

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

Dr. Uwe Kern

v2.11 (21/01/2007) *

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	Exécution de commandes additionnelles à l'initialisation	9
2.2	Modèles colorimétriques	9
2.2.1	Modèles colorimétriques supportés	9
2.2.2	Substitution de modèles colorimétriques individuels	12
2.2.3	Changement du modèle colorimétrique cible dans un document	13
2.3	Arguments et terminologie	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	Remarques additionnelles et restrictions sur les arguments .	13
2.3.2	Signification des expressions de couleur standards	17
2.3.3	Signification des expressions de couleur étendues	18
2.3.4	Fonctions de couleur	18
2.4	Couleurs prédéfinies	19
2.4.1	Couleurs qui sont toujours disponibles	19
2.4.2	Ensembles additionnels de couleurs	19
2.5	Définition de couleur	20
2.5.1	Couleurs ordinaires et nommées	20
2.5.2	Définition de couleur dans xcolor	21
2.5.3	Defining sets of colors	22
2.5.4	Immediate and deferred definitions	23
2.5.5	Global color definitions	24
2.6	Color application	24
2.6.1	Standard color commands	24
2.6.2	Colored boxes	25
2.6.3	Using the current color	26
2.6.4	Color testing	26
2.7	Color blending	26
2.8	Color masks and separation	27
2.9	Color series	28
2.9.1	Definition of a color series	28
2.9.2	Initialisation of a color series	29
2.9.3	Application of a color series	29
2.9.4	Differences between colors and color series	30
2.10	Border colors for hyperlinks	30
2.11	Additional color specification in the pstricks world	31
2.12	Color in tables	31
2.13	Color information	32
2.14	Color conversion	32
2.15	Problems and solutions	32
2.15.1	Name clashes between dvipsnames and svgnames	32
2.15.2	Page breaks and pdfTeX	33
2.15.3	Change color of included .eps file	33
3	Examples	35
4	Colors by Name	42
4.1	Base colors (always available)	42
4.2	Colors via dvipsnames option	42
4.3	Colors via svgnames option	42
4.4	Colors via x11names option	43

5	Technical Supplement	45
5.1	Color models supported by drivers	45
5.2	How xcolor handles driver-specific color models	45
5.3	Behind the scenes : internal color representation	46
5.4	A remark on accuracy	46
6	The Formulas	48
6.1	Color mixing	48
6.2	Conversion between integer and real models	48
6.2.1	Real to integer conversion	50
6.2.2	Integer to real conversion	50
6.3	Color conversion and complements	51
6.3.1	The rgb model	52
6.3.2	The cmY model	54
6.3.3	The cmYk model	55
6.3.4	The hsb model	56
6.3.5	The Hsb model	58
6.3.6	The tHsb model	58
6.3.7	The gray model	58
6.3.8	The RGB model	59
6.3.9	The HTML model	59
6.3.10	The HSB model	59
6.3.11	The Gray model	59
6.3.12	The wave model	60
	Références	62
	Annexes	63
	Acknowledgement	63
	Marques déposées	63
	Problèmes connus	63
	Historique	63
	Index	69

Liste des tableaux

1	Options de l'extension	10
2	Ordre de chargement des extensions	11
3	Modèles colorimétriques supportés	11
4	Arguments et terminologie	14
5	Drivers and color models	45
6	Driver-dependent internal color representation	47
7	Color constants	49
8	Color conversion pairs	49

Table des figures










1	Color spectrum	35
2	Color testing	35
3	Progressing from one to another color	36
4	Target color model	37
5	Standard color expressions	37
6	Standard color expressions	37
7	Current color	37
8	Color series	38
9	Color masking	39
10	Alternating row colors in tables : <code>\rowcolors</code> vs. <code>\rowcolors*</code> . .	39
11	Hsb and tHsb : hue° in 15° steps	40
12	Color harmony	41

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 page 49. 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 page 51 pour le détail des calculs et la section 1.4 page suivante 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 page 41.

