

Élargissement des capacités de L^AT_EX en matière de couleur : l’extension xcolor

Dr. Uwe Kern

v2.11 (2007/01/21) *

Résumé

`xcolor` met à disposition, simplement et indépendamment des pilotes graphiques, à de multiples types de couleurs, teintes, nuances, tons et mélanges de couleurs arbitraires par le biais d’expressions dédiées comme `\color{red!50!green!20!blue}`. Il permet de sélectionner un modèle de couleur à l’échelle du document et offre des outils d’assortiment de couleurs automatiques, de conversion des couleurs entre douze modèles colorimétriques, d’utilisation de couleurs alternées pour des lignes de tableau, de mélange et de masque de couleur, de séparation de couleur et de calculs de cercle chromatique.

Table des matières

1	Introduction	5
1.1	Objectif de cette extension	5
1.2	Teintes, nuances, tons et couleurs complémentaires	6
1.3	Modèles colorimétriques	6
1.4	Cercles chromatiques et accords de couleurs	7
2	L’interface utilisateur	8
2.1	Préparation	8
2.1.1	Installation de l’extension	8
2.1.2	Options de l’extension	8
2.1.3	Executing additional initialisation commands	9
2.2	Color models	9
2.2.1	Supported color models	9
2.2.2	Substituting individual color models	12
2.2.3	Changing the target color model within a document	12
2.3	Arguments and terminology	13

* Cette extension peut être téléchargée à partir de `CTAN/macros/latex/contrib/xcolor/`. Un site Internet dédié à `xcolor` existe également : www.ukern.de/tex/xcolor.html. N’hésitez à envoyer vos constats d’erreur et suggestions d’amélioration à l’auteur : xcolor@ukern.de.

2.3.1	Additional remarks and restrictions on arguments	13
2.3.2	Meaning of standard color expressions	16
2.3.3	Meaning of extended color expressions	17
2.3.4	Color functions	18
2.4	Predefined colors	18
2.4.1	Colors that are always available	18
2.4.2	Additional sets of colors	18
2.5	Color definition	19
2.5.1	Ordinary and named colors	19
2.5.2	Color definition in xcolor	20
2.5.3	Defining sets of colors	21
2.5.4	Immediate and deferred definitions	22
2.5.5	Global color definitions	23
2.6	Color application	23
2.6.1	Standard color commands	23
2.6.2	Colored boxes	24
2.6.3	Using the current color	25
2.6.4	Color testing	25
2.7	Color blending	25
2.8	Color masks and separation	26
2.9	Color series	27
2.9.1	Definition of a color series	27
2.9.2	Initialisation of a color series	28
2.9.3	Application of a color series	28
2.9.4	Differences between colors and color series	29
2.10	Border colors for hyperlinks	29
2.11	Additional color specification in the pstricks world	30
2.12	Color in tables	30
2.13	Color information	31
2.14	Color conversion	31
2.15	Problems and solutions	31
2.15.1	Name clashes between dvipsnames and svgnames	31
2.15.2	Page breaks and pdfTeX	32
2.15.3	Change color of included .eps file	32
3	Examples	34
4	Colors by Name	41
4.1	Base colors (always available)	41
4.2	Colors via dvipsnames option	41
4.3	Colors via svgnames option	41
4.4	Colors via x11names option	42

5	Technical Supplement	44
5.1	Color models supported by drivers	44
5.2	How xcolor handles driver-specific color models	44
5.3	Behind the scenes : internal color representation	45
5.4	A remark on accuracy	45
6	The Formulas	47
6.1	Color mixing	47
6.2	Conversion between integer and real models	47
6.2.1	Real to integer conversion	49
6.2.2	Integer to real conversion	49
6.3	Color conversion and complements	50
6.3.1	The rgb model	51
6.3.2	The cmY model	53
6.3.3	The cmYk model	54
6.3.4	The hsb model	55
6.3.5	The Hsb model	57
6.3.6	The tHsb model	57
6.3.7	The gray model	57
6.3.8	The RGB model	58
6.3.9	The HTML model	58
6.3.10	The HSB model	58
6.3.11	The Gray model	58
6.3.12	The wave model	59
	References	61
	Appendix	62
	Acknowledgement	62
	Trademarks	62
	Known Issues	62
	History	62
	Index	67

Liste des tableaux

1	Package options	10
2	Package loading order	11
3	Supported color models	11
4	Arguments and terminology	14
5	Drivers and color models	44
6	Driver-dependent internal color representation	46
7	Color constants	48
8	Color conversion pairs	48

Table des figures










1	Color spectrum	34
2	Color testing	34
3	Progressing from one to another color	35
4	Target color model	36
5	Standard color expressions	36
6	Standard color expressions	36
7	Current color	36
8	Color series	37
9	Color masking	38
10	Alternating row colors in tables : <code>\rowcolors</code> vs. <code>\rowcolors*</code> . .	38
11	Hsb and tHsb : hue° in 15° steps	39
12	Color harmony	40

1 Introduction

1.1 Objectif de cette extension

L'extension `color` met à disposition un outil puissant et stable pour manipuler les couleurs dans (pdf) \LaTeX de façon cohérente, indépendamment des pilotes graphiques, tout en supportant différents modèles colorimétriques (de manière un peu moins indépendante des pilotes).

Néanmoins, il est parfois un peu laborieux de l'utiliser, particulièrement dans les cas où de légères variations de couleur, des mélanges de couleur ou des conversions de couleur sont en jeu : ceci implique d'habitude l'utilisation d'un autre programme qui calcule les paramètres souhaités, paramètres alors copiés dans une commande `\definecolor` dans \LaTeX . Assez fréquemment, une calculatrice de poche est également retenue dans le traitement de problèmes comme ceux indiqués ci-après :

- Ma société a défini une couleur d'entreprise et le service des impressions m'a dit combien il est coûteux d'utiliser plus de deux couleurs dans notre nouveau brochure, alors même que toutes les teintes de notre couleur (par exemple, une version à 75%) peuvent être utilisées sans aucun surcoût. Comment accéder à ces variations de couleur avec \LaTeX ?
(Réponse : `\color{CouleurEntreprise!75}` etc.)
- Mon ami utilise une belle couleur que je souhaiterais appliquer à mes propres documents ; malheureusement, elle est définie avec le modèle **hsb** qui n'est pas accepté par mon application pdf \LaTeX favorite. Que faire alors ?
(Réponse : utiliser tout simplement les définitions **hsb**, `xcolor` fera les calculs nécessaires.)
- Qu'affiche le mélange de 40% de *green* et de 60% de *yellow* ?
(Réponse : 40%  + 60%  = , soit `\color{green!40!yellow}`)
- Et quelle est l'aspect de sa couleur complémentaire ?
(Réponse : , accessible via `\color{-green!40!yellow}`)
- Maintenant, je souhaite mélanger trois mesures de la dernière couleur avec deux mesures de sa complémentaire et une mesure de *rouge*. Qu'est-ce que cela donne ?
(Réponse : $3 \times$  $+ 2 \times$  $+ 1 \times$  = , cette couleur étant accessible via `\color{rgb:-green!40!yellow,3;green!40!yellow,2;red,1}`)
- Je sais qu'un rayonnement de longueur d'onde de 485nm appartient au spectre visible. Mais quelle couleur a-t-il ?
(Réponse : approximativement , via `\color[wave]{485}`)
- Mon service des impressions souhaite que toutes les définitions de couleur dans mon document soit basculées en modèle **cmyk** (CMJN en français). Comment puis-je faire ces calculs efficacement ?
(Réponse : `\usepackage[cmyk]{xcolor}` ou `\selectcolormodel{cmyk}`)
- J'ai un tableau de 50 lignes. Comment puis-je obtenir des lignes de tableau avec deux couleurs alternées (A pour les lignes paires et B pour les lignes impaires) sans recours à la copie de 50 commandes `\rowcolor` ? Ce motif alterné devrait d'ailleurs commencer à partir de la troisième ligne.

(Réponse : *grosso modo* `\rowcolors{3}{CouleurA}{CouleurB}`)

Ceci représente quelques uns des problème résolu par l'extension xcolor. Son objectif peut être résumé en la conservation des caractéristiques de `color`, tout en apportant des fonctionnalités additionnelles et de la flexibilité avec des interfaces simples d'utilisation (avec un peu de chance).

1.2 Teintes, nuances, tons et couleurs complémentaires

Sur la base de [15] nous définissons les termes suivants :

- **teinte** : une couleur à laquelle est ajoutée du *blanc* (white) ;
- **nuance** : une couleur à laquelle est ajoutée du *noir* (black) ;
- **ton** : une couleur à laquelle est ajoutée du *gris* (gray).

Ce sont des cas spéciaux d'une fonction plus générale $\text{mélange}(C, C', p)$ qui construit une nouvelle couleur, composée de p mesures de la couleur C et de $1 - p$ mesures de la couleur C' , où $0 \leq p \leq 1$. Aussi, nous posons

$$\text{teinte}(C, p) := \text{mélange}(C, \text{white}, p) \quad (1)$$

$$\text{nuance}(C, p) := \text{mélange}(C, \text{black}, p) \quad (2)$$

$$\text{ton}(C, p) := \text{mélange}(C, \text{gray}, p) \quad (3)$$

où **white**, **black**, and **gray** sont des constantes dépendantes des modèles, comme présentées en table 7 on page 48. Par la suite, nous définissons le terme :

- **complémentaire** : une couleur C^* qui génère du *blanc* si elle est mélangée avec la couleur d'origine C ,

sachant qu'il existe également différents concepts de complémentarité (par exemple des couleurs opposées sur les *cercles chromatiques*). Voir la section 6.3 on page 50 pour le détail des calculs et la section 1.4 on the following page pour certaines remarques sur les cercles chromatiques.

1.3 Modèles colorimétriques

Un modèle colorimétrique est un outil décrivant ou représentant un certain ensemble de couleurs d'une manière compatible avec l'appareil cible souhaité, par exemple un écran ou une imprimante. Il existe des modèles propriétaires (comme Pantone ou HKS) qui fournissent des ensembles finis de couleurs (chaque couleur étant appelés *ton direct*), dans lequel l'utilisateur doit piocher sans se soucier des paramétrages ; à l'inverse, se trouvent des modèles paramétriques comme **gray**, **rgb**, et **cmymk**, dont le but est de représenter de larges ensembles finis ou même infinis (théoriquement) de couleurs, construits sur de très petits sous-ensembles de couleurs de base et de règles permettant de construire d'autres couleurs à partir des couleurs de base. Par exemple, un large ensemble de couleurs peut être construit par combinaison linéaire des couleurs de base *rouge*, *vert* et *bleu*. En contrepartie, un ton direct ne peut souvent être qu'*approximé* par des valeurs de paramètres dans les modèles comme **cmymk** ou **rgb** ; les couleurs originales se mélangent physiquement et dépendent aussi du type de papier retenu. Enfin, il existe certaines couleurs comme l'*or* ou l'*argent* qui sont difficilement reproductibles par des modèles paramétriques avec des encres standard ou des imprimantes laser.

1.4 Cercles chromatiques et accords de couleurs

Il s'est développé une longue histoire du placement de couleurs (*teintes*) sur des cercles pour discuter de problèmes théoriques ou pratiques sur les couleurs (par exemple Isaac Newton, Johann Wolfgang von Goethe). Une explication de ceci pourrait être que le cercle lui-même est un outil tout naturel pour illustrer des relations communes aussi bien que des propriétés opposées.

De nos jours, une certaine confusion portant sur les notions associées aux couleurs existe, dans la mesure où deux grands domaines qui y sont liés — l'art et le design graphique d'une part, la théorie scientifique de couleur de l'autre — tendent à utiliser les même mots pour décrire les propriétés de la couleur bien qu'en décrivant parfois des faits très différents ! Ainsi, l'apparence des cercles chromatiques diffère autant que la signification de concepts comme couleurs 'primaires' ou 'complémentaires'.

Construction d'un cercle chromatique typique Tout d'abord, trois *couleurs primaires* sont placées sur le cercle à 0° , 120° , 240° (les artistes choisissent souvent le triplet *rouge, jaune, bleu*, tandis que les scientifiques spécialistes de la couleur préféreront le triplet *rouge, vert, bleu*). Ensuite, les trois *couleurs secondaires* sont placées à 60° , 180° , 300° . Puis, six *couleurs tertiaires* pourront être placées au milieu de chaque arc (30° , 90° , ...). C'est pourquoi les cercles chromatiques sont fréquemment décrits par douze couleurs équidistantes bien que l'algorithme puisse être prolongé à merci.

Harmonies de couleur issues du cercle Nous commençons avec un cercle chromatique quelconque :

- les **couleurs complémentaires** sont situées à une distance de 180° sur le cercle ;
- les ** correspondent à trois couleurs séparées par 120° ;
- les ** correspondent à quatre couleurs séparées par 90° .

Nous supposons maintenant que le cercle est décomposé en $2n$ secteurs de taille égale :

- les **couleurs complémentaires adjacentes** d'une couleur donnée sont les deux voisines immédiates de la couleur complémentaire, caractérisées par les positions $\frac{n \pm 1}{2n} \cdot 360^\circ$,
- les **couleurs analogues** d'une couleur donnée sont les deux ou quatre voisines, caractérisées par les positions $\pm \frac{1}{2n} \cdot 360^\circ$ and $\pm \frac{2}{2n} \cdot 360^\circ$.

Vu les méthodes utilisées pour générer des accords de couleur, nous concluons que les résultats dépendent fortement de la manière dont nous construisons le cercle. Qui plus est, le choix de n affectera également le résultat visuel. Des exemples sont donnés en figure 12 on page 40.

A traduire

2 L'interface utilisateur

2.1 Préparation

2.1.1 Installation de l'extension

Il faut tout d'abord placer `xcolor.sty` et tous les fichiers dans un répertoire où (pdf)LaTeX les trouvera. Un emplacement classique selon le **TeX Directory Structure* (TDS)* serait le répertoire `texmf/tex/latex/xcolor`, où `texmf` est à remplacer par le répertoire principal de votre installation de TeX. De plus, il faut placer `xcolor.pro` à un endroit où `dvips` le trouvera, typiquement `texmf/dvips/xcolor`. Normalement, vous devez lancer une mise à jour de votre base de noms de fichiers afin que ces fichiers soient connus et facilement retrouvables dans l'arborescence TeX. Par la suite, il suffit simplement d'utiliser `xcolor` (au lieu de `color`) dans votre document. Pour cela, la commande générale d'appel est `\usepackage[options]{xcolor}` dans le préambule du document. La table 2 on page 11 montre dans quel ordre certaines extensions doivent être alors chargées.

2.1.2 Options de l'extension

En général, plusieurs types d'options existent :

- les options qui déterminent le pilote graphique comme expliqué dans [5] et [6], soit actuellement : `dvips`, `xdvi`, `dvipdf`, `dvipdfm`, `dvipdfmx`, `pdftex`, `dvipsone`, `dviwindo`, `emtex`, `dviwin`, `oztex`, `textures`, `pctexps`, `pctexwin`, `pctexhp`, `pctex32`, `truetex`, `tcidvi`, `vtex`, `xetex`;
- les options qui déterminent le modèle colorimétrique cible¹ (`natural`, `rgb`, `cmv`, `cmv`, `hsb`, `gray`, `RGB`, `HTML`, `HSB`, `Gray`) ou une sortie avec des couleurs désactivées (`monochrome`);
- les options qui contrôlent si certains ensembles de couleurs prédéfinies sont chargés et comment : `dvipsnames`, `dvipsnames*`, `svgnames`, `svgnames*`, `x11names`, `x11names*`;
- les options qui déterminent quelles autres extensions doivent être chargées ou supportées : `table`, `fixpdftex`, `hyperref`;
- les options qui influencent le comportement d'autres commandes : `prologue`, `kernelbbox`, `xcdraw`, `noxcdraw`, `fixinclude`, `showerrors`, `hideerrors`;
- les options obsolètes : `pst`, `override`, `usenames`, `nodvipsnames`.

All available package options (except driver selection and obsolete options) are listed in table 1 on page 10. In order to facilitate the co-operation with the `hyperref` package, there is a command `\GetGinDriver`² that grabs the driver actually used and puts it into the command `\GinDriver`. The latter can then be used within `hyperref` (or other packages), see the code example on page 12. If there is no corresponding `hyperref` option, `hypertex` will be taken as default.

