & \text{if } \lambda \in [700, 780] 
\end{cases} 
\tag{98}
\]

and finally

\[
(red, green, blue) := ((f \cdot r)^\gamma, (f \cdot g)^\gamma, (f \cdot b)^\gamma) 
\tag{99}
\]

The intermediate colors \((r, g, b)\) at the interval borders of equation (97) are well-known: for \( \lambda = 380, 440, 490, 510, 580, 645 \) we get magenta, blue, cyan, green, yellow, red, respectively. These turn out to be represented in the hsb model by hue = \(\frac{3}{6}, \frac{4}{6}, \frac{5}{6}, \frac{2}{6}, \frac{1}{6}, \frac{0}{6}\), whereas saturation = brightness = 1 throughout the 6 colors. Furthermore, these hsb representations are independent of the actual \( \gamma \) value. Staying within this model framework, we observe that the intensity fall off near the vision limits — as represented by equation (98) — translates into decreasing brightness parameters towards the margins. A simple calculation shows that the edges \( \lambda = 380, 780 \) of the algorithm yield the colors magenta!0.3!black, red!0.3!black, respectively. We see no reason why we should not extend these edges in a similar fashion to end-up with true black on either side. Now we are prepared to translate everything into another, more natural algorithm.

Conversion wave to hsb  Let \( \lambda > 0 \) be a wavelength, given in nanometers (nm), and let

\[
g : \mathbb{R} \to [0, 1], \ x \mapsto (\min\{1, \max\{0, x\}\})^\gamma 
\tag{100}
\]
with a fixed correction number $\gamma > 0$. Then

$$
\text{hue} := \frac{1}{6} \cdot 
\begin{cases}
4 + \varphi \left(\frac{\lambda - 440}{380 - 440}\right) & \text{if } \lambda < 440 \\
4 - \varphi \left(\frac{\lambda - 440}{490 - 440}\right) & \text{if } \lambda \in [440, 490[ \\
2 + \varphi \left(\frac{\lambda - 510}{490 - 510}\right) & \text{if } \lambda \in [490, 510[ \\
2 - \varphi \left(\frac{\lambda - 510}{580 - 510}\right) & \text{if } \lambda \in [510, 580[ \\
0 + \varphi \left(\frac{\lambda - 645}{580 - 645}\right) & \text{if } \lambda \in [580, 645[ \\
0 & \text{if } \lambda \geq 645
\end{cases}
$$

(101)

$$
\text{saturation} := 1
$$

(102)

$$
\text{brightness} := 
\begin{cases}
\varphi \left(0.3 + 0.7 \cdot \frac{\lambda - 380}{420 - 380}\right) & \text{if } \lambda < 420 \\
1 & \text{if } \lambda \in [420, 700[ \\
\varphi \left(0.3 + 0.7 \cdot \frac{\lambda - 780}{700 - 780}\right) & \text{if } \lambda > 700
\end{cases}
$$

(103)

For the sake of completeness we note that, independent of $\gamma$,

$$(\text{hue, saturation, brightness}) = 
\begin{cases}
\left(\frac{2}{5}, 1, 0\right) & \text{if } \lambda \leq 380 - \frac{3(420 - 380)}{7} = 362.857 \ldots \\
\left(0, 1, 0\right) & \text{if } \lambda \geq 780 + \frac{3(780 - 700)}{7} = 814.285 \ldots
\end{cases}
$$

What is the best (or, at least, a good) value for $\gamma$? In the original algorithm [4], $\gamma = 0.8$ is chosen. However, we could not detect significant visible difference between the cases $\gamma = 0.8$ and $\gamma = 1$. Thus, for the time being, xcolor’s implementation uses the latter value which implies a pure linear approach. In the \texttt{pstricks} examples file \texttt{xcolor2.tex}, there is a demonstration of different $\gamma$ values.
References