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 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`, `cmj`, `cmj`, `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`.

Toutes les options de l'extension (hors les sélections des pilotes et les options obsolètes) sont listées en table 1 page 10. Afin de faciliter la coopération avec l'extension `hyperref`, il existe une commande `\GetGinDriver`² qui récupère le nom du pilote effectivement utilisé et qui le place dans la commande `\GinDriver`.

Ce dernier peut alors être utilisé au sein de l'extension `hyperref` (ou toute autre extension), voir l'exemple de code en page 12. S'il n'y a pas d'option `hyperref` correspondante, l'option `hypertex` sera prise par défaut.

1. La section 2.2.3 page 13 explique comment ce paramétrage peut être annulé n'importe où dans un document.

2. Cette commande est exécutée automatiquement si l'option d'extension `hyperref` est sélectionnée.

Attention : il y a une différence substantielle entre `xcolor` et `color` dans la façon de manier l'option `dvips`. L'extension `color` appelle implicitement l'option `dvipsnames` dès qu'un des pilotes `dvips`, `oztex` ou `xdvi` est sélectionné. Ceci rend les documents moins portables dans la mesure où, à chaque fois qu'une des couleurs est utilisée sans l'appel explicite de l'option `dvipsnames`, les autres pilotes comme `pdftex` renvoient un message d'erreur pour cause de couleur inconnue. C'est pourquoi `xcolor` nécessite toujours explicitement l'option `dvipsnames` pour utiliser ces noms — qui fonctionnent alors pour tous les pilotes.

2.1.3 Exécution de commandes additionnelles à l'initialisation

`\xcolorcmd` Voici un interface simple pour passer des commandes devant être exécutées à la fin de l'extension `xcolor` (immédiatement avant que l'initialisation de `\color{black}` ne soit traitée). Indiquez juste `\def\xcolorcmd{<commandes>}` avant le chargement de `xcolor`.

Exemple : en supposant que `a.tex` soit un document L^AT_EX complet, une commande comme « `latex \def\xcolorcmd{\colorlet{black}{red}}\input{a}` » saisie en invite de commande génère un fichier `a.dvi` avec toutes les occurrences de *noir* remplacées par du *rouge*, sans besoin de modifier le fichier source lui-même. (La ligne de commande peut varier selon les systèmes d'exploitation et les distributions de T_EX).

2.2 Modèles colorimétriques

2.2.1 Modèles colorimétriques supportés

La liste des modèles colorimétriques et les plages de valeur de leurs paramètres sont données en table 3 page 11. Notez bien que le support de ces couleurs est indépendant du pilote graphique choisi.

Ce support permet d'ailleurs de spécifier des couleurs directement avec leurs paramètres, par exemple avec `\textcolor{cmy}{0.7,0.5,0.3}{toto}` (*toto*) ou `\textcolor[HTML]{AFFE90}{toto}` (*toto*).

`\adjustUCRBG` **rgb, cmyk, hsb, gray** Ce sont les modèles supportés directement par PostScript. C'est pourquoi **✗ nous nous référons ✗** à [1] pour une description de leurs propriétés et relations. Il existe une commande spécifique pour régler finement les mécanismes de **✗ undercolor-removal ✗** et **✗ black-generation/ ✗** durant la conversion vers le modèle **cmyk**, voir section 6.3.2 page 54 pour plus de détails.

cmy Il s'agit principalement d'un modèle pour les étapes de calcul intermédiaire. De ce fait, il s'agit d'un simple complément de **rgb**. En terme d'affichage, **cmy** est traité comme **cmyk** avec $k = 0$.