Warning : there is a substantial difference between `xcolor` and `color` regarding how the `dvips` option is being handled. The extension `color` implicitly invokes the

1. La section 2.2.3 on page 12 explique comment ce paramétrage peut être annulé n'importe où dans un document.

2. This command is executed automatically if the package option `hyperref` is used.

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

\xcolorcmd Here is a simple interface to pass commands that should be executed at the end of the extension **xcolor** (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 L^AT_EX document, a command like **'\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 T_EX 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 11. 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 **\textcolor[cm]{0.7,0.5,0.3}{foo}** (*foo*) or **\textcolor[HTML]{AFFE90}{foo}** (*foo*).

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 53 for details.

cm 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, **cm** is treated as **cmyk** with $k = 0$.

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.

TABLE 1 – Package options

<i>Option</i>	<i>Description</i>
<code>natural</code>	(Default.) Keep all colors in their model, except RGB (converted to rgb), HSB (converted to hsb), and Gray (converted to gray).
<code>rgb</code>	Convert all colors to the rgb model.
<code>cmv</code>	Convert all colors to the cmv model.
<code>cmv</code>	Convert all colors to the cmv model.
<code>hsb</code>	Convert all colors to the hsb model.
<code>gray</code>	Convert all colors to the gray model. Especially useful to simulate how a black & white printer will output the document.
<code>RGB</code>	Convert all colors to the RGB model (and afterwards to rgb).
<code>HTML</code>	Convert all colors to the HTML model (and afterwards to rgb).
<code>HSB</code>	Convert all colors to the HSB model (and afterwards to hsb).
<code>Gray</code>	Convert all colors to the Gray model (and afterwards to gray).
<code>dvipsnames, dvipsnames*</code>	Load a set of predefined colors. ¹
<code>svgnames, svgnames*</code>	Load a set of predefined colors according to SVG 1.1. ¹
<code>x11names, x11names*</code>	Load a set of predefined colors according to Unix/X11. ¹
<code>table</code>	Load the <code>colortbl</code> package, in order to use the tools for coloring rows, columns, and cells within tables.
<code>fixpdftex</code>	Load the <code>pdfcolmk</code> package, in order to improve <code>pdftex</code> 's color behaviour (see section 2.15.2 on page 32).
<code>hyperref</code>	Support the <code>hyperref</code> package in terms of color expressions by defining additional keys (see section 2.10 on page 29).
<code>prologue</code>	Write prologue information to <code>.xcp</code> file for every color definition (as described in section 2.5.1 on page 19).
<code>kernelbbox</code>	Use L ^A T _E X kernel method to draw <code>\f(rame)box</code> frames ² .
<code>xcdraw</code>	Use driver-specific commands to draw frames and color boxes. ²
<code>noxcdraw</code>	(Default.) Use generic code to draw frames and color boxes. ²
<code>fixinclude</code>	Prevent <code>dvips</code> color reset before <code>.eps</code> file inclusion (see section 2.15.3 on page 32).
<code>showerrors</code>	(Default.) Display an error message if an undefined color is being used (same behaviour as in the original extension <code>color</code>).
<code>hideerrors</code>	Display only a warning if an undefined color is being used, and replace this color by <i>black</i> .

¹ See section 2.4.2 on page 18. ² See section 2.6.2 on page 24.

TABLE 2 – Package loading order

<i>Action/Package</i>	<i>colortbl</i>	<i>pdfcolmk</i>	<i>hyperref</i>	<i>pstricks</i>	<i>color</i>	<i>pstcol</i>
load before xcolor	no	no	allowed	allowed ¹	no	no
load with xcolor option	table	fixpdxftex	—	—	—	—
load after xcolor	no	allowed	allowed	allowed	no	no
¹ recent pstricks versions do load xcolor by default						

TABLE 3 – Supported color models

<i>Name</i>	<i>Base colors/notions</i>	<i>Parameter range</i>	<i>Default</i>
rgb	<i>red, green, blue</i>	$[0, 1]^3$	
cmY	<i>cyan, magenta, yellow</i>	$[0, 1]^3$	
cmYk	<i>cyan, magenta, yellow, black</i>	$[0, 1]^4$	
hsb	<i>hue, saturation, brightness</i>	$[0, 1]^3$	
Hsb	<i>hue[°], saturation, brightness</i>	$[0, H] \times [0, 1]^2$	$H = 360$
tHsb	<i>hue[°], saturation, brightness</i>	$[0, H] \times [0, 1]^2$	$H = 360$
gray	<i>gray</i>	$[0, 1]$	
RGB	<i>Red, Green, Blue</i>	$\{0, 1, \dots, L\}^3$	$L = 255$
HTML	<i>RRGGBB</i>	$\{000000, \dots, \text{FFFFFF}\}$	
HSB	<i>Hue, Saturation, Brightness</i>	$\{0, 1, \dots, M\}^3$	$M = 240$
Gray	<i>Gray</i>	$\{0, 1, \dots, N\}$	$N = 15$
wave	<i>lambda (nm)</i>	$[363, 814]$	
L, M, N are positive integers; H is a positive real number			

Hsb, tHsb Firstly, **Hsb** is a ‘user interface’ model, mapping $hue \in [0, 1]$ onto $hue^\circ \in [0, H]$, where H is given by `\def\rangeHsb{\langle H \rangle}`. 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 *tuned Hsb*, which enables the user to apply a piecewise linear transformation on hue° by shifting selected hue° values forward or backward on the circle. This transformation is defined by `\def\rangetHsb{x_1, y_1; x_2, y_2; \dots}` which specifies that $hue^\circ = x_1$ in **tHsb** means $hue^\circ = y_1$ in **Hsb**, etc. For example, *yellow* is at 60° in the **Hsb** circle (*red* being at 0°), however, in most color wheels used by artists, *yellow* is at 120° . Therefore, a ‘120, 60’ entry would make sense if we planned to replicate an artist’s color wheel via **tHsb**. See section 6.3.6 on page 57 for the exact transformation formula and further restrictions, and section 1.4 on page 7 for color wheels and color harmonies. Figure 11 on page 39 may serve for comparison purposes.

Example : ‘`\def\rangetHsb{60, 30; 120, 60; 180, 120; 210, 180; 240, 240}`’ is in

fact xcolor’s default setting.

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 xcolor generally allows for all real ‘wavelengths’ λ that have an absolute value less than 16383.99998 (the biggest number T_EX 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 34 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 extension xcolor is loaded, e.g.,

```
\rangeRGB
\rangeHSB
\rangeGray

\documentclass{article}
...
\def\rangeRGB{15}
\usepackage[dvips]{xcolor}
...
\GetGinDriver
\usepackage[\GinDriver]{hyperref}
...
\begin{document}
...
\def\rangeRGB{63}
...
```

2.2.2 Substituting individual color models

`\substitutecolormodel` $\{ \langle source\ model \rangle \} \{ \langle target\ model\ list \rangle \}$
 Substitute $\langle source\ model \rangle$ by the first actually present model that occurs in $\langle target\ model\ list \rangle$. Only color models of type $\langle num\ model \rangle$ 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` $\{ \langle num\ model \rangle \}$

Sets the target model to $\langle num\ model \rangle$, where the latter is one of the model names allowed as package option (i.e., `natural`, `rgb`, `cm`, `cm`, `cm`, `cm`, `hsb`, `gray`, `RGB`, `HTML`, `HSB`, `Gray`), see figure 4 on page 36 for an example. There are two possible hooks, where the conversion to the target model can take place :

- `\ifconvertcolorsD` — at color *definition* time³ (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 specified directly like `\color[rgb]{.1,.2,.3}`); this is controlled by the switch `\ifconvertcolorsU`.

Both switches are set to ‘true’ by selecting any of the models, except `natural`, which sets them to ‘false’. This applies for selection via a package option as well as via `\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 `\usepackage[rgb,...]{xcolor}\convertcolorsUfalse`, which is actually the way how xcolor worked up to version 1.07.

2.3 Arguments and terminology

Before we describe 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 following page.

2.3.1 Additional remarks and restrictions on arguments

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

$\langle name \rangle$ **Color names** A $\langle name \rangle$ denotes the declared name (or the name to be declared) of a *color* or a *color series*; it may be declared *explicitly* by one of the following commands: `\definecolor`, `\providecolor`, `\colorlet`, `\definecolorset`, `\providecolorset`, `\definecolorseries`, `\definecolors`, `\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 $\langle name \rangle$ declarations, but the given restriction avoids misunderstandings and ensures compatibility with future extensions of xcolor.

3. This means that all *newly* defined colors will be first converted to the target model, then saved.

TABLE 4 – Arguments and terminology

<i>Element</i>	<i>Replacement string</i>	
$\langle \text{empty} \rangle$	→ empty string ‘ ’	
$\langle \text{minus} \rangle$	→ non-empty string consisting of one or more minus signs ‘-’	
$\langle \text{plus} \rangle$	→ non-empty string consisting of one or more plus signs ‘+’	
$\langle \text{int} \rangle$	→ integer number	(integer)
$\langle \text{num} \rangle$	→ non-negative integer number	(number)
$\langle \text{dec} \rangle$	→ real number	(decimal)
$\langle \text{div} \rangle$	→ non-zero real number	(divisor)
$\langle \text{pct} \rangle$	→ real number from the interval [0, 100]	(percentage)
$\langle \text{id} \rangle$	→ non-empty string consisting of letters and digits	(identifier)
$\langle \text{ext id} \rangle$	→ $\langle \text{id} \rangle$ → $\langle \text{id} \rangle_1 = \langle \text{id} \rangle_2$	
$\langle \text{id-list} \rangle$	→ $\langle \text{ext id} \rangle_1, \langle \text{ext id} \rangle_2, \dots, \langle \text{ext id} \rangle_l$	
$\langle \text{name} \rangle$	→ $\langle \text{id} \rangle$ → ‘.’	(explicit name) (implicit name)
$\langle \text{core model} \rangle$	→ ‘rgb’, ‘cmy’, ‘cmyk’, ‘hsb’, ‘gray’	(core models)
$\langle \text{num model} \rangle$	→ $\langle \text{core model} \rangle$ → ‘RGB’, ‘HTML’, ‘HSB’, ‘Gray’ → ‘Hsb’, ‘tHsb’, ‘wave’	(integer models) (decimal models)
$\langle \text{model} \rangle$	→ $\langle \text{num model} \rangle$ → ‘named’	(numerical models) (pseudo model)
$\langle \text{model-list} \rangle$	→ $\langle \text{model} \rangle_1 / \langle \text{model} \rangle_2 / \dots / \langle \text{model} \rangle_m$ → $\langle \text{core model} \rangle : \langle \text{model} \rangle_1 / \langle \text{model} \rangle_2 / \dots / \langle \text{model} \rangle_m$	(multiple models)
$\langle \text{spec} \rangle$	→ comma-separated list of numerical values → space-separated list of numerical values → name of a ‘named’ color	(explicit specification) (explicit specification) (implicit specification)
$\langle \text{spec-list} \rangle$	→ $\langle \text{spec} \rangle_1 / \langle \text{spec} \rangle_2 / \dots / \langle \text{spec} \rangle_m$	(multiple specifications)
$\langle \text{type} \rangle$	→ $\langle \text{empty} \rangle$ → ‘named’, ‘ps’	
$\langle \text{expr} \rangle$	→ $\langle \text{prefix} \rangle \langle \text{name} \rangle \langle \text{mix expr} \rangle \langle \text{postfix} \rangle$	(standard color expression)
$\langle \text{prefix} \rangle$	→ $\langle \text{empty} \rangle$ → $\langle \text{minus} \rangle$	(complement indicator)
$\langle \text{mix expr} \rangle$	→ $! \langle \text{pct} \rangle_1 ! \langle \text{name} \rangle_1 ! \langle \text{pct} \rangle_2 ! \langle \text{name} \rangle_2 ! \dots ! \langle \text{pct} \rangle_n ! \langle \text{name} \rangle_n$ → $! \langle \text{pct} \rangle_1 ! \langle \text{name} \rangle_1 ! \langle \text{pct} \rangle_2 ! \langle \text{name} \rangle_2 ! \dots ! \langle \text{pct} \rangle_n$	(complete mix expr.) (incomplete mix expr.)
$\langle \text{postfix} \rangle$	→ $\langle \text{empty} \rangle$ → $!! \langle \text{plus} \rangle$ → $!! [\langle \text{num} \rangle]$	(series step) (series access)
$\langle \text{ext expr} \rangle$	→ $\langle \text{core model} \rangle, \langle \text{div} \rangle : \langle \text{expr} \rangle_1, \langle \text{dec} \rangle_1 ; \langle \text{expr} \rangle_2, \langle \text{dec} \rangle_2 ; \dots ; \langle \text{expr} \rangle_k, \langle \text{dec} \rangle_k$ → $\langle \text{core model} \rangle : \langle \text{expr} \rangle_1, \langle \text{dec} \rangle_1 ; \langle \text{expr} \rangle_2, \langle \text{dec} \rangle_2 ; \dots ; \langle \text{expr} \rangle_k, \langle \text{dec} \rangle_k$	
$\langle \text{func expr} \rangle$	→ $> \langle \text{function} \rangle, \langle \text{arg} \rangle_1, \langle \text{arg} \rangle_2, \dots, \langle \text{arg} \rangle_j$	(color function expression)
$\langle \text{function} \rangle$	→ ‘wheel’, ‘twheel’	(color functions)
$\langle \text{color} \rangle$	→ $\langle \text{color expr} \rangle \langle \text{func expr} \rangle_1 \langle \text{func expr} \rangle_2 \dots \langle \text{func expr} \rangle_i$	
$\langle \text{color expr} \rangle$	→ $\langle \text{name} \rangle$ → $\langle \text{expr} \rangle$ → $\langle \text{ext expr} \rangle$	

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, $k, l, m, n > 0$, $m \leq 8$.

Examples : ‘red’, ‘MySpecialGreen1980’, ‘.’.

$\langle \text{core model} \rangle$ **Color models** The differentiation between *core models* (**rgb**, **cm**y, **cm**yk, **h**sb, **gray**), *integer models* (**RGB**, **HTML**, **HSB**, **Gray**), *decimal models* (**Hsb**, **tHsb**, **wave**) and *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 **Hsb** and **tHsb** are special-purpose versions of **h**sb, whereas **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 T_EX.

$\langle \text{spec} \rangle$ **Color specifications** The $\langle \text{spec} \rangle$ 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 11, the latter — ideally — referring to driver-provided names.

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

$\langle \text{model-list} \rangle$ **Multiple models and specifications** These arguments always appear in $\langle \text{spec-list} \rangle$ (explicit or implicit) pairs within the following color definition commands : `\definecolor`, `\providecolor`, `\definecolorset`, `\providecolorset`. First, $\langle \text{model-spec} \rangle$ 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 $\langle \text{spec-list} \rangle$, such that we arrive at a proper $(\langle \text{model} \rangle, \langle \text{spec} \rangle)$ pair. Therefore, in the actual executed color definition there is no ambiguity anymore. The extended form $\langle \text{core model} \rangle : \langle \text{model} \rangle_1 / \langle \text{model} \rangle_2 / \dots / \langle \text{model} \rangle_m$ causes an immediate conversion of the relevant $\langle \text{spec} \rangle$ to $\langle \text{core model} \rangle$; an unknown target model will be silently ignored here.

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

y/hsb’.

$\langle \text{type} \rangle$ **The type argument** This is used only in the context of color defining commands, see the description of `\definecolor` and friends.

$\langle \text{expr} \rangle$ **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. $\langle \text{prefix} \rangle$ The $\langle \text{prefix} \rangle$ 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 $\langle \text{name} \rangle_n = \text{white}$, in order to save some keystrokes in the

case of tints. The $\langle postfix \rangle$ string is usually empty, but it offers some additional functionality in the case of a *color series* : the non-empty cases require that

- $\langle name \rangle$ denotes the name of a *color series*,
- $\langle mix\ expr \rangle$ is a *complete mix expression*.

Examples : ‘red’, ‘-red’, ‘--red!50!green!12.345’, ‘red!50!green!20!blue’, ‘foo!!+’, ‘foo!![7]’, ‘foo!25!red!!++’, ‘foo!25!red!70!green!![7]’.

$\langle ext\ expr \rangle$ **Extended color expressions** These expressions provide another method of color mixing, see section 2.3.3 on the following page for details. The shorter form

$$\langle core\ model \rangle : \langle expr \rangle_1, \langle dec \rangle_1 ; \langle expr \rangle_2, \langle dec \rangle_2 ; \dots ; \langle expr \rangle_k ! \langle dec \rangle_k$$

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

$$\langle core\ model \rangle, \langle div \rangle : \langle expr \rangle_1, \langle dec \rangle_1 ; \langle expr \rangle_2, \langle dec \rangle_2 ; \dots ; \langle expr \rangle_k ! \langle dec \rangle_k$$

with the following definition (requiring a non-zero sum of all $\langle dec \rangle_k$ coefficients) :

$$\langle div \rangle := \langle dec \rangle_1 + \langle dec \rangle_2 + \dots + \langle dec \rangle_k \neq 0.$$

Examples : ‘rgb:red,1’, ‘cmyk:red,1;-green!25!blue!60,11.25;blue,-2’.