[2] Donald Arseneau: “Patch so \fbox draws frame on top of text”. \LaTeX\ bug report, latex/3655, 2004/03/18.
http://www.latex-project.org/cgi-bin/ltxbugs2html?pr=latex/3655


http://www.physics.sfasu.edu/astro/color/spectra.html

CTAN/macros/latex/required/graphics/grfguide.*

[6] David P. Carlisle: \texttt{color} package, “2016/01/03 v1.1b Standard \LaTeX\ Color (DPC)”.
CTAN/macros/latex/contrib/color/color.dtx

[7] David P. Carlisle: \texttt{colortbl} package, “2001/02/13 v0.1j Color table columns”.
CTAN/macros/latex/contrib/colortbl/


[10] Uwe Kern: \texttt{xcolor} package, “\LaTeX\ color extensions”.
CTAN/macros/latex/contrib/xcolor/ and http://www.ukern.de/tex/xcolor.html


[12] Rolf Niepraschk: \texttt{colorinfo} package, “2003/05/04 v0.3c Info from defined colors”.
CTAN/macros/latex/contrib/colorinfo/

[13] Heiko Oberdiek: \texttt{pdfcolmk} package, “2006/02/20 v0.8 PDF\texttt{tex} COLor MarK”.
CTAN/macros/latex/contrib/oberdiek/pdfcolmk.*

[14] Sebastian Rahtz, Heiko Oberdiek: \texttt{hyperref} package, “2006/09/06 v6.75e Hypertext links for \LaTeX\”.
CTAN/macros/latex/contrib/oberdiek/hyperref/


Appendix

Acknowledgement

This package is based on and contains code copied from [6] (Copyright (C) 1994–1999 David P. Carlisle), which is part of the Standard \LaTeXX ‘Graphics Bundle’. Although many commands and features have been added and most of the original color commands have been rewritten or adapted within xcolor, the latter package would not exist without color. Thus, the author is grateful to David P. Carlisle for having created color and its accompanying files.

Trademarks

Trademarks appear throughout this documentation without any trademark symbol; they are the property of their respective trademark owner. There is no intention of infringement; the usage is to the benefit of the trademark owner.

Known Issues

- \rowcolors[hline]... does not work with longtable.

History

2023/11/15 v3.01
- only warn and not error if obsolete option usenames is used (issue #31)

2023/11/11 v3.00
- if a \LaTeXX older than 2022-06-01 is detected a frozen version is used. All new features apply only if \LaTeXX 2022-06-01 or newer is detected. The frozen version can also be requested with \usepackage{xcolor}[=v2]
- The new \LaTeXX key/val handler is used for the package options. That means there will be fewer option clash errors (issue #29).
- The \rowcolors command has moved to the colortbl package and is no longer defined by xcolor.
- The fixpdfex option is obsolete and does nothing at all (issue #20)
- Extended \normalcolor to set also the dot color, issue #19
- Fix Gray color model, issue #23

2022/06/12 v2.14
- Load code from \LaTeXX to define \mathcolor if it is available.
- Bug fixes:
  - Modify \XC@info to avoid spurious error from \colorlet{foo}{}, issue #12
  - make \normalcolor, \colorbox, \fcolorbox, \textcolor, \pagecolor, \nopagecolor, \definecolor, \DefineNamedColor protected as in color.sty, issue #17

2021/10/31 v2.13
- Bug Fixes:
  - Avoid use of \string to generate csnames
  - Declare dvisvgm option and make dvipdfm an alias to dvipdfmx to match graphics and color packages.
  - Modify \XC@info to avoid spurious text from \colorlet{foo}{.150}

2016/05/11 v2.12
- New features:
  - \nopagecolor command as introduced in color v1.1a (example added to xcolor3.tex);
  - \luatex driver option (code provided by DPC) to fix incompatibilities due to changes in new Lua\LaTeXX version.
- Bugfixes:
  - possible name conflict by \XC@ifxcase call;
  - incorrect internal \@hex@@Hex macro.
2007/01/21 v2.11

- New features:
  - color names *lime* and *teal* added to the set of predefined colors.