HTML Ce modèle est dérivé de **rgb** afin de permettre l'entrée de paramètres de couleurs pour les pages web ou les fichiers CSS. Aussi, ce n'est pas un modèle colorimétrique en tant que tel mais plutôt une interface utilisateur commode. Il est

TABLE 1 – Options de l’extension

<i>Option</i>	<i>Description</i>
natural	(valeur par défaut.) Garde toutes les couleurs dans leur modèle, à l’exception de RGB (converti en rgb), HSB (converti en hsb), et Gray (converti en gray).
rgb	Convertit toutes les couleurs en modèle rgb .
cmY	Convertit toutes les couleurs en modèle cmY .
cmYk	Convertit toutes les couleurs en modèle cmYk .
hsb	Convertit toutes les couleurs en modèle hsb .
gray	Convertit toutes les couleurs en modèle gray . Particulièrement utile pour simuler un rendu en noir et blanc d’une imprimante monochrome.
RGB	Convertit toutes les couleurs en modèle RGB (et ensuite en rgb).
HTML	Convertit toutes les couleurs en modèle HTML (et ensuite en rgb).
HSB	Convertit toutes les couleurs en modèle HSB (et ensuite en hsb).
Gray	Convertit toutes les couleurs en modèle Gray (et ensuite en gray).
dvipsnames, dvipsnames*	Charge un ensemble de couleurs prédéfinies. ¹
svgnames, svgnames*	Charge un ensemble de couleurs prédéfinies selon la norme SVG 1.1. ¹
x11names, x11names*	Charge un ensemble de couleurs prédéfinies selon la norme Unix/X11. ¹
table	Charge l’extension colortbl contenant les outils de colorisation des lignes, colonnes et cellules dans des tables.
fixpdftex	Charge l’extension pdfcolmk permettant d’améliorer la gestion des couleurs de pdftex (voir section 2.15.2 page 33).
hyperref	Permet de prendre en charge l’extension hyperref pour les ✕ expressions de couleur ✕ en définissant des ✕ clés ✕ additionnelles (voir section 2.10 page 30).
prologue	Écrit des informations en début de fichier .xcp pour chaque définition de couleur (comme décrit en section 2.5.1 page 20).
kernelFbox	Utilise la méthode du noyau L^AT_EX pour dessiner des boîtes \f(rame)box ² .
xcdraw	Use driver-specific commands to draw frames and color boxes. ²
noxcdraw	(Valeur par défaut.) Utilise un code générique pour dessiner les encadrements et boîtes de couleur. ²
fixinclude	Empêche la réinitialisation de couleur de dvips avant l’inclusion de fichier .eps (voir section 2.15.3 page 33).
showerrors	(Valeur par défaut.) Affiche un message d’erreur si une couleur non définie est utilisée (comportement similaire à celui de l’extension color originale).
hideerrors	Affiche seulement une alerte en cas d’utilisation d’une couleur non définie et remplace cette couleur par du <i>noir</i> .

¹ Voir section 2.4.2 page 19. ² Voir section 2.6.2 page 25.

TABLE 2 – Ordre de chargement des extensions

Chargement/Extension	colortbl	pdfcolmk	hyperref	pstricks	color	pstcol
Avant xcolor	non	non	permis	permis ¹	non	non
Avec l'option xcolor	table	fixpdfetex	—	—	—	—
Après xcolor	non	permis	permis	permis	non	non
¹ Les versions récentes de pstricks chargent xcolor par défaut.						

TABLE 3 – Modèles colorimétriques supportés

Nom	Couleurs de base/notions	Intervalle de valeur	Par défaut
rgb	rouge, vert, bleu	$[0, 1]^3$	
cmy	cyan, magenta, jaune	$[0, 1]^3$	
cmyk	cyan, magenta, jaune, noir	$[0, 1]^4$	
hsb	teinte, saturation, luminosité	$[0, 1]^3$	
Hsb	teinte°, saturation, luminosité	$[0, H] \times [0, 1]^2$	$H = 360$
tHsb	teinte°, saturation, luminosité	$[0, H] \times [0, 1]^2$	$H = 360$
gray	gris	$[0, 1]$	
RGB	Rouge, Vert, Bleu	$\{0, 1, \dots, L\}^3$	$L = 255$
HTML	RRGGBB	$\{000000, \dots, FFFFFFFF\}$	
HSB	Teinte, Saturation, Luminosité	$\{0, 1, \dots, M\}^3$	$M = 240$
Gray	Gris	$\{0, 1, \dots, N\}$	$N = 15$
wave	lambda (nm)	$[363, 814]$	