$\langle func\ expression \rangle$
 $\langle function \rangle$ **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 page 18 for details.

Examples : ‘>wheel,30’, ‘>wheel,30,’ , ‘>twheel,1,12’, ‘>twheel,-11,12’.

$\langle color \rangle$
 $\langle color\ expr \rangle$ **Colors** Finally, $\langle color \rangle$ is the ‘umbrella’ argument, covering the different concepts of specifying colors. This means, whenever there is a $\langle color \rangle$ 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

$$\langle prefix \rangle \langle name \rangle ! \langle pct \rangle_1 ! \langle name \rangle_1 ! \langle pct \rangle_2 ! \dots ! \langle pct \rangle_n ! \langle name \rangle_n \langle postfix \rangle$$

is being interpreted and processed :

1. First of all, the model and color parameters of $\langle name \rangle$ are extracted to define a temporary color $\langle temp \rangle$. If $\langle postfix \rangle$ has the form ‘!![$\langle num \rangle$]’, then $\langle temp \rangle$ will be the corresponding (direct-accessed) color $\langle num \rangle$ from the series $\langle name \rangle$.
2. Then a color mix, consisting of $\langle pct \rangle_1\%$ of color $\langle temp \rangle$ and $(100 - \langle pct \rangle_1)\%$ of color $\langle name \rangle_1$ is computed ; this is the new temporary color $\langle temp \rangle$.

3. The previous step is being repeated for all remaining parameter pairs $(\langle pct \rangle_2, \langle name \rangle_2), \dots, (\langle pct \rangle_n, \langle name \rangle_n)$.
4. If $\langle prefix \rangle$ consists of an odd number of minus signs '-', then $\langle temp \rangle$ will be changed into its complementary color.
5. If $\langle postfix \rangle$ has the form '!!+', '!!++', '!!+++', etc., a number of step commands (= number of '+' signs) are performed on the underlying color series $\langle name \rangle$. This has no consequences for the color $\langle temp \rangle$.
6. Now the color $\langle temp \rangle$ is being displayed or serves as an input for other operations, depending on the invoking command.

Note that in a typical step 2 expression $\langle temp \rangle ! \langle pct \rangle_\nu ! \langle name \rangle_\nu$, if $\langle pct \rangle_\nu = 100$ resp. $\langle pct \rangle_\nu = 0$, the color $\langle temp \rangle$ resp. $\langle name \rangle_\nu$ is used without further transformations. In the true mix case, $0 < \langle pct \rangle_\nu < 100$, the two involved colors may have been defined in different color models, e.g., `\definecolor{foo}{rgb}{...}` and `\definecolor{bar}{cmyk}{...}`. In general, the second color, $\langle name \rangle_\nu$, is transformed into the model of the first color, $\langle temp \rangle$, then the mix is calculated within that model.⁴ Thus, $\langle temp \rangle ! \langle pct \rangle_\nu ! \langle name \rangle_\nu$ and $\langle name \rangle_\nu ! \langle 100 - pct \rangle_\nu ! \langle temp \rangle$, which should be equivalent theoretically, will not necessarily yield identical visual results.

Figures 5 to 6 on page 36 show some first applications of colors and expressions. More examples are given in figure 3 on page 35. 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*

$$\langle core\ model \rangle : \langle expr \rangle_1, \langle dec \rangle_1 ; \langle expr \rangle_2, \langle dec \rangle_2 ; \dots ; \langle expr \rangle_k, \langle dec \rangle_k$$

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


$$\langle div \rangle := \langle dec \rangle_1 + \langle dec \rangle_2 + \dots + \langle dec \rangle_k.$$

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  via `\color{rgb:red,4;green,2;yellow,1}`. Trying the same with -1 parts of yellow instead, we get . Note that this mechanism can also be used to display an individual color (expression) in a certain color model : `\color{rgb:yellow,1}` results in such a conversion. The general form

$$\langle core\ model \rangle, \langle div \rangle : \langle expr \rangle_1, \langle dec \rangle_1 ; \langle expr \rangle_2, \langle dec \rangle_2 ; \dots ; \langle expr \rangle_k, \langle dec \rangle_k$$

4. Exception : in order to avoid strange results, this rule is being reversed if $\langle temp \rangle$ origins from the **gray** model ; in this case it is converted into the underlying model of $\langle name \rangle_\nu$.

does the same operation with the only difference that the divisor $\langle div \rangle$ is being specified instead of calculated. In the above example, we get a shaded version  via `\color{rgb,9:red,4;green,2;yellow,1}`. Note that it is not forbidden to specify a $\langle div \rangle$ argument which is smaller than the sum of all $\langle dec \rangle_\kappa$, 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 : $\langle angle \rangle$ or $\langle angle \rangle, \langle full\ circle \rangle$, the former being an abbreviation of $\langle angle \rangle, \backslash rangeHsb$. These functions allow to calculate related colors by harmonic relations based on color wheels (cf. section 1.4 on page 7). The second argument $\langle full\ circle \rangle$ declares how many units a full circle consists of, the first argument states by how many units the given color has to be rotated. To this end, the given color is first converted to **Hsb** (in case of *wheel*), yielding hue° , *saturation*, and *brightness*, respectively. Then

$$hue^\circ := hue^\circ + \frac{\langle angle \rangle}{\langle full\ circle \rangle} \cdot H, \quad hue := u\left(\frac{hue^\circ}{H}\right) \quad (4)$$

where u is the range-reduction function of equation (7) and $H = \backslash rangeHsb$. With *saturation* and *brightness* left untouched, the final model is **hsb**. The *twheel* function works similarly, but its arguments refer to **tHsb** instead of **Hsb**. Examples are shown in figure 12 on page 40.

2.4 Predefined colors

2.4.1 Colors that are always available

Within `xcolor.sty`, the following color names are defined :  *red*,  *green*,  *blue*,  *cyan*,  *magenta*,  *yellow*,  *black*,  *gray*,  *white*,  *darkgray*,  *lightgray*,  *brown*,  *lime*,  *olive*,  *orange*,  *pink*,  *purple*,  *teal*,  *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 13.

2.4.2 Additional sets of colors

There are also sets of color names that may be loaded by `xcolor` via package options, available in two variants : a ‘normal’ version (e.g., `dvipsnames`) and a ‘starred’ version (e.g., `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 `\definecolors` or

`\providecolors` commands, as described in section 2.5.4 on page 22, before they can be applied in a document.

- `dvipsnames/dvipsnames*` loads a set of 68 **cm**yk colors as defined in the `dvips` driver. However, these colors may be used in all supported drivers.
- `svgnames/svgnames*` loads a set of 151 **rgb** color names⁵ according to the SVG 1.1 specification [17]⁶, enhanced by 4 names taken from the file `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 41.
- `x11names/x11names*` loads a set of 317 **rgb** color names⁷ that are basically variations of a subset of the SVG set mentioned before, according to the file `rgb.txt` which is part of Unix/X11 distributions⁸. We describe now how to access all 752 color names of `rgb.txt` without much effort :
 - Load `x11namees` as well as `svgnames`.
 - 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 $N = 0, 1, \dots, 100$ use ‘`[gray]{N/100}`’ or ‘`black!100 - N`’ instead of *grayN* or *greyN*.

The color names and corresponding displays are listed in section 4 on page 41. Section 2.15.1 on page 31 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 extension `color` there is a distinction between ‘colors’ (defined by the command `\definecolor`) and ‘named colors’ (defined by `\DefineNamedColor`, which is allowed only in the preamble). Whenever an ordinary color is being used in a document, it will be translated into a `\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*

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 **RGB** parameters, converted to **rgb** by the author.

7. These names represent 315 different colors.

8. Again, the original definitions are given in **RGB** parameters, converted to **rgb** by the author.

behind that, i.e. employing names instead of parameters, ranges from ‘none’ to ‘complete’. We demonstrate the current situation for three different drivers :

- **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 command⁹ (**\definecolor**, **\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 *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 : **\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

9. This is not only true for the document preamble, but for the document body as well.

This is one of the commands that may be used to assign a $\langle name \rangle$ 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 13. It replaces both `color's` `\DefineNamedColor` and `\definecolor`. Note that an already existing color $\langle name \rangle$ will be overwritten. The variable `\tracingcolors` controls whether such an overwriting will be logged or not (see section 2.13 on page 31 for details). The arguments are described in section 2.3 on page 13. 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 30.

`\providecolor` [$\langle type \rangle$]{ $\langle name \rangle$ }{ $\langle model-list \rangle$ }{ $\langle spec-list \rangle$ }
Similar to `\definecolor`, but the color $\langle name \rangle$ is only defined if it does not exist already.

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

Example : we said `\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 `\rowcolor{tableheadcolor}`.

2.5.3 Defining sets of colors

`\definecolorset` [$\langle type \rangle$]{ $\langle model-list \rangle$ }{ $\langle head \rangle$ }{ $\langle tail \rangle$ }{ $\langle set spec \rangle$ }
This command facilitates the construction of *color sets*, i.e. (possibly large) sets of individual colors with common underlying $\langle model-list \rangle$ and $\langle type \rangle$. Here, $\langle set spec \rangle = \langle name \rangle_1, \langle spec-list \rangle_1 ; \dots ; \langle name \rangle_l, \langle spec-list \rangle_l$ ($l \geq 1$ name/specification-list pairs). Individual colors are being constructed by single

`\definecolor` [$\langle type \rangle$]{ $\langle head \rangle \langle name \rangle_\lambda \langle tail \rangle$ }{ $\langle model-list \rangle$ }{ $\langle spec-list \rangle_\lambda$ }

commands, $\lambda = 1, \dots, l$. For example,

10. Prior to version 2.00, this command was called `\xdefinecolor`, the latter name still being available for compatibility reasons.

— `\definecolorset{rgb}{head}{tail}{set spec}`
could be used to define the basic colors *red*, *green*, and *blue*; ¹¹
— `\definecolorset{rgb}{x}{10}{red,1,0,0;green,0,1,0;blue,0,0,1}`
would define the colors *xred10*, *xgreen10*, and *xblue10*.

`\providecolorset` [*type*]{*model-list*}{*head*}{*tail*}{*set spec*}

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

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 T_EX 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 *deferred color definition* comes into play. Its principle is simple : for every definition of this type (e.g., via `\preparecolor`), all necessary 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*}

`\ifdefinecolors` Similar to `\definecolorset`, but depending on the `\ifdefinecolors` 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 `\definecolorsfalse\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 compati-

¹¹. Actually, xcolor uses a more complicated variant to provide the basic colors for different underlying models (see the source code for the full command) :

```
\definecolorset{rgb/hsb/cmyk/gray}{head}{tail}{red,1,0,0/0,1,1/0,1,1,0/.3;green,...}.
```

lity reasons, especially to support the predefined colors in `dvipsnam.def`. It is the same as $\langle cmd \rangle [\langle type \rangle] \{ \langle name \rangle \} \{ \langle model \rangle \} \{ \langle spec \rangle \}$, where $\langle cmd \rangle$ is either `\definecolor` or `\preparecolor`, depending on the state of `\ifdefinecolors`. Note that `color`'s restriction to allow `\DefineNamedColor` only in the document preamble has been abolished in `xcolor`.

\definecolors $\{ \langle id-list \rangle \}$
Recall that $\langle id-list \rangle$ has the form $\langle ext\ id \rangle_1, \dots, \langle ext\ id \rangle_l$ where each $\langle ext\ id \rangle_\lambda$ is either an identifier $\langle id \rangle_\lambda$ or an assignment $\langle id \rangle_{\lambda'} = \langle id \rangle_\lambda$. We consider the first case to be an abbreviation for $\langle id \rangle_\lambda = \langle id \rangle_\lambda$ and describe the general case: the definition stack is searched for the name $\langle id \rangle_\lambda$ and its corresponding color parameters; if there is no match, nothing happens; if the name $\langle id \rangle_\lambda$ is on the stack and its color parameters are $\langle type \rangle_\lambda$, $\langle model \rangle_\lambda$, and $\langle spec \rangle_\lambda$, then the command `\definecolor` $[\langle type \rangle_\lambda] \{ \langle id \rangle_{\lambda'} \} \{ \langle model \rangle_\lambda \} \{ \langle spec \rangle_\lambda \}$ is executed. Thus, the user may control by which names the *prepared* colors are to be used in the document. Note that the entry $\langle id \rangle_\lambda$ is not removed from the stack, such that it can be used several times (even within the same `\definecolors` command).

\providecolors $\{ \langle id-list \rangle \}$
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

\ifglobalcolors 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.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}...`

2.6 Color application

2.6.1 Standard color commands

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

\color $\{ \langle color \rangle \}$
 $[\langle model-list \rangle] \{ \langle spec-list \rangle \}$
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 $\{ \langle color \rangle \} \{ \langle text \rangle \}$
 $[\langle model-list \rangle] \{ \langle spec-list \rangle \} \{ \langle text \rangle \}$
are just alternative syntax for `\color`, in which the groups are added implicitly. Thus $\langle text \rangle$ 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 $\{ \langle color \rangle \}$

12. The switch may also be set in the preamble in order to control the whole document.

`[$\langle model-list \rangle$]{ $\langle spec-list \rangle$ }`

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 $\langle color \rangle$ resp. $\langle spec \rangle$ 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.

2.6.2 Colored boxes

`\colorbox` `{ $\langle color \rangle$ }{ $\langle text \rangle$ }`

`[$\langle model-list \rangle$]{ $\langle spec-list \rangle$ }{ $\langle text \rangle$ }`

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

`\fcolorbox` `{ $\langle frame color \rangle$ }{ $\langle background color \rangle$ }{ $\langle text \rangle$ }`

`[$\langle model-list \rangle$]{ $\langle frame spec-list \rangle$ }{ $\langle background spec-list \rangle$ }{ $\langle text \rangle$ }`

`[$\langle fr. model-list \rangle$]{ $\langle fr. spec-list \rangle$ }[$\langle backgr. model-list \rangle$]{ $\langle backgr. spec-list \rangle$ }{ $\langle text \rangle$ }`

`{ $\langle frame color \rangle$ }[$\langle background model-list \rangle$]{ $\langle background spec-list \rangle$ }{ $\langle text \rangle$ }`

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 :

- `\test` `\fcolorbox{gray}{yellow}{test}`,
- `\test` `\fcolorbox[cmk]{0,0,0,0.5}{0,0,1,0}{test}`,
- `\test` `\fcolorbox[gray]{0.5}[wave]{580}{test}`,
- `\test` `\fcolorbox[gray][wave]{580}{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 :

`\boxframe` `{ $\langle width \rangle$ }{ $\langle height \rangle$ }{ $\langle depth \rangle$ }`

Draws a frame with a linewidth of `\fboxrule`. Returns a `\hbox` with outer dimensions $\langle width \rangle$, $\langle height \rangle$, $\langle depth \rangle$. 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 `kernel\fbbox` option is specified, which returns

to L^AT_EX's original definitions of `\f(rame)box`.

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 Using the current color

Within a color expression, `'.` serves as a placeholder for the current color. See figure 7 on page 36 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.

2.6.4 Color testing

`testcolors` [*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,cmym,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 34 and figures 11, 12.

2.7 Color blending

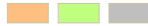
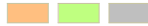
The purpose of *color blending* is to add some mixing color (expression) to all subsequent explicit color commands. Thus, it is possible to perform such a mix (or blend) operation for many colors without touching the individual commands.

`\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 *mix expr* argument will switch blending off.

Example : after `\blendcolors{!50!yellow}`, the colors  are transformed into , an additional `\blendcolors*{!50}` yields .

`\xglobal` 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 , an additional `\blendcolors*{!50}\color{black}` yields .

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 **cm**yk 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 `cm`yk, 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 $\langle m\text{-}model \rangle$ and a parameter vector $\langle m\text{-}spec \rangle$. Whenever a color is to be displayed in the document, it will first be converted to $\langle m\text{-}model \rangle$, afterwards each component of the resulting color vector will be multiplied by the corresponding component of $\langle m\text{-}spec \rangle$. For example, let’s assume that $\langle m\text{-}model \rangle$ equals `cm`yk, and $\langle m\text{-}spec \rangle$ equals $(\mu_c, \mu_m, \mu_y, \mu_k)$. Then an arbitrary color *foo* will be transformed according to

$$foo \mapsto (c, m, y, k) \mapsto (\mu_c \cdot c, \mu_m \cdot m, \mu_y \cdot y, \mu_k \cdot k) \quad (5)$$

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 **cm**yk model, see the last two rows in figure 9 on page 38.