- Bugfix:
  - incorrect \XC@strip@comma call within hyperref-related options.

2006/11/28 v2.10

- New features:
  - \fixincude option prevents dvips from explicitly resetting current color to black before actually inserting an .eps file via \color{red}\includegraphics{foo}.

- Changes:
  - \colorbox and \fcolorbox made robust;
  - obsolete package option pst removed;
  - several changes to internal macros.

- Bugfixes:
  - incorrect processing of *cmyk*-type current color .'

2005/12/21 v2.09

- New features:
  - more flexibility for \fcolorbox arguments, e.g., \colorbox [gray]{0.5}[wave]{580}{test};
  - \boxframe returns a frame of given dimensions;
  - new implementation of \framebox and \fcolorbox as an extension of bug report latex/3655 to reduce pixel positioning errors in output devices;
  - kernelfbox option for those who prefer the previous \framebox approach;
  - experimental xcdraw option uses PostScript commands to draw frames and color boxes in case of dvips.

- Changes:
  - test file xcolor2.tex made compatible with recent changes in pstricks;
  - test file xcolor3.tex extended;
  - driver test file xcolor4.tex extended to demonstrate the different frame drawing approaches;
  - more efficient implementation of driver-specific code.

2005/11/25 v2.08

- New features:
  - color model Hsb allows to specify hue in degrees;
  - color model tHsb (tuned Hsb) for user-defined hue configuration on color wheels;
  - easy generation of color harmonies derived from Hsb or tHsb color wheels, e.g., \color{red>wheel,1,12} yields an 'analogous' color to red on a 12-spoke wheel;
  - additional 317 predefined color names according to rgb.txt, which is part of Unix/X11 distributions;
  - svgnames option extended by 4 colors taken from rgb.txt;
  - enhanced syntax for immediate conversion, e.g., \definecolor {foo}{rgb:gray}{0.3} or \color [rgb:wave]{478};
  - \@ifundefinedcolor and \@ifundefinedmodel commands.

- Changes:
– enhanced documentation;
– several changes to internal macros.

• Bugfixes:
– wrong calculation of color series components in some cases of negative step parameters.

2005/10/15 v2.06

• New features:
– color model wave for (approximate) visualisation of light wavelengths, still somewhat experimental;
– pseudo-model ‘ps’ for colors defined by literal PostScript code in conjunction with pstricks and dvips; an illustrative example for a γ-correction approach is given in xcolor2.tex;
– \substitutecolormodel command for replacement of missing or faulty driver-specific color models;
– improved detection and handling of driver-specific color models;
– dvipdfmx and xetex options to support these drivers;
– generic driver test file xcolor4.tex.

• Changes:
– \XC@strip@comma doesn’t generate a trailing space anymore, which improves also the output of the testcolors environment.

2005/09/30 v2.05

• New features:
– testcolors environment helps to test colors in different models, showing both the visual result and the model-specific parameters;
– \extractcolorspecs puts model/color specification into two separate commands, as opposed to \extractcolorspec;
– color names pink and olive added to the set of predefined colors.

• Bugfixes:
– \definecolor{foo}{named}{bar} did not work in v2.04.

2005/09/23 v2.04

• New features:
– preparation for usage of additional – driver-provided – color models;
– pstricks users may now specify explicit color parameters within \psset and related commands, e.g., \psset{linecolor=[rgb]{1,0,0}}; an illustrative example is given in xcolor2.tex.

• Changes:
– color model names sanitized (i.e., turned to catcode 12) throughout the package;
– \@namelet command deprecated because of name clash with memoir — please use \XC@let@cc instead (more \XC@let@.. commands are available as well);
– simplified color conversion code by using the new \XC@ifxcase command;
– some minor changes to internal macros.