L, M, N sont des nombres entiers positifs ; H est un entier réel positif

important de mentionner que **HTML** accepte toutes les combinaisons de caractères 0–9, A–F, a–f, tant que la chaîne de caractères a une longueur de 6 caractères exactement. Cependant, les résultats de conversion en **HTML** consisteront en des nombres et des lettres *majuscules*.

Hsb, tHsb Premièrement, **Hsb** est un modèle « interface utilisateur » transformant $teinte \in [0, 1]$ en $teinte^\circ \in [0, H]$, où H est donné par `\def\rangeHsb{⟨H⟩}`.

`\rangeHsb` Aussi, si $H = 360$, nous pouvons penser à un cercle ou une roue pour **✕ décrire** **✕** le paramètre $teinte^\circ$.

Deuxièmement, **Hsb** est la base du **tHsb**, également nommé **Hsb réglé**, qui permet à l'utilisateur d'appliquer une transformation linéaire **✕ piecewise ✕** sur $teinte^\circ$ en déplaçant la $teinte^\circ$ sélectionnée en avant ou en arrière sur le cercle.

`\rangetHsb` La transformation est définie par `\def\rangetHsb{x_1, y_1; x_2, y_2; \dots}` qui indique que $hue^\circ = x_1$ dans **tHsb** signifie $hue^\circ = y_1$ dans **Hsb**, etc. Par exemple, le *jaune*

est placé à 60° dans le cercle **Hsb** (le rouge étant à 0°) ; cependant dans la plus plupart des cercles chromatiques servant aux artistes, le *jaune* est à 120° . Ainsi, une entrée « 120,60 » ferait sens si nous avions décidé de répliquer un cercle chromatique d'artiste par le biais de **tHsb**. Voir la section 6.3.6 page 58 pour la formule exacte de la transformation et les restrictions avancées, et la section 1.4 page 7 pour les cercles chromatiques et les accords de couleur. La figure 11 page 40 peut servir pour effectuer des comparaisons.

Exemple : `\def\rangetHsb{60,30;120,60;180,120;210,180;240,240}` correspond en fait au paramétrage par défaut de xcolor.

wave Avec ce modèle nous essayons de transformer les longueurs d'onde en un modèle de colorimétrie standard afin de réaliser une approximation de l'apparence visuelle des ondes lumineuses. Tandis que le spectre visible couvre un intervalle de valeur de 400 à 750 nm, l'implémentation dans xcolor permet de traiter toutes les longueurs d'onde qui ont une valeur absolue inférieure à 16383.99998 (le plus grand nombre que T_EX puisse considérer comme une dimension). Toutefois, la probabilité d'obtenir une couleur différente du noir hors de plage de valeur [363,814] est très précisément nulle. Aussi, la figure 1 page 35 illustre seulement l'intervalle de valeur mention ci-dessus. Notez qu'il n'est pas possible de convertir fidèlement les autres modèles en **wave** puisque ce dernier ne couvre qu'un ensemble limité de couleurs.