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

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

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

```
\def\temp{cyan}\ifx\colormask\temp \includegraphics{foo_c}\else
\def\temp{magenta}\ifx\colormask\temp \includegraphics{foo_m}\else
...
\fi\fi
```

2.9 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 *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).

2.9.1 Definition of a color series

`\definecolorseries` $\{\langle name \rangle\}\{\langle core\ model \rangle\}\{\langle method \rangle\}[\langle b-model \rangle][\langle b-spec \rangle][\langle s-model \rangle][\langle s-spec \rangle]$
 Defines a color series called $\langle name \rangle$, whose calculations are performed within the color model $\langle core\ model \rangle$, where $\langle method \rangle$ selects the algorithm (one of **step**, **grad**, **last**, see below). The method details are determined by the remaining arguments :
 — $[\langle b-model \rangle][\langle b-spec \rangle]$ specifies the *base* (= first) color in the algorithm, either directly, e.g., `[rgb]{1,0.5,0.5}`, or as a $\langle color \rangle$, e.g., `{-yellow!50}`, if the optional argument is missing.
 — $[\langle s-model \rangle][\langle s-spec \rangle]$ specifies how the *step* vector is calculated in the algorithm, according to the chosen $\langle method \rangle$:
 — **step**, **grad** : the optional argument is meaningless, and $\langle s-spec \rangle$ is a parameter vector whose dimension is determined by $\langle core\ model \rangle$, e.g., `{0.1,-0.2,0.3}` in case of **rgb**, **cmY**, or **hsb**.
 — **last** : the last color is specified either directly, e.g., `[rgb]{1,0.5,0.5}`, or as a $\langle color \rangle$, e.g., `{-yellow!50}`, if the optional argument is missing.

This is the general scheme :

$$color_1 := base, \quad color_{n+1} := U(color_n + step) \quad (6)$$

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

$$U(x_1, \dots, x_m) = (u(x_1), \dots, u(x_m)), \quad u(x) = \begin{cases} 1 & \text{if } x = 1 \\ x - [x] & \text{if } x \neq 1 \end{cases} \quad (7)$$

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 :

- **step**, **grad** : the last argument, $\{\langle s\text{-spec} \rangle\}$, defines the directional vector *grad*.
- **last** : $\{\langle s\text{-spec} \rangle\}$ resp. $[\langle s\text{-model} \rangle][\langle s\text{-spec} \rangle]$ defines the color parameter vector *last*.

Then, during `\resetcolorseries`, the actual *step* vector is calculated :

$$step := \begin{cases} grad & \text{if } \langle method \rangle = \text{step} \\ \frac{1}{\langle div \rangle} \cdot grad & \text{if } \langle method \rangle = \text{grad} \\ \frac{1}{\langle div \rangle} \cdot (last - base) & \text{if } \langle method \rangle = \text{last} \end{cases} \quad (8)$$

Please note that it is also possible to use the current color placeholder ‘.’ within the definition of color series. Thus, `\definecolorseries{foo}{rgb}{last}{.}{-..}` will set up a series that starts with the current color and ends with its complement. Of course, similar to T_EX’s `\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

`\resetcolorseries` $[\langle div \rangle][\langle name \rangle]$

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

`\colorseriescycle`

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 13 used within a `\color`, `\textcolor`, ... command will display this color according to the usual syntax of such expressions. However, in the cases when $\langle postfix \rangle$ 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 $\langle num \rangle = 0, 1, 2, \dots$, starting with 0 for the base color. See figure 8 on page 37 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

```
\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

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*, *urlcolor* with color expressions, e.g., `\hypersetup{urlcolor=-green!50}`;
- display a colored border around hyperlinks, using the keys *citebordercolor*, *filebordercolor*, *linkbordercolor*, *menubordercolor*, *pagebordercolor*, *runbordercolor*, *urlbordercolor* 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, `xcolor` provides — via the package option `hyperref` — a set of extended keys *xcitebordercolor*, *xfilebordercolor*, *xlinkbordercolor*, *xmenubordercolor*, *xpagebordercolor*, *xrunbordercolor*, *xurlbordercolor* which are being used in conjunction with color expressions, e.g., `\hypersetup{xurlbordercolor=-green!50}`.

Another new key, *xpdfborder*, provides a way to deal with a *dvips*-related problem : for most of the drivers, a setting like `pdfborder={0 0 1}` will determine the width of the border that is drawn around hyperlinks in points. However, in the *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 *pdfborder* may be useful or not. Within `hyperref`, the default for *dvips* is

`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 `-Ppdf`. On the other hand, setting `pdfborder={0 0 80}` works fine for *dvips* at 8000 dpi, but makes a document unportable, since other drivers (or even *dvips* in a low resolution) will draw very thick boxes in that case. This is where the *xpdfborder* key comes in handy : it rescales its arguments for the *dvips* case by a factor 80 (ready for 8000 dpi) and leaves everything unchanged for other drivers. Thus one can say `xpdfborder={0 0 1}` in a driver-independent way.

2.11 Additional color specification in the pstricks world

For *pstricks* users, there are different ways of invoking colors within command option keys :

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

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

`\definecolor` `[ps]{<name>}{<core model-list>}{<code>}`

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

2.12 Color in tables

`\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{<stuff>}`.

`\showrowcolors` The *rowcolors status* is activated (i.e., use coloring scheme) by default and/or
`\hiderowcolors` `\showrowcolors`, it is inactivated (i.e., ignore coloring scheme) by the command
`\rownum` `\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 38. These commands require the `table` option (which loads the `colortbl` package).

Note that table coloring may be combined with color series. This method was used to construct the examples in figure 8 on page 37.

2.13 Color information

<code>\extractcolorspec</code>	<code>{\color}{\cmd}</code> Extracts the color specification of $\langle color \rangle$ and puts it into $\langle cmd \rangle$; equivalent to <code>\def\cmd{{\model}}{\spec}}</code> .
<code>\extractcolorspecs</code>	<code>{\color}{\model-cmd}{\color-cmd}</code> Extracts the color specification of $\langle color \rangle$ and puts it into $\langle model-cmd \rangle$ and $\langle color-cmd \rangle$, respectively.
<code>\tracingcolors</code>	<code>=\int</code> Controls the amount of information that is written into the log file : <ul style="list-style-type: none"> — $\langle int \rangle \leq 0$: no specific color logging. — $\langle int \rangle \geq 1$: ignored color definitions due to <code>\providecolor</code> are logged. — $\langle int \rangle \geq 2$: multiple (i.e. overwritten) color definitions are logged. — $\langle int \rangle \geq 3$: every command that defines a color will be logged. — $\langle int \rangle \geq 4$: every command that sets a color will be logged.

Like T_EX'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

<code>\convertcolorspec</code>	<code>{\model}{\spec}{\target model}{\cmd}</code> Converts a color, given by the $\langle spec \rangle$ in model $\langle model \rangle$, into $\langle target model \rangle$ and stores the new color specification in <code>\cmd</code> . $\langle target model \rangle$ must be of type $\langle num model \rangle$, whereas $\langle model \rangle$ may also be 'named', in which case $\langle spec \rangle$ is simply the name of the color. Example : <code>\convertcolorspec{cmyk}{0.81,1,0,0.07}{HTML}\tmp</code> acts like <code>\def\tmp{1F00ED}</code> .
--------------------------------	--

2.15 Problems and solutions

2.15.1 Name clashes 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 `\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  under the regime of `dvipsnames` and  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 :

13. This should not happen since usually there is no reason to load `color` in parallel to `xcolor`.

```
\usepackage[dvipsnames*,svgnames]{xcolor}
\definecolors{Fuchsia}
```

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

2.15.2 Page breaks and pdfTeX

Since pdfTeX does not maintain a *color stack* — in contrast to *dvips* — a typical problem is the behaviour of colors in the case of page breaks, as illustrated by the following example :

```
\documentclass{minimal}
\usepackage{xcolor}
\begin{document}
black\color{red}red1\newpage red2\color{black}black
\end{document}
```

This works as expected with *dvips*, i.e., ‘red1’ and ‘red2’ being *red*, however, with *pdftex*, ‘red2’ is displayed in *black*. The problem may be solved by using the *fixpdftex* option which simply loads Heiko Oberdiek’s *pdfcolmk* package [13]. However, its author also lists some limitations :

- Mark limitations : page breaks in math.
- LaTeX’s output routine is redefined.
 - Changes in the output routine of newer versions of LaTeX are not detected.
 - Packages that change the output routine are not supported.
- It does not support several independent text streams like footnotes.

2.15.3 Change color of included .eps file

In general, *xcolor* cannot change colors of an image that is being included via the `\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 :

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 60 12
0 0 60 12 rectfill
0.75 setgray
2 2 56 8 rectfill
```

Now run the following code through L^AT_EX and *dvips* :

```
\documentclass{minimal}
\usepackage[fixinclude]{xcolor}
\usepackage{graphics}
```

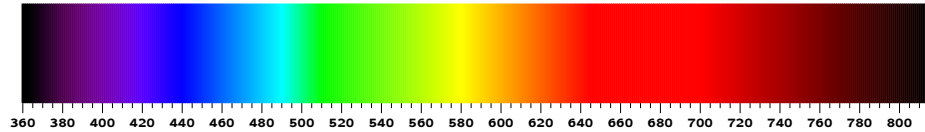


```
\begin{document}  
\includegraphics{foo} \textcolor{red}{\includegraphics{foo}}  
\end{document}
```

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



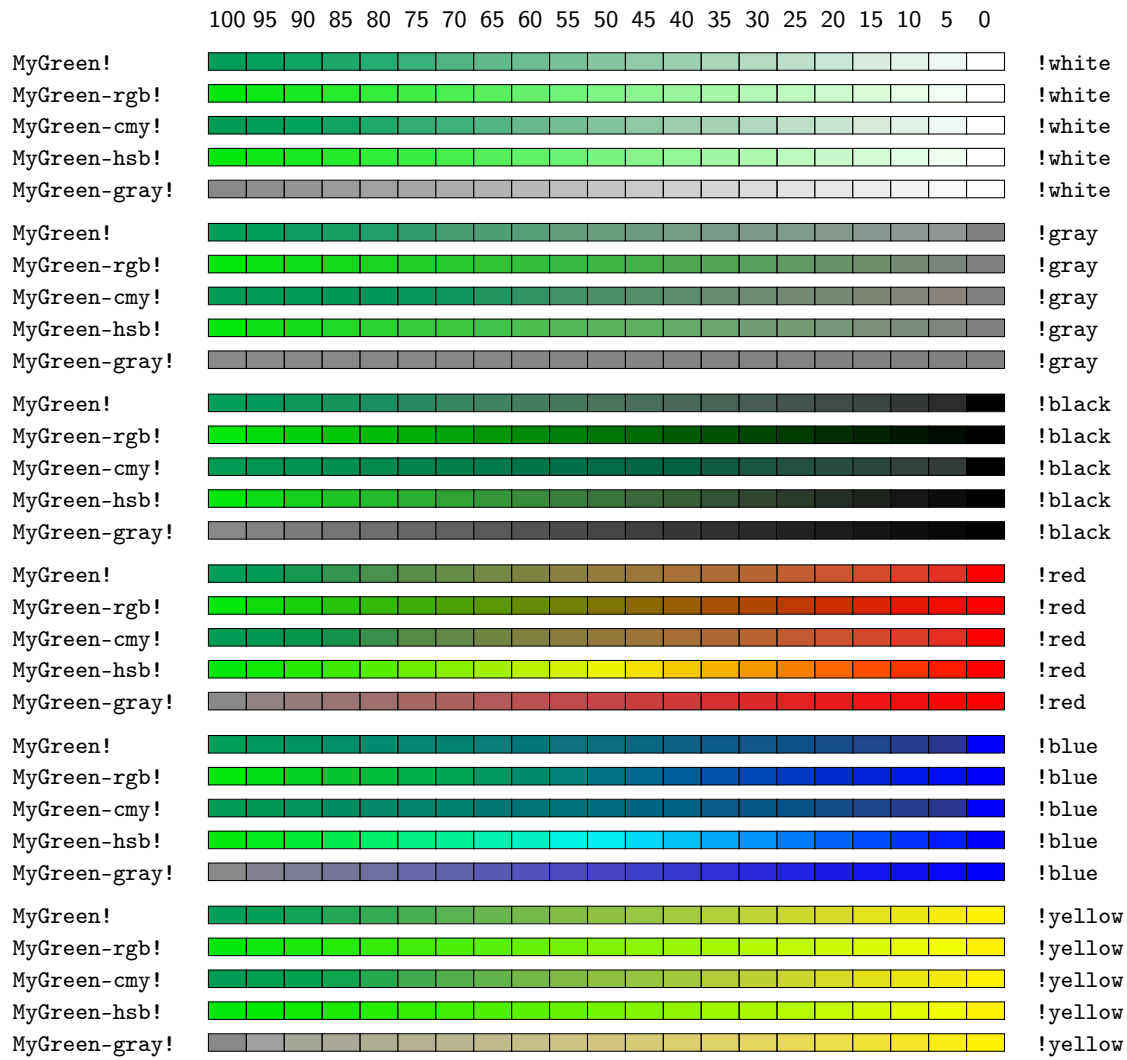
```
\newcount\WL \unitlength.75pt
\begin{picture}(460,60)(355,-10)
\sffamily \tiny \linethickness{1.25\unitlength} \WL=360
\multiput(360,0)(1,0){456}%
{\color[wave]{\the\WL}\line(0,1){50}}\global\advance\WL1}
\linethickness{0.25\unitlength}\WL=360
\multiput(360,0)(20,0){23}%
{\picture(0,0)
\line(0,-1){5}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

color	rgb	cmYk	hsb	HTML	gray
olive	 0.5 0.5 0	 <u>0 0 1 0.5</u>	 0.16667 1 0.5	 808000	 0.39
red!50!green	 <u>0.5 0.5 0</u>	 0 0 0.5 0.5	 0.16667 1 0.5	 808000	 0.445
-cyan!50!magenta	 0.5 0.5 0	 <u>0 0 0.5 0.5</u>	 0.16667 1 0.5	 808000	 0.445
[cmYk]0,0,1,0.5	 0.5 0.5 0	 <u>0 0 1 0.5</u>	 0.16667 1 0.5	 808000	 0.39
[cmYk]0,0,.5,.5	 0.5 0.5 0	 <u>0 0 0.5 0.5</u>	 0.16667 1 0.5	 808000	 0.445
[rgb:cmYk]0,0,.5,.5	 <u>0.5 0.5 0</u>	 0 0 0.5 0.5	 0.16667 1 0.5	 808000	 0.445

```
\sffamily
\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



Color	Definition/representation (pdf _{tex} driver)
MyGreen	{0.92 0 0.87 0.09 k 0.92 0 0.87 0.09 K}{cmyk}{0.92,0,0.87,0.09}
MyGreen-rgb	{0 0.91 0.04001 rg 0 0.91 0.04001 RG}{rgb}{0,0.91,0.04001}
MyGreen-cmy	{1 0.09 0.95999 0 k 1 0.09 0.95999 0 K}{cmy}{1,0.09,0.95999}
MyGreen-hsb	{0 0.91 0.03995 rg 0 0.91 0.03995 RG}{hsb}{0.34065,1,0.91}
MyGreen-gray	{0.5383 g 0.5383 G}{gray}{0.5383}

FIGURE 4 – Target color model

<code>\selectcolormodel</code>	
<code>...{natural}</code>	
<code>...{rgb}</code>	
<code>...{cmy}</code>	
<code>...{cmyk}</code>	
<code>...{hsb}</code>	
<code>...{gray}</code>	

FIGURE 5 – Standard color expressions

	<code>red</code>		<code>-red</code>
	<code>red!75</code>		<code>-red!75</code>
	<code>red!75!green</code>		<code>-red!75!green</code>
	<code>red!75!green!50</code>		<code>-red!75!green!50</code>
	<code>red!75!green!50!blue</code>		<code>-red!75!green!50!blue</code>
	<code>red!75!green!50!blue!25</code>		<code>-red!75!green!50!blue!25</code>
	<code>red!75!green!50!blue!25!gray</code>		<code>-red!75!green!50!blue!25!gray</code>