2005/06/06 v2.03

• New features:
– fixpdfexit option loads pdfcolmk package in order to improve pdffonts’s color behaviour during page breaks.

• Changes:
– some minor changes to internal macros.

• Bugfixes:
– due to an incorrect \if statement within \XC@info, \colorlet caused trouble whenever its second argument started with two identical letters, e.g., \colorlet{rab}{oof};
– argument processing of \XC@getcolor caused incompatibility with msc package;
– prologue option caused incompatibility with preview package.
2005/03/24 v2.02

- New features:
  - \aftergroupeddef command to reproduce \aftergroupeddef's behaviour prior to v2.01;
  - xcolor's homepage www.ukern.de/tex/xcolor.html now provides also a ready-to-run TDS-compliant archive containing all required files.

- Changes:
  - \rowcolors and friends are solely enabled by the table option;
  - \@ifixempty changed back to more robust variant of v2.00.

- Bugfixes:
  - \psset{linecolor=\ifcase\foo red\or green\or blue\fi} did not work with \pspicture (error introduced in v2.01).

2005/03/15 v2.01

- New features:
  - prologue option for comprehensive ‘named’ color support in conjunction with \dvips: on-the-fly generation of PostScript prologue files with all color definitions, ready for \dvips inclusion and/or post-processing with device-specific parameters (e.g., spot colors);
  - \dvips prologue file xcolor.pro to support additional ‘named’ colors;
  - \xglobal may now also be used to create named colors from arbitrary color expressions;
  - enhanced color definition syntax to allow for target-model specific color parameters, e.g., \definecolor{red}{rgb/cmyk}{1,0,0/0,1,1,0}, facilitating the usage of tailor-made colors both for displays and printers;
  - ‘deferred definition’ of colors: \preparecolor and \definecolors enable decoupling of color specification and control sequence generation, especially useful (= memory saving) for large lists of colors, of which only a few names are actually used;
  - dvipsnames* and svgnames* options to support deferred definition.

- Changes:
  - higher accuracy: most complement calculations are now exact for all 5-digit decimals;
  - \rangeRGB and similar variables may now be changed at any point in a document;
  - \aftergroupeddef now performs only a first-level expansion of its code argument;
  - \XFileversion and similar internal constants removed from .sty and .def files;
  - improved memory management (reduced generation of ‘multiletter control sequences’ by \@ifdefined tests);
  - several internal macros improved and/or renamed.

- Bugfixes:
  - \XC@getcolor could cause unwanted spaces when \psset was used inside pspicture environments (pstricks);
  - arithmetic overflow could happen when too many decimal digits were used within color parameters, e.g., as a result of fp calculations.

2004/07/04 v2.00

- New features:
  - extended functionality for color expressions: mix colors like a painter;
  - support for color blending: specify color mix expressions that are being blended with every displayed color;
  - \xglobal command for selective control of globality for color definitions, blends, and masks;
  - multiple step operations (e.g., \color{foo!!+++}) and access to individual members (e.g., \color{foo!![7]}) in color series;
  - \providecolor command to define only non-existent colors;
- \definecolorset and \providecolorset commands to facilitate the construction of color sets with common underlying color model;
- additional 147 predefined color names according to SVG 1.1 specification;
- \xpdfborder key for setting the width of hyperlink borders in a more driver-independent way if dvips is used.

**Changes:**
- color package now completely integrated within xcolor;
- override, usenames, nodvipsnames options and \xdefinecolor command no longer needed;
- dvips and dvipsnames options now independent of each other;
- \tracingcolors’s behaviour changed to make it more versatile and reduce log file size in standard cases;
- \divide’s syntax made more flexible (divide by numbers and/or dimensions);
- code restructured, some internal commands renamed;
- documentation rearranged and enhanced.

**Bugfixes:**
- \definecolor{foo}{named}{bar} did not work (error introduced in v1.11);
- more robust behaviour of conditionals within pstricks key-values.