RGB, HSB, Gray Ce sont des modèles dérivés, transformant la plage de valeurs continue $[0,1]$ des paramètres de **rgb**, **hsb** et **gray** en un ensemble de valeurs finies ; ce qui nous fait les désigner par le terme de *modèles entiers*. Les constantes L, M, N de la table 3 sont définies par les commandes

`\rangeRGB` `\def\rangeRGB{<L>}`, `\rangeHSB` `\def\rangeHSB{<M>}`, and `\rangeGray` `\def\rangeGray{<N>}`. La modification de ces constantes peut être fait *avant* ou *après* que l'extension xcolor ait été chargée, par exemple :

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

2.2.2 Substitution de modèles colorimétriques individuels

`\substitutecolormodel` `{<modèle source>}{<liste de modèles cibles>}`

Substitue le *⟨modèle source⟩* par le premier modèle disponible apparaissant dans la *⟨liste de modèles cibles⟩*. Seuls les modèles de type *⟨modèle numérique⟩* sont possibles ; tous les changements sont locaux au groupe courant, mais un `\xglobal` préfixé est respecté.

Exemple : supposons que le pilote actuel a une implémentation incorrecte de **hsb** tandis que **rgb** paraît correct. Alors `\substitutecolormodel{hsb}{rgb}` pourrait être un bon choix puisqu’il convertit — à partir de ce point — toutes les définitions des couleurs **hsb** en spécifications du modèle **rgb** par le biais des algorithmes de `xcolor`, sans toucher aux autres modèles.

2.2.3 Changement du modèle colorimétrique cible dans un document

$$\backslash\mathrm{selectcolormodel} \quad \{\langle num\ model\rangle\}$$

Définit le modèle cible au *⟨modèle numérique⟩*, où ce dernier est un des noms de modèles autorisés comme option de l'extension (autrement dit, **natural**, **rgb**, **cm**, **cm**, **cm**, **gray**, **RGB**, **HTML**, **HSB**, **Gray**), voir figure 4 page 37 pour un exemple. Il y a deux possibilités pour rendre possible la conversion au modèle cible :

`\ifconvertcolorsD` — au moment de la *définition* des couleurs³ (autrement dit dans `\definecolor` et ses assimilées) ; ceci est contrôlé par la bascule `\ifconvertcolorsD` ;

`\ifconvertcolorsU` — au moment de l'*utilisation* des couleurs (immédiatement avant que la couleur soit affichée, ce qui traite qui ont été définies dans d'autres modèles ou qui ont été définies directement comme avec `\color[rgb]{.1,.2,.3}`); ceci est contrôlé par la bascule `\ifconvertcolorsU`.

Les deux bascules valent « vrai » en sélectionnant n'importe quel modèle, à l'exception de `natural` qui leur donne la valeur « faux ». Ceci ✖ s'applique ✖ à une sélection par le biais d'une option d'extension comme par le biais de `\selectcolormodel`. Pourquoi ne convertissons-nous pas toutes les couleurs au moment de l'utilisation ? Si de nombreuses couleurs sont impliquées, cela peut économiser du temps de traitement lorsque les conversions sont déjà faites au moment des définitions. De meilleures performances peuvent être obtenues par `\usepackage[rgb,...]{xcolor}\convertcolorsUfalse`, ce qui est en fait la façon dont `xcolor` fonctionnait jusqu'à la version 1.07.

2.3 Arguments et terminologie

Avant de décrire en détail les commandes liées aux couleurs de `xcolor`, nous définissons plusieurs éléments ou identifiants qui apparaissent de façon répétée dans les arguments de ces commandes. Une vue générale de la syntaxe est donnée dans la table 4 page suivante.

2.3.1 Remarques additionnelles et restrictions sur les arguments

Chaînes basiques et nombres Ces arguments ne nécessitent pas beaucoup
 3. Ceci signifie que toute couleur *nouvellement* définie sera d’abord convertie dans le modèle
 cible, puis sauvegardée.