FIGURE 6 – Standard color expressions

```

\fbboxrule6pt
\fcboxrule6pt
\fcboxrule6pt
{red!70!green}% outer frame
{yellow!30!blue}% outer background
{\fcboxrule6pt
{-yellow!30!blue}% inner frame
{-red!70!green}% inner background
{Test\textcolor{red!72.75}{Test}\color{-green}Test}}

```



FIGURE 7 – Current color

```

\def\test{current, \textcolor{.!50}{50\%},
\textcolor{-.}{complement},
\textcolor{yellow!50!}{mix}}
\textcolor{blue}{\test}\
and \textcolor{red}{\test}\
\def\Test{\color{.!80}Test}
\textcolor{blue}{\Test\Test\Test\Test\Test}\
and \textcolor{red}{\Test\Test\Test\Test\Test}

```

`current`, 50%, `complement`, mix
and `current`, 50%, `complement`, mix
TestTestTestTestTest
and TestTestTestTestTest

FIGURE 8 – Color series

S_1	S_2	G_1	G_2	L_1	L_2	L_3	L_4	L_5
1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9
10	10	10	10	10	10	10	10	10
11	11	11	11	11	11	11	11	11
12	12	12	12	12	12	12	12	12
13	13	13	13	13	13		13	13
14	14	14	14	14	14	14	14	14
15	15	15	15	15	15	15	15	15
16	16	16	16	16	16	16	16	16

Individual definitions

S_1	<code>\definecolorseries{test}{rgb}{step}{rgb}{.95,.85,.55}{.17,.47,.37}</code>
S_2	<code>\definecolorseries{test}{hsb}{step}{hsb}{.575,1,1}{.11,-.05,0}</code>
G_1	<code>\definecolorseries{test}{rgb}{grad}{rgb}{.95,.85,.55}{3,11,17}</code>
G_2	<code>\definecolorseries{test}{hsb}{grad}{hsb}{.575,1,1}{.987,-.234,0}</code>
L_1	<code>\definecolorseries{test}{rgb}{last}{rgb}{.95,.85,.55}[rgb]{.05,.15,.55}</code>
L_2	<code>\definecolorseries{test}{hsb}{last}{hsb}{.575,1,1}[hsb]{-.425,.15,1}</code>
L_3	<code>\definecolorseries{test}{rgb}{last}{yellow!50}{blue}</code>
L_4	<code>\definecolorseries{test}{hsb}{last}{yellow!50}{blue}</code>
L_5	<code>\definecolorseries{test}{cmy}{last}{yellow!50}{blue}</code>