2004/05/09 v1.11

**New features:**
- switch \ifglobalcolors to control whether color definitions are global or local;
- option hyperref provides color expression support for the border colors of hyperlinks, e.g., \hypersetup \xurlbordercolor=red!50!yellow;
- internal hooks \xcolorbc, \xcolormc, and \xcolorec for additional code that has to be executed immediately before/after the current color is being displayed.

**Changes:**
- \xclogcolor renamed to \xcolordisplay, which is now the core color display command;
- improved interface to pstricks.

2004/03/27 v1.10

**New features:**
- support for ‘named’ model;
- support for dvips colors (may now be used within color expressions);
- internal representation of ‘ordinary’ and ‘named’ colors merged into unified data structure;
- allow multiple ‘-’ signs at the beginning of color expressions.

**Bugfixes:**
- commands like \color[named]{foo} caused errors when color masking or target model conversion were active;
- incompatibility with soul package: commands \hl, \ul, etc. could yield unexpected results.

**Documentation:**
- added formula for general color expressions;
- enhanced text and index;
- removed dependence of index generation on local configuration file.

2004/02/16 v1.09

**New features:**
- color model HTML, a 24-bit hexadecimal RGB variant; allows to specify colors like \color[HTML]{AFFE90};
- color names orange, violet, purple, and brown added to the set of predefined colors.

**New xcolor homepage:**
www.ukern.de/tex/xcolor.html

**Bugfix:** \xdefinecolor sometimes did not normalise its parameters.

**Changes:**
- slight improvements of the documentation;
- example file xcolor1.tex reorganised and abridged.
2004/02/04 v1.08

- New commands:
  - \selectcolormodel to change the target model within a document;
  - \adjustUCRBG to fine-tune undercolor-removal and black-generation during conversion to cmyk.
- Bugfix: color expressions did not work correctly in connection with active ‘!’ character, e.g., in case of \usepackage{frenchb}babel}.
- Code re-organisation:
  - \XC@xdefinecolor merged into \xdefinecolor, making the first command obsolete;
  - several internal commands improved/streamlined.

2004/01/20 v1.07

- New feature: support for color masking and color separation.
- New commands:
  - \multiply to multiply a dimension register by a real number;
  - \xcolorcmd to pass commands that are to be executed at the end of the package.
- Changes:
  - more consistent color handling: extended colors now always take precedence over standard colors;
  - several commands improved by using code from the L\TeX kernel.
- Documentation: some minor changes.
- Example files: additional pstricks examples (file xcolor2.tex).

2003/12/15 v1.06

- New feature: extended color expressions, allowing for cascaded mix operations, e.g., \color{red!30!green!40!blue}.
- Documentation: new section on color expressions.
- Bugfix: color series stepping did not work correctly within non-displaying commands like \extractcolorspec{foo!!+} (this bug was introduced in v1.05).
- Renamed commands: \ukfileversion and similar internal constants renamed to \XCfileversion etc.
- Removed commands: \ifXCpst and \ifXCtable made obsolete by a simple trick.

2003/11/21 v1.05

- Bugfixes:
  - package option hideerrors should now work as expected;
  - usage of ‘.’ in the first color expression in a document caused an error due to incorrect initialisation.
- Code re-organisation: \extractcolorspec now uses \XC@splitcolor, making \XC@extract obsolete.

2003/11/09 v1.04

- New feature: easy access to current color within color expressions.
- New option: override to replace \definecolor by \xdefinecolor.
- New command: \tracingcolors for logging color-specific information.

2003/09/21 v1.03

- Change: bypass strange behaviour of some drivers.
- New feature: driver-sharing with hyperref.

2003/09/19 v1.02

- Change: \extractcolorspec and \colorlet now also accept color series as arguments.
2003/09/15 v1.01

- New feature: \definecolorseries and friends.
- Documentation: removed some doc-related side-effects.
- Code re-organisation: all calculation-related tools put to one place.
- Bugfixes:
  - \divide: added \relax to fix problem with negative numerators;
  - \rowcolors: replaced \ifempty by \ifxempty.