TABLE 4 – Arguments et terminologie

Élément	Chaîne de remplacement	
$\langle vide \rangle$	→ chaîne vide ‘ ’	
$\langle moins \rangle$	→ chaîne non vide contenant un ou plusieurs signes ‘-’	
$\langle plus \rangle$	→ chaîne non vide contenant un ou plusieurs signes ‘+’	
$\langle ent \rangle$	→ nombre entier	(entier)
$\langle num \rangle$	→ nombre entier positif	(nombre)
$\langle déc \rangle$	→ nombre réel	(décimal)
$\langle div \rangle$	→ nombre réel non nul	(diviseur)
$\langle pct \rangle$	→ nombre réel dans l’intervalle [0, 100]	(pourcentage)
$\langle id \rangle$	→ chaîne non vide contenant des lettres et des chiffres	(identifiant)
$\langle id \text{ étendu} \rangle$	→ $\langle id \rangle$ → $\langle id \rangle_1 = \langle id \rangle_2$	
$\langle liste-id \rangle$	→ $\langle id \text{ étendu} \rangle_1, \langle id \text{ étendu} \rangle_2, \dots, \langle id \text{ étendu} \rangle_l$	
$\langle nom \rangle$	→ $\langle id \rangle$ → ‘.’	(nom explicite) (nom implicite)
$\langle \text{modèle central} \rangle$	→ ‘rgb’, ‘cmY’, ‘cmyk’, ‘hsb’, ‘gray’	(modèles centraux)
$\langle \text{modèle numérique} \rangle$	→ $\langle \text{modèle central} \rangle$ → ‘RGB’, ‘HTML’, ‘HSB’, ‘Gray’ → ‘Hsb’, ‘tHsb’, ‘wave’	(modèles entiers) (modèles décimaux)
$\langle \text{modèle} \rangle$	→ $\langle \text{modèle num} \rangle$ → ‘named’	(modèles numériques) (pseudo-modèle)
$\langle \text{liste-modèle} \rangle$	→ $\langle \text{modèle} \rangle_1 / \langle \text{modèle} \rangle_2 / \dots / \langle \text{modèle} \rangle_m$ → $\langle \text{modèle central} \rangle : \langle \text{modèle} \rangle_1 / \langle \text{modèle} \rangle_2 / \dots / \langle \text{modèle} \rangle_m$	(modèles multiples)
$\langle \text{spéc} \rangle$	→ liste de valeurs numériques séparées par des virgules → liste de valeurs numériques séparées par des virgules → nom d’une couleur « nommée »	(spécification explicite) (spécification explicite) (spécification implicite)
$\langle \text{liste-spéc} \rangle$	→ $\langle \text{spéc} \rangle_1 / \langle \text{spéc} \rangle_2 / \dots / \langle \text{spéc} \rangle_m$	(spécifications multiples)
$\langle \text{type} \rangle$	→ $\langle vide \rangle$ → ‘named’, ‘ps’	
$\langle \text{expr} \rangle$	→ $\langle \text{préfixe} \rangle \langle \text{nom} \rangle \langle \text{expr de mélange} \rangle \langle \text{suffixe} \rangle$	(expression de couleur standard)
$\langle \text{préfixe} \rangle$	→ $\langle vide \rangle$ → $\langle moins \rangle$	(indicateur complémentaire)
$\langle \text{expr de mélange} \rangle$	→ $! \langle pct \rangle_1 ! \langle nom \rangle_1 ! \langle pct \rangle_2 ! \langle nom \rangle_2 ! \dots ! \langle pct \rangle_n ! \langle nom \rangle_n$ → $! \langle pct \rangle_1 ! \langle nom \rangle_1 ! \langle pct \rangle_2 ! \langle nom \rangle_2 ! \dots ! \langle pct \rangle_n$	(expression de mélange complète) (expression de mélange incomplète)
$\langle \text{suffixe} \rangle$	→ $\langle vide \rangle$ → $!! \langle plus \rangle$ → $!! [\langle num \rangle]$	(✖ series step ✖) (✖ series access ✖)
$\langle \text{expr étendue} \rangle$	→ $\langle \text{modèle central} \rangle, \langle div \rangle : \langle \text{expr} \rangle_1, \langle \text{déc} \rangle_1 ; \langle \text{expr} \rangle_2, \langle \text{déc} \rangle_2 ; \dots ; \langle \text{expr} \rangle_k, \langle \text{déc} \rangle_k$ → $\langle \text{modèle central} \rangle : \langle \text{expr} \rangle_1, \langle \text{déc} \rangle_1 ; \langle \text{expr} \rangle_2, \langle \text{déc} \rangle_2 ; \dots ; \langle \text{expr} \rangle_k, \langle \text{déc} \rangle_k$	
$\langle \text{expr fonctionnelle} \rangle$	→ $> \langle \text{fonction} \rangle, \langle \text{arg} \rangle_1, \langle \text{arg} \rangle_2, \dots, \langle \text{arg} \rangle_j$	(expression fonctionnelle de couleur)
$\langle \text{fonction} \rangle$	→ ‘wheel’, ‘twheel’	(fonctions de couleur)
$\langle \text{couleur} \rangle$	→ $\langle \text{expr de couleur} \rangle \langle \text{expr fonctionnelle} \rangle_1 \langle \text{expr fonctionnelle} \rangle_2 \dots \langle \text{expr fonctionnelle} \rangle_i$	
$\langle \text{expr de couleur} \rangle$	→ $\langle nom \rangle$ → $\langle \text{expr} \rangle$ → $\langle \text{expr étendue} \rangle$	
Remarques :	chaque → indique une chaîne de remplacement possible pour un élément de la colonne de gauche; cependant, des restrictions avancées dépendantes du contexte peuvent s’appliquer. Voir le texte principal pour plus de détails. La chaîne ‘toto’ doit toujours être comprise sans les apostrophes. i, j, k, l, m, n indiquent des entiers positifs, $k, l, m, n > 0, m \leq 8$.	