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\ \number\rownum\
\end{tabular}

```

FIGURE 9 – Color masking

\maskcolors	
...{}	
...[cmyk]{cyan}	
...[cmyk]{magenta}	
...[cmyk]{yellow}	
...[cmyk]{black}	
...[cmyk]{red}	
...[cmyk]{green}	
...[cmyk]{blue}	
...[rgb]{red}	
...[rgb]{green}	
...[rgb]{blue}	
...[hsb]{red}	
...[hsb]{green}	
...[hsb]{blue}	
...[rgb]{gray}	
...[cmy]{gray}	

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

\rowcolors[\hline]{3}{green!25}{yellow!50} \arrayrulecolor{red!75!gray}		
\begin{tabular}{ll}		
test & row \number\rownum\\	test row 1	test row 1
test & row \number\rownum\\	test row 2	test row 2
test & row \number\rownum\\	test row 3	test row 3
\arrayrulecolor{black}		
test & row \number\rownum\\	test row 4	test row 4
test & row \number\rownum\\	test row 5	test row 5
\rowcolor{blue!25}	test row 6	test row 6
test & row \number\rownum\\	test row 7	test row 7
test & row \number\rownum\\	test row 8	test row 8
\hiderowcolors	test row 9	test row 9
test & row \number\rownum\\	test row 10	test row 10
\showrowcolors	test row 11	test row 11
test & row \number\rownum\\	test row 12	test row 12
test & row \number\rownum\\	test row 13	test row 13
\multicolumn{1}%		
{>{\columncolor{red!12}}1}{test} & row \number\rownum\\		
\end{tabular}		

FIGURE 11 – Hsb and tHsb : hue° in 15° steps

















































































































































































































































































































































































color	rgb	cmYk	hsb	Hsb	tHsb
[Hsb]0,1,1	 1 0 0	 0 1 1 0	 0 1 1	 0 1 1	 0 1 1
[Hsb]15,1,1	 1 0.25002 0	 0 0.74998 1 0	 0.04167 1 1	 15.00128 1 1	 30.00256 1 1
[Hsb]30,1,1	 1 0.49998 0	 0 0.50002 1 0	 0.08333 1 1	 29.99872 1 1	 59.99744 1 1
[Hsb]45,1,1	 1 0.75 0	 0 0.25 1 0	 0.125 1 1	 45 1 1	 90 1 1
[Hsb]60,1,1	 0.99998 1 0	 0.00002 0 1 0	 0.16667 1 1	 60.00128 1 1	 120.00128 1 1
[Hsb]75,1,1	 0.75002 1 0	 0.24998 0 1 0	 0.20833 1 1	 74.99872 1 1	 134.99872 1 1
[Hsb]90,1,1	 0.5 1 0	 0.5 0 1 0	 0.25 1 1	 90 1 1	 150 1 1
[Hsb]105,1,1	 0.24998 1 0	 0.75002 0 1 0	 0.29167 1 1	 105.00128 1 1	 165.00128 1 1
[Hsb]120,1,1	 0.00002 1 0	 0.99998 0 1 0	 0.33333 1 1	 119.99872 1 1	 179.99872 1 1
[Hsb]135,1,1	 0 1 0.25	 1 0 0.75 0	 0.375 1 1	 135 1 1	 187.5 1 1
[Hsb]150,1,1	 0 1 0.50002	 1 0 0.49998 0	 0.41667 1 1	 150.00128 1 1	 195.00064 1 1
[Hsb]165,1,1	 0 1 0.74998	 1 0 0.25002 0	 0.45833 1 1	 164.99872 1 1	 202.49936 1 1
[Hsb]180,1,1	 0 1 1	 1 0 0 0	 0.5 1 1	 180 1 1	 210 1 1
[Hsb]195,1,1	 0 0.74998 1	 1 0.25002 0 0	 0.54167 1 1	 195.00128 1 1	 217.50064 1 1
[Hsb]210,1,1	 0 0.50002 1	 1 0.49998 0 0	 0.58333 1 1	 209.99872 1 1	 224.99936 1 1
[Hsb]225,1,1	 0 0.25 1	 1 0.75 0 0	 0.625 1 1	 225 1 1	 232.5 1 1
[Hsb]240,1,1	 0.00002 0 1	 0.99998 1 0 0	 0.66667 1 1	 240.00128 1 1	 240.00128 1 1
[Hsb]255,1,1	 0.24998 0 1	 0.75002 1 0 0	 0.70833 1 1	 254.99872 1 1	 254.99872 1 1
[Hsb]270,1,1	 0.5 0 1	 0.5 1 0 0	 0.75 1 1	 270 1 1	 270 1 1
[Hsb]285,1,1	 0.75002 0 1	 0.24998 1 0 0	 0.79167 1 1	 285.00128 1 1	 285.00128 1 1
[Hsb]300,1,1	 0.99998 0 1	 0.00002 1 0 0	 0.83333 1 1	 299.99872 1 1	 299.99872 1 1
[Hsb]315,1,1	 1 0 0.75	 0 1 0.25 0	 0.875 1 1	 315 1 1	 315 1 1
[Hsb]330,1,1	 1 0 0.49998	 0 1 0.50002 0	 0.91667 1 1	 330.00128 1 1	 330.00128 1 1
[Hsb]345,1,1	 1 0 0.25002	 0 1 0.74998 0	 0.95833 1 1	 344.99872 1 1	 344.99872 1 1
[Hsb]360,1,1	 1 0 0	 0 1 1 0	 1 1 1	 360 1 1	 360 1 1
[tHsb]0,1,1	 1 0 0	 0 1 1 0	 0 1 1	 0 1 1	 0 1 1
[tHsb]15,1,1	 1 0.12498 0	 0 0.87502 1 0	 0.02083 1 1	 7.49872 1 1	 14.99744 1 1
[tHsb]30,1,1	 1 0.25002 0	 0 0.74998 1 0	 0.04167 1 1	 15.00128 1 1	 30.00256 1 1
[tHsb]45,1,1	 1 0.375 0	 0 0.625 1 0	 0.0625 1 1	 22.5 1 1	 45 1 1
[tHsb]60,1,1	 1 0.49998 0	 0 0.50002 1 0	 0.08333 1 1	 29.99872 1 1	 59.99744 1 1
[tHsb]75,1,1	 1 0.62502 0	 0 0.37498 1 0	 0.10417 1 1	 37.50128 1 1	 75.00256 1 1
[tHsb]90,1,1	 1 0.75 0	 0 0.25 1 0	 0.125 1 1	 45 1 1	 90 1 1
[tHsb]105,1,1	 1 0.87498 0	 0 0.12502 1 0	 0.14583 1 1	 52.49872 1 1	 104.99744 1 1
[tHsb]120,1,1	 0.99998 1 0	 0.00002 0 1 0	 0.16667 1 1	 60.00128 1 1	 120.00128 1 1
[tHsb]135,1,1	 0.75002 1 0	 0.24998 0 1 0	 0.20833 1 1	 74.99872 1 1	 134.99872 1 1
[tHsb]150,1,1	 0.5 1 0	 0.5 0 1 0	 0.25 1 1	 90 1 1	 150 1 1
[tHsb]165,1,1	 0.24998 1 0	 0.75002 0 1 0	 0.29167 1 1	 105.00128 1 1	 165.00128 1 1
[tHsb]180,1,1	 0.00002 1 0	 0.99998 0 1 0	 0.33333 1 1	 119.99872 1 1	 179.99872 1 1
[tHsb]195,1,1	 0 1 0.50002	 1 0 0.49998 0	 0.41667 1 1	 150.00128 1 1	 195.00064 1 1
[tHsb]210,1,1	 0 1 1	 1 0 0 0	 0.5 1 1	 180 1 1	 210 1 1
[tHsb]225,1,1	 0 0.50002 1	 1 0.49998 0 0	 0.58333 1 1	 209.99872 1 1	 224.99936 1 1
[tHsb]240,1,1	 0.00002 0 1	 0.99998 1 0 0	 0.66667 1 1	 240.00128 1 1	 240.00128 1 1
[tHsb]255,1,1	 0.24998 0 1	 0.75002 1 0 0	 0.70833 1 1	 254.99872 1 1	 254.99872 1 1
[tHsb]270,1,1	 0.5 0 1	 0.5 1 0 0	 0.75 1 1	 270 1 1	 270 1 1
[tHsb]285,1,1	 0.75002 0 1	 0.24998 1 0 0	 0.79167 1 1	 285.00128 1 1	 285.00128 1 1
[tHsb]300,1,1	 0.99998 0 1	 0.00002 1 0 0	 0.83333 1 1	 299.99872 1 1	 299.99872 1 1
[tHsb]315,1,1	 1 0 0.75	 0 1 0.25 0	 0.875 1 1	 315 1 1	 315 1 1
[tHsb]330,1,1	 1 0 0.49998	 0 1 0.50002 0	 0.91667 1 1	 330.00128 1 1	 330.00128 1 1
[tHsb]345,1,1	 1 0 0.25002	 0 1 0.74998 0	 0.95833 1 1	 344.99872 1 1	 344.99872 1 1
[tHsb]360,1,1	 1 0 0	 0 1 1 0	 1 1 1	 360 1 1	 360 1 1

FIGURE 12 – Color harmony

color	rgb	cmyk	Hsb	tHsb
<i>complementary colors (two-color harmony) :</i>				
yellow>wheel,1,2	 0.00002 0 1	 0.99998 1 0 0	 240.00128 1 1	 240.00128 1 1
yellow	 1 1 0	 0 0 1 0	 60.00128 1 1	 120.00128 1 1
yellow>twheel,1,2	 1 0 0.99995	 0 1 0.00005 0	 300.00256 1 1	 300.00256 1 1
<i>color triad (three-color harmony) :</i>				
yellow>wheel,2,3	 1 0 0.99995	 0 1 0.00005 0	 300.00256 1 1	 300.00256 1 1
yellow>wheel,1,3	 0 1 1	 1 0 0 0	 180 1 1	 210 1 1
yellow	 1 1 0	 0 0 1 0	 60.00128 1 1	 120.00128 1 1
yellow>twheel,1,3	 0.00002 0 1	 0.99998 1 0 0	 240.00128 1 1	 240.00128 1 1
yellow>twheel,2,3	 1 0.00012 0	 0 0.99988 1 0	 0.00714 1 1	 0.01428 1 1
<i>color tetrad (four-color harmony) :</i>				
yellow>wheel,3,4	 1 0 0.49998	 0 1 0.50002 0	 330.00128 1 1	 330.00128 1 1
yellow>wheel,2,4	 0.00002 0 1	 0.99998 1 0 0	 240.00128 1 1	 240.00128 1 1
yellow>wheel,1,4	 0 1 0.50002	 1 0 0.49998 0	 150.00128 1 1	 195.00064 1 1
yellow	 1 1 0	 0 0 1 0	 60.00128 1 1	 120.00128 1 1
yellow>twheel,1,4	 0 0.99988 1	 1 0.00012 0 0	 180.00714 1 1	 210.00357 1 1
yellow>twheel,2,4	 1 0 0.99995	 0 1 0.00005 0	 300.00256 1 1	 300.00256 1 1
yellow>twheel,3,4	 1 0.25002 0	 0 0.74998 1 0	 15.00128 1 1	 30.00256 1 1
<i>split complementary colors :</i>				
yellow>wheel,7,12	 0.5 0 1	 0.5 1 0 0	 270 1 1	 270 1 1
yellow>wheel,5,12	 0 0.49995 1	 1 0.50005 0 0	 210.00256 1 1	 225.00128 1 1
yellow	 1 1 0	 0 0 1 0	 60.00128 1 1	 120.00128 1 1
yellow>twheel,5,12	 0.50018 0 1	 0.49982 1 0 0	 270.01099 1 1	 270.01099 1 1
yellow>twheel,7,12	 1 0 0.49998	 0 1 0.50002 0	 330.00128 1 1	 330.00128 1 1
<i>analogous (adjacent) colors :</i>				
yellow>wheel,11,12	 1 0.50005 0	 0 0.49995 1 0	 30.00256 1 1	 60.00513 1 1
yellow>wheel,10,12	 1 0 0	 0 1 1 0	 360 1 1	 360 1 1
yellow>wheel,2,12	 0 1 0.00005	 1 0 0.99995 0	 120.00256 1 1	 180.00128 1 1
yellow>wheel,1,12	 0.5 1 0	 0.5 0 1 0	 90 1 1	 150 1 1
yellow	 1 1 0	 0 0 1 0	 60.00128 1 1	 120.00128 1 1
yellow>twheel,1,12	 0.5 1 0	 0.5 0 1 0	 90 1 1	 150 1 1
yellow>twheel,2,12	 0 1 0.00021	 1 0 0.99979 0	 120.013 1 1	 180.0065 1 1
yellow>twheel,10,12	 1 0.50005 0	 0 0.49995 1 0	 30.00256 1 1	 60.00513 1 1
yellow>twheel,11,12	 1 0.75012 0	 0 0.24988 1 0	 45.00714 1 1	 90.01428 1 1

4 Colors by Name

4.1 Base colors (always available)

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

4.2 Colors via dvipsnames option

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

4.3 Colors via svgnames option

 AliceBlue	 DarkCyan	 DodgerBlue	 LemonChiffon
 AntiqueWhite	 DarkGoldenrod	 FireBrick	 LightBlue
 Aqua	 DarkGray	 FloralWhite	 LightCoral
 Aquamarine	 DarkGreen	 ForestGreen	 LightCyan
 Azure	 DarkGrey	 Fuchsia	 LightGoldenrod
 Beige	 DarkKhaki	 Gainsboro	 LightGoldenrodYellow
 Bisque	 DarkMagenta	 GhostWhite	 LightGray
 Black	 DarkOliveGreen	 Gold	 LightGreen
 BlanchedAlmond	 DarkOrange	 Goldenrod	 LightGrey
 Blue	 DarkOrchid	 Gray	 LightPink
 BlueViolet	 DarkRed	 Green	 LightSalmon
 Brown	 DarkSalmon	 GreenYellow	 LightSeaGreen
 BurlyWood	 DarkSeaGreen	 Grey	 LightSkyBlue
 CadetBlue	 DarkSlateBlue	 Honeydew	 LightSlateBlue
 Chartreuse	 DarkSlateGray	 HotPink	 LightSlateGray
 Chocolate	 DarkSlateGrey	 IndianRed	 LightSlateGrey
 Coral	 DarkTurquoise	 Indigo	 LightSteelBlue
 CornflowerBlue	 DarkViolet	 Ivory	 LightYellow
 Cornsilk	 DeepPink	 Khaki	 Lime
 Crimson	 DeepSkyBlue	 Lavender	 LimeGreen
 Cyan	 DimGray	 LavenderBlush	 Linen
 DarkBlue	 DimGrey	 LawnGreen	 Magenta













 Maroon	 NavyBlue	 PowderBlue	 Snow
 MediumAquamarine	 OldLace	 Purple	 SpringGreen
 MediumBlue	 Olive	 Red	 SteelBlue
 MediumOrchid	 OliveDrab	 RosyBrown	 Tan
 MediumPurple	 Orange	 RoyalBlue	 Teal
 MediumSeaGreen	 OrangeRed	 SaddleBrown	 Thistle
 MediumSlateBlue	 Orchid	 Salmon	 Tomato
 MediumSpringGreen	 PaleGoldenrod	 SandyBrown	 Turquoise
 MediumTurquoise	 PaleGreen	 SeaGreen	 Violet
 MediumVioletRed	 PaleTurquoise	 Seashell	 VioletRed
 MidnightBlue	 PaleVioletRed	 Sienna	 Wheat
 MintCream	 PapayaWhip	 Silver	 White
 MistyRose	 PeachPuff	 SkyBlue	 WhiteSmoke
 Moccasin	 Peru	 SlateBlue	 Yellow
 NavajoWhite	 Pink	 SlateGray	 YellowGreen
 Navy	 Plum	 SlateGrey	

Duplicate colors : Aqua = Cyan, Fuchsia = Magenta; Navy = NavyBlue; Gray = Grey, DarkGray = DarkGrey, LightGray = LightGrey, SlateGray = SlateGrey, DarkSlateGray = DarkSlateGrey, LightSlateGray = LightSlateGrey, DimGray = DimGrey.

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

 AntiqueWhite1	 Burlywood4	 DarkGoldenrod3	 DeepSkyBlue2
 AntiqueWhite2	 CadetBlue1	 DarkGoldenrod4	 DeepSkyBlue3
 AntiqueWhite3	 CadetBlue2	 DarkOliveGreen1	 DeepSkyBlue4
 AntiqueWhite4	 CadetBlue3	 DarkOliveGreen2	 DodgerBlue1
 Aquamarine1	 CadetBlue4	 DarkOliveGreen3	 DodgerBlue2
 Aquamarine2	 Chartreuse1	 DarkOliveGreen4	 DodgerBlue3
 Aquamarine3	 Chartreuse2	 DarkOrange1	 DodgerBlue4
 Aquamarine4	 Chartreuse3	 DarkOrange2	 Firebrick1
 Azure1	 Chartreuse4	 DarkOrange3	 Firebrick2
 Azure2	 Chocolate1	 DarkOrange4	 Firebrick3
 Azure3	 Chocolate2	 DarkOrchid1	 Firebrick4
 Azure4	 Chocolate3	 DarkOrchid2	 Gold1
 Bisque1	 Chocolate4	 DarkOrchid3	 Gold2
 Bisque2	 Coral1	 DarkOrchid4	 Gold3
 Bisque3	 Coral2	 DarkSeaGreen1	 Gold4
 Bisque4	 Coral3	 DarkSeaGreen2	 Goldenrod1
 Blue1	 Coral4	 DarkSeaGreen3	 Goldenrod2
 Blue2	 Cornsilk1	 DarkSeaGreen4	 Goldenrod3
 Blue3	 Cornsilk2	 DarkSlateGray1	 Goldenrod4
 Blue4	 Cornsilk3	 DarkSlateGray2	 Green1
 Brown1	 Cornsilk4	 DarkSlateGray3	 Green2
 Brown2	 Cyan1	 DarkSlateGray4	 Green3
 Brown3	 Cyan2	 DeepPink1	 Green4
 Brown4	 Cyan3	 DeepPink2	 Honeydew1
 Burlywood1	 Cyan4	 DeepPink3	 Honeydew2
 Burlywood2	 DarkGoldenrod1	 DeepPink4	 Honeydew3
 Burlywood3	 DarkGoldenrod2	 DeepSkyBlue1	 Honeydew4

	HotPink1		LightYellow2		PaleVioletRed3		SlateBlue4
	HotPink2		LightYellow3		PaleVioletRed4		SlateGray1
	HotPink3		LightYellow4		PeachPuff1		SlateGray2
	HotPink4		Magenta1		PeachPuff2		SlateGray3
	IndianRed1		Magenta2		PeachPuff3		SlateGray4
	IndianRed2		Magenta3		PeachPuff4		Snow1
	IndianRed3		Magenta4		Pink1		Snow2
	IndianRed4		Maroon1		Pink2		Snow3
	Ivory1		Maroon2		Pink3		Snow4
	Ivory2		Maroon3		Pink4		SpringGreen1
	Ivory3		Maroon4		Plum1		SpringGreen2
	Ivory4		MediumOrchid1		Plum2		SpringGreen3
	Khaki1		MediumOrchid2		Plum3		SpringGreen4
	Khaki2		MediumOrchid3		Plum4		SteelBlue1
	Khaki3		MediumOrchid4		Purple1		SteelBlue2
	Khaki4		MediumPurple1		Purple2		SteelBlue3
	LavenderBlush1		MediumPurple2		Purple3		SteelBlue4
	LavenderBlush2		MediumPurple3		Purple4		Tan1
	LavenderBlush3		MediumPurple4		Red1		Tan2
	LavenderBlush4		MistyRose1		Red2		Tan3
	LemonChiffon1		MistyRose2		Red3		Tan4
	LemonChiffon2		MistyRose3		Red4		Thistle1
	LemonChiffon3		MistyRose4		RosyBrown1		Thistle2
	LemonChiffon4		NavajoWhite1		RosyBrown2		Thistle3
	LightBlue1		NavajoWhite2		RosyBrown3		Thistle4
	LightBlue2		NavajoWhite3		RosyBrown4		Tomato1
	LightBlue3		NavajoWhite4		RoyalBlue1		Tomato2
	LightBlue4		OliveDrab1		RoyalBlue2		Tomato3
	LightCyan1		OliveDrab2		RoyalBlue3		Tomato4
	LightCyan2		OliveDrab3		RoyalBlue4		Turquoise1
	LightCyan3		OliveDrab4		Salmon1		Turquoise2
	LightCyan4		Orange1		Salmon2		Turquoise3
	LightGoldenrod1		Orange2		Salmon3		Turquoise4
	LightGoldenrod2		Orange3		Salmon4		VioletRed1
	LightGoldenrod3		Orange4		SeaGreen1		VioletRed2
	LightGoldenrod4		OrangeRed1		SeaGreen2		VioletRed3
	LightPink1		OrangeRed2		SeaGreen3		VioletRed4
	LightPink2		OrangeRed3		SeaGreen4		Wheat1
	LightPink3		OrangeRed4		Seashell1		Wheat2
	LightPink4		Orchid1		Seashell2		Wheat3
	LightSalmon1		Orchid2		Seashell3		Wheat4
	LightSalmon2		Orchid3		Seashell4		Yellow1
	LightSalmon3		Orchid4		Sienna1		Yellow2
	LightSalmon4		PaleGreen1		Sienna2		Yellow3
	LightSkyBlue1		PaleGreen2		Sienna3		Yellow4
	LightSkyBlue2		PaleGreen3		Sienna4		Gray0
	LightSkyBlue3		PaleGreen4		SkyBlue1		Green0
	LightSkyBlue4		PaleTurquoise1		SkyBlue2		Grey0
	LightSteelBlue1		PaleTurquoise2		SkyBlue3		Maroon0
	LightSteelBlue2		PaleTurquoise3		SkyBlue4		Purple0
	LightSteelBlue3		PaleTurquoise4		SlateBlue1		
	LightSteelBlue4		PaleVioletRed1		SlateBlue2		
	LightYellow1		PaleVioletRed2		SlateBlue3		

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 [11] distributions and their color model support. Probably, other distributions behave similarly.

TABLE 5 – Drivers and color models

<i>Driver</i>	<i>Version</i>	rgb	cm	my	cm	yk	hsb	gray	RGB	HTML	HSB	Gray
dvipdf	1999/02/16 v3.0i	d	n	d	n	d	n	d	i	n	n	n
dvips	1999/02/16 v3.0i	d	n	d	d	d	d	d	i	n	n	n
dvipsone	1999/02/16 v3.0i	d	n	d	d	d	d	d	i	n	n	n
pctex32	1999/02/16 v3.0i	d	n	d	d	d	d	d	i	n	n	n
pctexps	1999/02/16 v3.0i	d	n	d	d	d	d	d	i	n	n	n
pdftex	2006/03/02 v0.03p	d	n	d	n	d	n	d	i	n	n	n
dvipdfm	1998/11/24 vx.x ¹	d	n	d	a	d	i	n	n	n	n	n
dvipdfm	1999/9/6 vx.x ²	d	n	d	a	d	i	n	n	n	n	n
dvipdfmx	?	d	n	d	f	d	i	n	n	n	n	n
textures	1997/5/28 v0.3	d	n	d	a	i	n	n	n	n	n	n
vtex	1999/01/14 v6.3	d	n	d	n	i	i	n	n	n	n	n
xetex	2004/05/09 v0.7	i	n	i	i	i	i	d	n	n	n	n
tcidvi	1999/02/16 v3.0i	i	n	i	n	i	d	n	n	n	n	n
truotex	1999/02/16 v3.0i	i	n	i	n	i	d	n	n	n	n	n
dviwin	1999/02/16 v3.0i	n	n	n	n	n	n	n	n	n	n	n
emtex	1999/02/16 v3.0i	n	n	n	n	n	n	n	n	n	n	n
pctexhp	1999/02/16 v3.0i	n	n	n	n	n	n	n	n	n	n	n
pctexwin	1999/02/16 v3.0i	n	n	n	n	n	n	n	n	n	n	n
dviwindo = dvipsone; oztex = dvips; xdvi = dvips + monochrome												
¹ part of graphics package ² additionally distributed with MiKTeX												
Driver's color model support : d = direct, i = indirect, a = alleged, n = none, f = faulty												

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 extension **color**. 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`'s `\definecolor{foo}{...}{...}` generates a command `\color@foo`¹⁴ 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 `colorinfo` package [12] for a solution).

`color`'s `\DefineNamedColor{named}{foo}{...}{...}` generates `\col@foo`¹⁵ which again contains some driver-dependent information. In this case, an additional `\color@foo` will only be defined if the package option `usecolors` is active.