2003/09/09 v1.00

- First published release.

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>\adjustUCRGB</td>
<td>\blendcolors*</td>
</tr>
<tr>
<td>arguments</td>
<td></td>
</tr>
<tr>
<td>(color expr)</td>
<td>\boxframe</td>
</tr>
<tr>
<td>(color)</td>
<td></td>
</tr>
<tr>
<td>(core model)</td>
<td></td>
</tr>
<tr>
<td>(dec)</td>
<td></td>
</tr>
<tr>
<td>(dir)</td>
<td></td>
</tr>
<tr>
<td>(empty)</td>
<td></td>
</tr>
<tr>
<td>(expr)</td>
<td></td>
</tr>
<tr>
<td>(ext expr)</td>
<td></td>
</tr>
<tr>
<td>(ext id)</td>
<td></td>
</tr>
<tr>
<td>(func expression)</td>
<td></td>
</tr>
<tr>
<td>(func expr)</td>
<td></td>
</tr>
<tr>
<td>(function)</td>
<td></td>
</tr>
<tr>
<td>(id-list)</td>
<td></td>
</tr>
<tr>
<td>(id)</td>
<td></td>
</tr>
<tr>
<td>(int)</td>
<td></td>
</tr>
<tr>
<td>(minus)</td>
<td></td>
</tr>
<tr>
<td>(mix expr)</td>
<td></td>
</tr>
<tr>
<td>(model-list)</td>
<td></td>
</tr>
<tr>
<td>(model)</td>
<td></td>
</tr>
<tr>
<td>(name)</td>
<td></td>
</tr>
<tr>
<td>(num model)</td>
<td></td>
</tr>
<tr>
<td>(num)</td>
<td></td>
</tr>
<tr>
<td>(pet)</td>
<td></td>
</tr>
<tr>
<td>(plus)</td>
<td></td>
</tr>
<tr>
<td>(postfix)</td>
<td></td>
</tr>
<tr>
<td>(prefix)</td>
<td></td>
</tr>
<tr>
<td>(spec-list)</td>
<td></td>
</tr>
<tr>
<td>(spec)</td>
<td></td>
</tr>
<tr>
<td>(type)</td>
<td></td>
</tr>
<tr>
<td>\blendcolors</td>
<td>8, 50</td>
</tr>
<tr>
<td>\blendcolors*</td>
<td>24</td>
</tr>
<tr>
<td>\boxframe</td>
<td>23</td>
</tr>
<tr>
<td>\definecolorseries</td>
<td></td>
</tr>
</tbody>
</table>

\blendcolors*: added \relax to fix problem with negative numerators;
\rowcolors: replaced \ifempty by \ifxempty.
\convertcolorspec ....... 29

\definecolor ....... 19, 28
\definecolors ....... 21
\definecolorseries ....... 25
\definecolorset ....... 20
\DefineNamedColor ....... 21
definition stack ....... 21

E

environments:
  testcolors ....... 23
\extractcolorspec ....... 29
\extractcolors ....... 29

F
\fcolorbox ....... 22
files
  .def ....... 7, 61
  .dvi ....... 18, 19, 23, 28
  .eps ....... 9, 25, 30, 59
  .jpg ....... 25
  .pdf ....... 25
  .png ....... 25
  .ps ....... 19, 28, 30
  .sty ....... 61
  .xcp ....... 9, 18
color.pro ....... 18
dvipsnam.def ....... 21
rkg.txt ....... 17, 59
xcolor.pro ....... 7, 18, 61
xcolor.sty ....... 17
xcolor2.tex ....... 28, 56, 59, 60, 63
xcolor3.tex ....... 58, 59
xcolor4.tex ....... 59, 60