`xcolor`'s `\definecolor{foo}{...}{...}` generates¹⁶ a command `\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, `\color@foo` expands to the same expression as the original command. On the other hand, `\col@foo` commands are no longer needed and therefore not generated in the 'named' case : `xcolor` works with a single color data structure (as described).

Table 6 on the next page shows some examples for the two most prominent drivers. See also figure 3 on page 35 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.¹⁷ 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 `realcalc` or `fp`. The marginal contribution of the latter packages seems not to justify their usage for our purposes. Thus, we stay within a sort of

14. The double backslash is intentional.

15. The single backslash is intentional.

16. This was introduced in version 1.10; prior to that, a command `\xcolor@foo` with a different syntax was generated.

17. For example, applying the 'transformation' `\dimen0=0.<int>pt \the\dimen0` to all 5-digit numbers `<int>` of the range 00000...99999, exactly 34464 of these 100000 numbers don't survive unchanged. We are not talking about gobbled final zeros here ...

TABLE 6 – Driver-dependent internal color representation

dvips driver		
<code>\\color@Plum=macro:</code>	<code>(\\definecolor{Plum}{rgb}{.5,0,1})</code>	color
<code>->rgb .5 0 1.</code>		
<code>\\color@Plum=macro:</code>	<code>(\\definecolor{Plum}{rgb}{.5,0,1})</code>	xcolor
<code>->\\xcolor@ {}{rgb 0.5 0 1}{rgb}{0.5,0,1}.</code>		
<code>\\col@Plum=macro:</code>	<code>(\\DefineNamedColor{Plum}{rgb}{.5,0,1})</code>	color
<code>->\\@nil .</code>		
<code>\\color@Plum=macro:</code>	<code>(with option usenames)</code>	
<code>-> Plum.</code>		
<code>\\color@Plum=macro:</code>	<code>(\\definecolor[named]{Plum}{rgb}{.5,0,1})</code>	xcolor
<code>->\\xcolor@ {named}{ Plum}{rgb}{0.5,0,1}.</code>		
pdftex driver		
<code>\\color@Plum=macro:</code>	<code>(\\definecolor{Plum}{rgb}{.5,0,1})</code>	color
<code>->.5 0 1 rg .5 0 1 RG.</code>		
<code>\\color@Plum=macro:</code>	<code>(\\definecolor{Plum}{rgb}{.5,0,1})</code>	xcolor
<code>->\\xcolor@ {}{0.5 0 1 rg 0.5 0 1 RG}{rgb}{0.5,0,1}.</code>		
<code>\\col@Plum=macro:</code>	<code>(\\DefineNamedColor{Plum}{rgb}{.5,0,1})</code>	color
<code>->.5 0 1 rg .5 0 1 RG.</code>		
<code>\\color@Plum=macro:</code>	<code>(with option usenames)</code>	
<code>->.5 0 1 rg .5 0 1 RG.</code>		
<code>\\color@Plum=macro:</code>	<code>(\\definecolor[named]{Plum}{rgb}{.5,0,1})</code>	xcolor
<code>->\\xcolor@ {}{0.5 0 1 rg 0.5 0 1 RG}{rgb}{0.5,0,1}.</code>		

fixed-point arithmetic framework, providing at most 5 decimal digits via T_EX's dimension registers.

6 The Formulas

6.1 Color mixing

In general, we use linear interpolation for color mixing :

$$\text{mélange}(C, C', p) = p \cdot C + (1 - p) \cdot C' \quad (9)$$

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

$$\text{teinte}_{\mathbf{hsb}}(C, p) = p \cdot C + (1 - p) \cdot (\text{hue}, 0, 1) \quad (10)$$

$$\text{nuance}_{\mathbf{hsb}}(C, p) = p \cdot C + (1 - p) \cdot (\text{hue}, 0, 0) \quad (11)$$

$$\text{ton}_{\mathbf{hsb}}(C, p) = p \cdot C + (1 - p) \cdot (\text{hue}, 0, \frac{1}{2}) \quad (12)$$

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 13, it is an easy proof by induction to verify that a color expression

$$C_0!P_1!C_1!P_2!\dots!P_n!C_n \quad (13)$$

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

$$\begin{aligned} C &= \sum_{\nu=0}^n \left(\prod_{\mu=\nu+1}^n p_{\mu} \right) (1 - p_{\nu}) \cdot C_{\nu} \\ &= p_n \cdots p_1 \cdot C_0 \\ &\quad + p_n \cdots p_2 (1 - p_1) \cdot C_1 \\ &\quad + p_n \cdots p_3 (1 - p_2) \cdot C_2 \\ &\quad + \dots \\ &\quad + p_n (1 - p_{n-1}) \cdot C_{n-1} \\ &\quad + (1 - p_n) \cdot C_n \end{aligned} \quad (14)$$

where $p_0 := 0$ and $p_{\nu} := P_{\nu}/100$ for $\nu = 1, \dots, n$. We note also a split formula :

$$\begin{aligned} C_0!P_1!C_1!\dots!P_{n+k}!C_{n+k} &= p_{n+k} \cdots p_{n+1} \cdot C_0!P_1!C_1!\dots!P_n!C_n \\ &\quad - p_{n+k} \cdots p_{n+1} \cdot C_n \\ &\quad + C_n!P_{n+1}!C_{n+1}!\dots!P_{n+k}!C_{n+k} \end{aligned} \quad (15)$$

6.2 Conversion between integer and real models

We fix a positive integer n and define the sets $\mathcal{I}_n := \{0, 1, \dots, n\}$ 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

<i>model/constant</i>	white	black	gray
rgb	(1, 1, 1)	(0, 0, 0)	$(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$
cmy	(0, 0, 0)	(1, 1, 1)	$(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$
cmyk	(0, 0, 0, 0)	(0, 0, 0, 1)	$(0, 0, 0, \frac{1}{2})$
hsb	$(h, 0, 1)$	$(h, 0, 0)$	$(h, 0, \frac{1}{2})$
Hsb	$(h^\circ, 0, 1)$	$(h^\circ, 0, 0)$	$(h^\circ, 0, \frac{1}{2})$
tHsb	$(h^\circ, 0, 1)$	$(h^\circ, 0, 0)$	$(h^\circ, 0, \frac{1}{2})$
gray	1	0	$\frac{1}{2}$
RGB	(L, L, L)	(0, 0, 0)	$(\lfloor \frac{L+1}{2} \rfloor, \lfloor \frac{L+1}{2} \rfloor, \lfloor \frac{L+1}{2} \rfloor)$
HTML	FFFFFF	000000	808080
HSB	$(H, 0, M)$	$(H, 0, 0)$	$(H, 0, \lfloor \frac{M+1}{2} \rfloor)$
Gray	N	0	$\lfloor \frac{N+1}{2} \rfloor$

TABLE 8 – Color conversion pairs

<i>from/to</i>	rgb	cmy	cmyk	hsb	Hsb	tHsb	gray	RGB	HTML	HSB	Gray
rgb	id	*	(cmy)	*	(hsb)	(hsb)	*	*	*	(hsb)	(gray)
cmy	*	id	*	(rgb)	(rgb)	(rgb)	*	(rgb)	(rgb)	(rgb)	(gray)
cmyk	(cmy)	*	id	(cmy)	(cmy)	(cmy)	*	(cmy)	(cmy)	(cmy)	(gray)
hsb	*	(rgb)	(rgb)	id	*	(Hsb)	(rgb)	(rgb)	(rgb)	*	(rgb)
Hsb	(hsb)	(hsb)	(hsb)	*	id	*	(hsb)	(hsb)	(hsb)	(hsb)	(hsb)
tHsb	(Hsb)	(Hsb)	(Hsb)	(Hsb)	*	id	(Hsb)	(Hsb)	(Hsb)	(Hsb)	(Hsb)
gray	*	*	*	*	*	*	id	*	*	*	*
RGB	*	(rgb)	(rgb)	(rgb)	(rgb)	(rgb)	(rgb)	id	(rgb)	(rgb)	(rgb)
HTML	*	(rgb)	(rgb)	(rgb)	(rgb)	(rgb)	(rgb)	(rgb)	id	(rgb)	(rgb)
HSB	(hsb)	(hsb)	(hsb)	*	(hsb)	(hsb)	(hsb)	(hsb)	(hsb)	id	(hsb)
Gray	(gray)	(gray)	(gray)	(gray)	(gray)	(gray)	*	(gray)	(gray)	(gray)	id
wave	(hsb)	(hsb)	(hsb)	*	(hsb)	(hsb)	(hsb)	(hsb)	(hsb)	(hsb)	(hsb)

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

$$\Gamma_n : \mathcal{R} \rightarrow \mathcal{I}_n, \quad x \mapsto \text{round}(n \cdot x, 0) = \lfloor \tfrac{1}{2} + n \cdot x \rfloor \quad (16)$$

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 $x \in \mathcal{R}$,*

$$\Gamma_n(x) + \Gamma_n(1 - x) = n \iff x \notin \mathcal{R}_n^\circ := \left\{ \tfrac{1}{n} \left(\nu - \tfrac{1}{2} \right) \mid \nu = 1, 2, \dots, n \right\}. \quad (17)$$

Démonstration. 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, $\eta = -\frac{1}{2} \iff x = \frac{1}{n} \left(\nu - \frac{1}{2} \right) \iff x \in \mathcal{I}'_n$. \square

Remark : the set \mathcal{R}_n° is obviously identical to the set of points where Γ_n is not continuous.

6.2.2 Integer to real conversion

The straightforward way in this case is the function

$$\Delta_n^* : \mathcal{I}_n \rightarrow \mathcal{R}, \quad \nu \mapsto \frac{\nu}{n}. \quad (18)$$

This is, however, only one out of a variety of solutions : every function $\Delta_n : \mathcal{I}_n \rightarrow \mathcal{R}$ that obeys the condition

$$\nu \in \mathcal{I}_n \Rightarrow \Gamma_n(\Delta_n(\nu)) = \nu \quad (19)$$

which is equivalent to

$$\nu \in \mathcal{I}_n \Rightarrow \nu + \frac{1}{2} > n \cdot \Delta_n(\nu) \geq \nu - \frac{1}{2} \quad (20)$$

does at least guarantee that all integers ν may be reconstructed from $\Delta_n(\nu)$ via multiplication by n and rounding to the nearest integer. Preservation of complements means now

$$\nu \in \mathcal{I}_n \Rightarrow \Delta_n(\nu) + \Delta_n(n - \nu) = 1 \quad (21)$$

which is obviously the case for $\Delta_n = \Delta_n^*$. If we consider, more generally, a transformation

$$\Delta_n(\nu) = \frac{\nu + \alpha}{n + \beta} \quad (22)$$

with $\beta \neq -n$, then the magic inequality (20) is equivalent to

$$\frac{1}{2} > \frac{\alpha n - \beta \nu}{n + \beta} \geq -\frac{1}{2} \quad (23)$$

which is obeyed by the function

$$\Delta'_n : \mathcal{I}_n \rightarrow \mathcal{R}, \nu \mapsto \begin{cases} \frac{\nu}{n+1} & \text{if } \nu \leq \frac{n+1}{2} \\ \frac{\nu+1}{n+1} & \text{if } \nu > \frac{n+1}{2} \end{cases} \quad (24)$$

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

Lemma 2 (Preservation of complements). *For odd n and each $\nu \in \mathcal{I}_n$,*

$$\Delta'_n(\nu) + \Delta'_n(n - \nu) = 1 \iff \nu \notin \mathcal{I}_n^\circ := \left\{ \frac{n-1}{2}, \frac{n+1}{2} \right\}. \quad (25)$$

Démonstration. The assertion is a consequence of the following arguments :

- $\nu < \frac{n-1}{2} \iff n - \nu > \frac{n+1}{2}$ and $\frac{n-1}{2} + \frac{n+1}{2} = n$;
- $\nu < \frac{n-1}{2} \Rightarrow \Delta'_n(\nu) + \Delta'_n(n - \nu) = \frac{\nu}{n+1} + \frac{n-\nu+1}{n+1} = 1$;
- $\nu = \frac{n-1}{2} \Rightarrow \Delta'_n(\nu) + \Delta'_n(n - \nu) = \frac{n-1}{2(n+1)} + \frac{1}{2} = \frac{n}{n+1} \neq 1$. □

For the time being, we choose $\boxed{\Delta_n := \Delta_n^*}$ 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 :

$$\Delta''_n : \mathcal{I}_n \rightarrow \mathcal{R}, \nu \mapsto \text{round}\left(\frac{\nu}{n}, d_n\left(\frac{\nu}{n}\right)\right) \quad (26)$$

where

$$d_n : [0, 1] \rightarrow \mathbb{N}, x \mapsto \min\{d \in \mathbb{N} \mid \Gamma_n(\text{round}(\Delta_n^*(\Gamma_n(x)), d)) = \Gamma_n(x)\} \quad (27)$$

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 48 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**, **cmymk**, **hsb**, and **gray** natively. Furthermore, Alvy Ray Smith's paper [15] is cited in [1] as reference for **hsb**-related formulas.

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 **cm**** Source : [1], p. 475.

$$(cyan, magenta, yellow) := E - (red, green, blue) \quad (29)$$

Conversion **rgb to **hsb** (1)** We set

$$x := \max\{red, green, blue\} \quad (30)$$

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

$$z := \min\{red, green, blue\} \quad (32)$$

$$(33)$$

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

$$brightness := x \quad (34)$$

Case $x = z$:

$$saturation := 0 \quad (35)$$

$$hue := 0 \quad (36)$$

Case $x \neq z$:

$$saturation := \frac{x - z}{x} \quad (37)$$

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

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

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 β be a function that takes a Boolean expression as argument and returns 1 if the expression is true, 0 otherwise; set

$$i := 4 \cdot \beta(\text{red} \geq \text{green}) + 2 \cdot \beta(\text{green} \geq \text{blue}) + \beta(\text{blue} \geq \text{red}), \quad (42)$$

and

$$(\text{hue}, \text{saturation}, \text{brightness}) := \begin{cases} \Phi(\text{blue}, \text{green}, \text{red}, 3, 1) & \text{if } i = 1 \\ \Phi(\text{green}, \text{red}, \text{blue}, 1, 1) & \text{if } i = 2 \\ \Phi(\text{green}, \text{blue}, \text{red}, 3, -1) & \text{if } i = 3 \\ \Phi(\text{red}, \text{blue}, \text{green}, 5, 1) & \text{if } i = 4 \\ \Phi(\text{blue}, \text{red}, \text{green}, 5, -1) & \text{if } i = 5 \\ \Phi(\text{red}, \text{green}, \text{blue}, 1, -1) & \text{if } i = 6 \\ (0, 0, \text{blue}) & \text{if } i = 7 \end{cases} \quad (43)$$

where

$$\Phi(x, y, z, u, v) := \left(\frac{u \cdot (x - z) + v \cdot (x - y)}{6(x - z)}, \frac{x - z}{x}, x \right) \quad (44)$$

The singular case $x = z$, which is equivalent to $\text{red} = \text{green} = \text{blue}$, 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 :

$6 \cdot \text{hue}$	Condition	$\text{red} \geq \text{green}$	$\text{green} \geq \text{blue}$	$\text{blue} \geq \text{red}$	i
$1 - f$	$\text{red} \geq \text{green} \geq \text{blue}$	1	1	*	6/7
$1 + f$	$\text{green} \geq \text{red} \geq \text{blue}$	*	1	*	2/3/6/7
$3 - f$	$\text{green} \geq \text{blue} \geq \text{red}$	*	1	1	3/7
$3 + f$	$\text{blue} \geq \text{green} \geq \text{red}$	*	*	1	1/3/5/7
$5 - f$	$\text{blue} \geq \text{red} \geq \text{green}$	1	*	1	5/7
$5 + f$	$\text{red} \geq \text{blue} \geq \text{green}$	1	*	*	4/5/6/7

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.

$$\text{gray} := 0.3 \cdot \text{red} + 0.59 \cdot \text{green} + 0.11 \cdot \text{blue} \quad (45)$$

Conversion rgb to RGB As described in section 6.2.1 on page 49.

$$(\text{Red}, \text{Green}, \text{Blue}) := (\Gamma_L(\text{red}), \Gamma_L(\text{green}), \Gamma_L(\text{blue})) \quad (46)$$

Conversion rgb to HTML As described in section 6.2.1 on page 49. Convert to 6-digit hexadecimal afterwards. Certainly, multiplication and summation can be replaced by simple text concatenation of 2-digit hexadecimals.

$$\text{RRGGBB} := (65536 \cdot \Gamma_L(\text{red}) + 256 \cdot \Gamma_L(\text{green}) + \Gamma_L(\text{blue}))_{\text{hex}} \quad (47)$$

Complement of rgb color We simply take the complementary vector :

$$(\text{red}^*, \text{green}^*, \text{blue}^*) := E - (\text{red}, \text{green}, \text{blue}) \quad (48)$$

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 51.

$$(\text{red}, \text{green}, \text{blue}) := E - (\text{cyan}, \text{magenta}, \text{yellow}) \quad (49)$$

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.

$$k := \min\{\text{cyan}, \text{magenta}, \text{yellow}\} \quad (50)$$

$$\text{cyan} := \min\{1, \max\{0, \text{cyan} - \text{UCR}_c(k)\}\} \quad (51)$$

$$\text{magenta} := \min\{1, \max\{0, \text{magenta} - \text{UCR}_m(k)\}\} \quad (52)$$

$$\text{yellow} := \min\{1, \max\{0, \text{yellow} - \text{UCR}_y(k)\}\} \quad (53)$$

$$\text{black} := \text{BG}(k) \quad (54)$$

Here, four additional functions are required :

$$\begin{aligned} UCR_c, UCR_m, UCR_y : [0, 1] &\rightarrow [-1, 1] && \text{undercolor-removal} \\ BG : [0, 1] &\rightarrow [0, 1] && \text{black-generation} \end{aligned}$$

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 :

$$UCR_c(k) := \beta_c \cdot k \quad (55)$$

$$UCR_m(k) := \beta_m \cdot k \quad (56)$$

$$UCR_y(k) := \beta_y \cdot k \quad (57)$$

$$BG(k) := \beta_k \cdot k \quad (58)$$

`\adjustUCRBG` where the parameters are given by `\def\adjustUCRBG{<\beta_c>,<\beta_m>,<\beta_y>,<\beta_k>}` at any point in a document, defaulting to `{1,1,1,1}`.

Conversion `cm`y to `gray` This is derived from the conversion chain `cm`y \rightarrow `rgb` \rightarrow `gray`.

$$gray := 1 - (0.3 \cdot cyan + 0.59 \cdot magenta + 0.11 \cdot yellow) \quad (59)$$

Complement of `cm`y color We simply take the complementary vector :

$$(cyan^*, magenta^*, yellow^*) := E - (cyan, magenta, yellow) \quad (60)$$

6.3.3 The `cm`yk model

Conversion `cm`yk to `cm`y Based on [1], p. 477, in connection with `rgb` \rightarrow `cm`y conversion.

$$cyan := \min\{1, cyan + black\} \quad (61)$$

$$magenta := \min\{1, magenta + black\} \quad (62)$$

$$yellow := \min\{1, yellow + black\} \quad (63)$$

Conversion `cm`yk to `gray` Source : [1], p. 475.

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

Complement of `cm`yk color The simple vector complement does not yield useful results. Therefore, we first convert $C = (cyan, magenta, yellow, black)$ to the `cm`y model, calculate the complement there, and convert back to `cm`yk.

6.3.4 The hsb model

Conversion hsb to rgb

$$(red, green, blue) := brightness \cdot (E - saturation \cdot F) \quad (65)$$

with

$$i := \lfloor 6 \cdot hue \rfloor, \quad f := 6 \cdot hue - i \quad (66)$$

and

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

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

$$m := 1 - saturation \quad (68)$$

$$n := 1 - f \cdot saturation \quad (69)$$

$$k := 1 - (1 - f) \cdot saturation \quad (70)$$

$$(red, green, blue) := brightness \cdot \begin{cases} (1, k, m) & \text{if } i = 0, 6 \\ (n, 1, m) & \text{if } i = 1 \\ (m, 1, k) & \text{if } i = 2 \\ (m, n, 1) & \text{if } i = 3 \\ (k, m, 1) & \text{if } i = 4 \\ (1, m, n) & \text{if } i = 5 \end{cases} \quad (71)$$

Note that the case $i = 6$ (which results from $hue = 1$) is missing in Smith's algorithm. Because of

$$\lim_{f \rightarrow 1} (0, 1, f) = (0, 1, 1) = \lim_{f \rightarrow 0} (0, 1 - f, 1) \quad (72)$$

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 hue over the whole range $[0, 1]$.

Conversion hsb to Hsb Only the first component has to be changed.

$$(hue^\circ, saturation, brightness) := (H \cdot hue, saturation, brightness) \quad (73)$$

Conversion hsb to HSB As described in section 6.2.1 on page 49.

$$(Hue, Saturation, Brightness) := (\Gamma_M(hue), \Gamma_M(saturation), \Gamma_M(brightness)) \quad (74)$$

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 :*

$$hue^* := \begin{cases} hue + \frac{1}{2} & \text{if } hue < \frac{1}{2} \\ hue - \frac{1}{2} & \text{if } hue \geq \frac{1}{2} \end{cases} \quad (75)$$

$$brightness^* := 1 - brightness \cdot (1 - saturation) \quad (76)$$

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

Démonstration. 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

$$C_{\mathbf{rgb}} + C_{\mathbf{rgb}}^* = b \cdot (E - s \cdot F) + b^* \cdot (E - s' \cdot F^*) \stackrel{!}{=} E, \quad (78)$$

which means that $C_{\mathbf{rgb}}$ is the complement of $C_{\mathbf{rgb}}^*$. 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

$$\begin{aligned} C_{\mathbf{rgb}} + C_{\mathbf{rgb}}^* &= b \cdot (E - s \cdot F) + b^* \cdot \left(E - \frac{b \cdot s}{b^*} \cdot F^* \right) \\ &= b \cdot E - b \cdot s \cdot F + b^* \cdot E - b \cdot s \cdot F^* \\ &= E - b \cdot s \cdot (F + F^* - E) \end{aligned}$$

since $b^* = 1 - b + bs$. Therefore, it is sufficient to show that

$$F + F^* = E. \quad (79)$$

From

$$h < \frac{1}{2} \Rightarrow h^* = h + \frac{1}{2} \Rightarrow 6h^* = 6h + 3 \Rightarrow i^* = i + 3 \text{ and } f^* = f$$

it is easy to see from (67) that equation (79) holds for the cases $i = 0, 1, 2$. Similarly,

$$h \geq \frac{1}{2} \Rightarrow h^* = h - \frac{1}{2} \Rightarrow 6h^* = 6h - 3 \Rightarrow i^* = i - 3 \text{ and } f^* = f$$

and again from (67) we derive (79) for the cases $i = 3, 4, 5$. Finally, if $i = 6$ then $f = 0$ and $F + F^* = (0, 1, 1) + (1, 0, 0) = E$. \square

6.3.5 The Hsb model

Conversion Hsb to hsb Only the first component has to be changed.

$$(hue, saturation, brightness) := (hue^\circ / H, saturation, brightness) \quad (80)$$

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 :

$$hue^\circ \in [y_{\eta-1}, y_\eta] \Rightarrow hue^\circ := x_{\eta-1} + \frac{x_\eta - x_{\eta-1}}{y_\eta - y_{\eta-1}} \cdot (hue^\circ - y_{\eta-1}) \quad (81)$$

while *saturation* and *brightness* are left unchanged.

6.3.6 The tHsb model

Conversion tHsb to Hsb We assume that `\rangeHsb` = H and `\rangetHsb` expands to

$$x_1, y_1; x_2, y_2; \dots; x_{h-1}, y_{h-1} \quad (82)$$

where

$$x_0 := 0 < x_1 < x_2 < \dots < x_{h-1} < x_h := H \quad (83)$$

$$y_0 := 0 < y_1 < y_2 < \dots < y_{h-1} < y_h := H \quad (84)$$

with an integer $h > 0$. Now the x and y values determine a piecewise linear transformation :

$$hue^\circ \in [x_{\eta-1}, x_\eta] \Rightarrow hue^\circ := y_{\eta-1} + \frac{y_\eta - y_{\eta-1}}{x_\eta - x_{\eta-1}} \cdot (hue^\circ - x_{\eta-1}) \quad (85)$$

while *saturation* and *brightness* are left unchanged.

6.3.7 The gray model

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

$$(red, green, blue) := gray \cdot E \quad (86)$$

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

$$(cyan, magenta, yellow) := (1 - gray) \cdot E \quad (87)$$

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

$$(cyan, magenta, yellow, black) := (0, 0, 0, 1 - gray) \quad (88)$$

Conversion gray to hsb This is derived from the conversion chain **gray** \rightarrow **rgb** \rightarrow **hsb**.

$$(hue, saturation, brightness) := (0, 0, gray) \quad (89)$$

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

$$(hue^\circ, saturation, brightness) := (0, 0, gray) \quad (90)$$

Conversion gray to Gray As described in section 6.2.1 on page 49.

$$Gray := \Gamma_N(gray) \quad (91)$$

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

$$gray^* := 1 - gray \quad (92)$$

6.3.8 The RGB model

Conversion RGB to rgb As described in section 6.2.2 on page 49.

$$(red, green, blue) := (\Delta_L(Red), \Delta_L(Green), \Delta_L(Blue)) \quad (93)$$

6.3.9 The HTML model

Conversion HTML to rgb As described in section 6.2.2 on page 49 : starting with *RRGGBB* set

$$(red, green, blue) := (\Delta_{255}(RR_{dec}), \Delta_{255}(GG_{dec}), \Delta_{255}(BB_{dec})) \quad (94)$$

6.3.10 The HSB model

Conversion HSB to hsb As described in section 6.2.2 on page 49.

$$(hue, saturation, brightness) := (\Delta_M(Hue), \Delta_M(Saturation), \Delta_M(Brightness)) \quad (95)$$

6.3.11 The Gray model

Conversion Gray to gray As described in section 6.2.2 on page 49.

$$gray := \Delta_N(Gray) \quad (96)$$

6.3.12 The wave model

Conversion wave to rgb Source : based on Dan Bruton's algorithm [4]. Let λ 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

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

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

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

and finally

$$(red, green, blue) := ((f \cdot r)^\gamma, (f \cdot g)^\gamma, (f \cdot b)^\gamma) \quad (99)$$

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 $hue = \frac{5}{6}, \frac{4}{6}, \frac{3}{6}, \frac{2}{6}, \frac{1}{6}, \frac{0}{6}$, whereas $saturation = brightness = 1$ throughout the 6 colors. Furthermore, these **hsb** representations are independent of the actual γ 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 $^\gamma$!black**, **red!0.3 $^\gamma$!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

$$\varrho : \mathbb{R} \rightarrow [0, 1], \quad x \mapsto (\min\{1, \max\{0, x\}\})^\gamma \quad (100)$$

with a fixed correction number $\gamma > 0$. Then

$$hue := \frac{1}{6} \cdot \begin{cases} 4 + \varrho\left(\frac{\lambda - 440}{380 - 440}\right) & \text{if } \lambda < 440 \\ 4 - \varrho\left(\frac{\lambda - 440}{490 - 440}\right) & \text{if } \lambda \in [440, 490[\\ 2 + \varrho\left(\frac{\lambda - 510}{490 - 510}\right) & \text{if } \lambda \in [490, 510[\\ 2 - \varrho\left(\frac{\lambda - 510}{580 - 510}\right) & \text{if } \lambda \in [510, 580[\\ 0 + \varrho\left(\frac{\lambda - 645}{580 - 645}\right) & \text{if } \lambda \in [580, 645[\\ 0 & \text{if } \lambda \geq 645 \end{cases} \quad (101)$$

$$saturation := 1 \quad (102)$$

$$brightness := \begin{cases} \varrho\left(0.3 + 0.7 \cdot \frac{\lambda - 380}{420 - 380}\right) & \text{if } \lambda < 420 \\ 1 & \text{if } \lambda \in [420, 700] \\ \varrho\left(0.3 + 0.7 \cdot \frac{\lambda - 780}{700 - 780}\right) & \text{if } \lambda > 700 \end{cases} \quad (103)$$

For the sake of completeness we note that, independent of γ ,

$$(hue, saturation, brightness) = \begin{cases} \left(\frac{5}{6}, 1, 0\right) & \text{if } \lambda \leq 380 - \frac{3 \cdot (420 - 380)}{7} = 362.857 \dots \\ (0, 1, 0) & \text{if } \lambda \geq 780 + \frac{3 \cdot (780 - 700)}{7} = 814.285 \dots \end{cases}$$

What is the best (or, at least, a good) value for γ ? 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 `pstricks` examples file `xcolor2.tex`, there is a demonstration of different γ values.

Références

- [1] Adobe Systems Incorporated : “PostScript Language Reference Manual”. Addison-Wesley, third edition, 1999. <http://www.adobe.com/products/postscript/pdfs/PLRM.pdf>
- [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>
- [3] Donald Arseneau : url package, “2005/06/27 ver 3.2 Verb mode for urls, etc.”.
CTAN/macros/latex/contrib/misc/url.sty
- [4] Dan Bruton : “Approximate RGB values for Visible Wavelengths”, 1996.
<http://www.physics.sfasu.edu/astro/color/spectra.html>
- [5] David P. Carlisle : “Packages in the ‘graphics’ bundle”, 2005.
CTAN/macros/latex/required/graphics/grfguide.*
- [6] David P. Carlisle : extension color, “2005/11/14 v1.0j Standard \LaTeX Color”.
CTAN/macros/latex/required/graphics/color.dtx
- [7] David P. Carlisle : colortbl package, “2001/02/13 v0.1j Color table columns”.
CTAN/macros/latex/contrib/colortbl/
- [8] David P. Carlisle, Herbert Voß, Rolf Niepraschk : pstcol package, “2005/11/16 v1.2 \LaTeX wrapper for ‘PSTricks’”. CTAN/macros/graphics/pstricks/latex/pstcol.sty
- [9] Uwe Kern : “Chroma : a reference book of \LaTeX colors”. CTAN/info/colour/chroma/ and
<http://www.ukern.de/tex/chroma.html>
- [10] Uwe Kern : xcolor package, “ \LaTeX color extensions”. CTAN/macros/latex/contrib/xcolor/ and
<http://www.ukern.de/tex/xcolor.html>
- [11] MiKTeX Project : <http://www.miktex.org/>
- [12] Rolf Niepraschk : colorinfo package, “2003/05/04 v0.3c Info from defined colors”.
CTAN/macros/latex/contrib/colorinfo/
- [13] Heiko Oberdiek : pdfcolmk package, “2006/02/20 v0.8 PDFtex COlor MaRK”.
CTAN/macros/latex/contrib/oberdiek/pdfcolmk.*
- [14] Sebastian Rahtz, Heiko Oberdiek : hyperref package, “2006/09/06 v6.75e Hypertext links for \LaTeX ”. CTAN/macros/latex/contrib/hyperref/
- [15] Alvy Ray Smith : “Color Gamut Transform Pairs”. *Computer Graphics* (ACM SIGGRAPH), Volume 12, Number 3, August 1978. <http://alvyray.com/Papers/PapersCG.htm>
- [16] World Wide Web Consortium : “HTML4 color keywords”.
<http://www.w3.org/TR/css3-color/#html4>
- [17] World Wide Web Consortium : “Scalable Vector Graphics (SVG) 1.1 Specification — Basic Data Types and Interfaces”. <http://www.w3.org/TR/SVG11/types.html#ColorKeywords>

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 L^AT_EX ‘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

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 :
 - `fixinclude` 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 **cm**yk-type current color ‘.’.

2005/12/21 v2.09

- New features :
 - `\definecolor` and `\color` now accept space-separated color specifications, e.g., `\color [rgb]{1 .5 0}`;
 - experimental `xcdraw` option extended to `pdftex` and `dvipdfm` drivers.
- 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 :
 - more flexibility for `\fcolorbox` arguments, e.g., `\fcolorbox [gray]{0.5}[wave]{580}{test}`;
 - `\boxframe` returns a frame of given dimensions;
 - new implementation of `\f(rame)box` and `\fcolorbox` as an extension of bug report latex/3655 to reduce pixel positioning errors in output devices;
 - `kernelbbox` option for those who prefer the previous `\f(rame)box`

- approach;
- experimental `xcdraw` option uses PostScript commands to draw frames and color boxes in case of `dvips`.
- Bugfixes :
 - insufficient expression type detection within `\colorlet`;
 - wrong calculation in the unit interval reduction for negative integers (affecting color series and extended color expressions).

2005/11/12 v2.07

- 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 :
 - `fixpdftex` option loads `pdfcolmk` package in order to improve pdf \TeX '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 :
 - `\aftergroupedef` command to reproduce `\aftergroupdef`'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;
 - `\@ifxempty` changed back to more robust variant of v2.00.
- Bugfixes :
 - `\psset{linecolor=\ifcase\foo red\or green\or blue\fi}` did not work with `pstricks` (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;
 - `\colorlet` 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;
 - `\aftergroupdef` now performs only a first-level expansion of its code argument;
 - `\XCfileversion` and similar internal constants removed from `.sty` and `.def` files;
 - improved memory management (reduced generation of ‘multiletter control sequences’ by `\@ifundefined` 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 :
 - extension `color` 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;
 - `\rdivide`’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 `\XC@bcolor`, `\XC@mcolor`, and `\XC@ecolor` for additional code that has to be executed immediately before/after the

- current color is being displayed.
- Changes :
 - `\XC@logcolor` renamed to `\XC@display`, 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 **cmymk**.
- 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 :
 - `\rmultiply` 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^AT_EX kernel.
- Documentation : some minor changes.
- Example files : additional `pstricks` examples (file `xcolor2.tex`).

<i><func expression></i>	16	rgb	6, 9–12, 15, 19, 21, 26, 29, 44, 48, 50–59	\convertcolorspec	31
<i><func expr></i>	14	tHsb	11, 15, 18, 39, 48, 57, 58, 63	D	
<i><function></i>	14, 16	wave	11, 12, 15, 48, 59, 63	\definecolor	20, 30
<i><id-list></i>	14	‘named’	15, 21, 66	\definecolors	23
<i><id></i>	14	‘ps’	15, 63	\definecolorseries	27
<i><int></i>	13, 14	color names		\definecolorset	21
<i><minus></i>	13, 14	Fuchsia	31, 32	\DefineNamedColor	22
<i><mix expr></i>	14, 15	Gray0	19	definition stack	22, 23
<i><model-list></i>	14, 15	Green0	19	E	
<i><model></i>	14, 15	Grey0	19	environments :	
<i><name></i>	13, 14	Maroon0	19	testcolors	25
<i><num model></i>	14, 15	Purple0	19	\extractcolorspec	31
<i><num></i>	13, 14	argent	6	\extractcolorspecs	31
<i><pct></i>	13, 14	black 9, 10, 18, 26, 32, 59, 62		F	
<i><plus></i>	13, 14	blanc	6	\fcolorbox	24
<i><postfix></i>	14, 15	bleu	6, 7	files	
<i><prefix></i>	14, 15	blue	18, 20, 22, 59	.def	65
<i><spec-list></i>	14, 15	brown	18, 66	.dvi	19, 20, 25, 29
<i><spec></i>	14, 15	cyan	18, 26, 59	.eps	10, 27, 32, 33, 62
<i><type></i>	14, 15	darkgray	18	.jpg	27
B		foo	20, 26	.pdf	27
\blendcolors	25	gray	18	.png	27
\blendcolors*	25	green	5, 17, 18, 22, 59	.ps	20, 29, 33
\boxframe	24	gris	6	.sty	65
C		jaune	7	.xcp	10, 20
\color	23	lightgray	18	color.pro	20
color expression	21	lime	18, 62	dvipsnam.def	23
color functions		magenta	18, 26, 59	rgb.txt	19, 63
twheel	18	noir	6	xcolor.pro	8, 20, 64
wheel	18	olive	18, 63	xcolor.sty	8, 18
color models		orange	18, 66	xcolor2.tex	30, 60, 62–64, 66
Gray	10–12, 15, 44, 48, 58	or	6	xcolor3.tex	62
HSB	10–12, 15, 44, 48, 56, 58	pink	18, 63	xcolor4.tex	62, 63
HTML	9–	purple	18, 66	G	
11, 15, 44, 48, 53, 58, 66		red	9, 11, 17, 18, 20–22, 32, 59, 63	\GetGinDriver	8
Hsb	11, 15, 18, 39, 48, 56–58, 63	rouge	5–7	\GinDriver	8
RGB	10–12, 15, 19, 44, 48, 53, 58, 66	teal	18, 62	H	
cmymk	5, 6, 9–	vert	6, 7	\hiderowcolors	30
11, 15, 19, 21, 26, 44,		violet	18, 66	HKS	6
48, 50, 53, 54, 58, 62, 66		white	18	HTML4	19, 42
cmym	9–11, 15, 26, 44, 48, 51, 53, 54, 57	yellow	5, 11, 17, 18, 26, 59	I	
gray	6, 9–12, 15, 17, 44, 48, 50, 53, 54, 57, 58	color set	21	\ifconvertcolorsD	13
hsb	5, 9–12, 15, 18, 21, 44, 45, 47, 48, 50–52, 55–59	color stack	32	\ifconvertcolorsU	13
		\colorbox	24		
		\colorlet	21		
		\colormask	27		
		\colorseriescycle	28		

<i>dvips</i>	8, 20,	<code>\rowcolors</code>	30	<code>testcolors</code> (environment) .	25
	29, 30, 32, 33, 62, 64–66	<code>\rowcolors*</code>	30	<code>\textcolor</code>	23
<code>\providecolor</code>	21	<code>\rownum</code>	30	<code>ton</code>	6
<code>\providecolors</code>	23			<code>ton direct</code>	6
<code>\providecolorset</code>	22	S		<code>\tracingcolors</code>	31
		<code>\selectcolormodel</code>	12		
R		<code>\showrowcolors</code>	30	U	
<code>\rangeGray</code>	12	<code>\substitutecolormodel</code> ..	12	Unix	10, 19, 42, 63
<code>\rangeHSB</code>	12	SVG	10, 19, 32, 61, 65		
<code>\rangeHsb</code>	11, 57			X	
<code>\rangeRGB</code>	12	T		X11	10, 19, 42, 63
<code>\rangetHsb</code>	11, 57	<code>teinte</code>	6	<code>\xcolorcmd</code>	9
<code>\resetcolorseries</code>	28	<code>\testcolor</code>	25	<code>\xglobal</code>	23, 26