G
\GetGinDriver ....... 7
\GinDriver ....... 7

H
\hiderowcolors ....... 29
HKS ....... 5
HTML4 ....... 17, 39

I
\ifconvertcolorsD ....... 11
\ifconvertcolorsU ....... 11
\ifdefinecolors ....... 21
\ifglobalcolors ....... 21
\ifmaskcolors ....... 25

K

\keys
  citebordercolor ....... 27
citecolor ....... 27
filebordercolor ....... 27
filecolor ....... 27
linkbordercolor ....... 27
linkcolor ....... 27
menubordercolor ....... 27
menucolor ....... 27
pagebordercolor ....... 27
pagecolor ....... 27
pdfborder ....... 28
runbordercolor ....... 27uncolor ....... 27
uribordercolor ....... 27
urcolor ....... 27
xctebordercolor ....... 27, 28
xfilebordercolor ....... 27, 28
xlinkbordercolor ....... 27, 28
xmenubordercolor ....... 27, 28
xpagebordercolor ....... 27, 28
xpdfborder ....... 27, 28, 62
xrundbordercolor ....... 27, 28
xurlbordercolor ....... 27, 28

M
\maskcolors ....... 25
MiKTeX ....... 19

N
\nopagecolor ....... 22

P
package options
  Gray ....... 7, 9, 11
  HSB ....... 7, 9, 11
  HTML ....... 7, 9, 11
  RGB ....... 7, 9, 11
  cmyk ....... 7, 9, 11, 24
cmy ....... 7, 9, 11
dvipdfex ....... 7, 41, 60
dvipdfm ....... 7, 19, 23, 41, 59
dvipdf ....... 7, 41
dviplannames* ....... 7, 9, 17, 61
dviplannames ....... 7–9, 17–19, 30, 38, 62
dvipxone ....... 7, 41
dvips ....... 7, 41
  19, 25, 28, 41, 43, 59–62
dvipsave ....... 7
  dvivm ....... 7
dvivmbo ....... 7
dvivin ....... 7, 41
dvivext ....... 7, 41
  fixinclude ....... 7, 9, 30, 59
  fixpdfx ....... 9, 60
  gray ....... 7, 9, 11
  hideerrors ....... 7, 9, 63
  hsb ....... 7, 9, 11
  hyperref ....... 7, 9, 27, 28, 62
  hypertex ....... 7
  kernelbox ....... 7, 9, 23, 59
  latex ....... 7, 41, 58
  monochrome ....... 7, 41
  natural ....... 7, 9, 11, 12
  noxdraw ....... 7, 9, 23
  oxtex ....... 7, 41
  pctx32 ....... 7, 41
  pctxex ....... 7, 41
  pctxewin ....... 7, 41
  pdftex ....... 7, 19, 23, 32, 41, 43, 59
  prologue ....... 7, 9, 18, 19, 60, 61
  rgb ....... 7, 9, 11
  showerrors ....... 7, 9
  svgnames* ....... 7, 9, 17, 21, 61
svgnames ....... 7, 9, 17, 18, 21, 30, 38, 59
table ....... 7, 9, 10, 28, 61
tcldvi ....... 7, 41
textures ....... 7, 41
truetex ....... 7, 41
usecolors ....... 42
vtex ....... 7, 41
xlllannames* ....... 7, 9, 17, 39
xlllannames ....... 7, 9, 17, 39
xcdraw ....... 7, 9, 23, 59
xdvi ....... 7, 41
xetex ....... 7, 41, 60
package options (obsolete)
  fixpdfx ....... 7
  hyperref ....... 7
  nodvipsnames ....... 7, 62
  override ....... 7, 62, 63
  pst ....... 7, 59
  usenames ....... 7, 43, 62
packages
  colorinfo ....... 42, 57
  colorbl ....... 9, 10, 28, 57
color ....... 4,
  7, 9, 10, 18, 19, 21–23,
  27, 29, 41–43, 57, 58, 62
doc ....... 64
fp ....... 42, 61
graphics ....... 18, 30
graphicx ....... 30