d'explications. Cependant, dans la mesure où nous traitons ici des valeurs numériques, il est important de noter que les nombres réels dans (La)TeX — tant qu'ils sont utilisés pour des longueurs, dimensions ou espaces — sont limités à une valeur maximale inférieure strictement à 16384. Cette contrainte, dans l'enchaînement des calculs numériques, doit aussi être respectée par tous les résultats intermédiaires, ce qui implique généralement des restrictions plus larges. Comme xcolor utilise énormément les registres internes de dimension de TeX pour la plupart des calculs, ce point doit être gardé à l'esprit à chaque fois que des expressions *<expr étendue>* doivent être utilisées.

<nom> **Noms de couleur** Un *<nom>* indique le nom déclaré (ou le nom qui va être déclaré) d'une *couleur* ou d'une **✕ série de couleur ✕**; il peut être déclaré *explicitement* par l'une des commandes suivantes : `\definecolor`, `\providecolor`, `\colorlet`, `\definecolorset`, `\providecolorset`, `\definecolorseries`, `\definecolors`, `\providecolors`. Par ailleurs, le nom de couleur réservé `'.'` est déclaré *implicitement* et indique la *couleur actuelle*. En fait, au-delà des chiffres et lettres, certains autres caractères peuvent également être utilisés pour les déclarations de *<nom>* mais les restrictions données évitent les incompréhensions et garantissent la compatibilité avec les futures évolutions de xcolor.
Exemples : `'red'`, `'MonVertSpecial1980'`, `'.'`.

<modèle central>
<modèle numérique>
<modèle> **Modèles colorimétriques**
La différence faite entre les *modèles centraux* (**rgb**, **cmy**, **cmyk**, **hsb**, **gray**), les *modèles entiers* (**RGB**, **HTML**, **HSB**, **Gray**), les *modèles décimaux* (**Hsb**, **tHsb**, **wave**) et les *pseudo-modèles* (actuellement `'named'`, `'ps'`) s'explique simplement : les modèles centraux avec leurs paramètres basés sur l'intervalle unité $[0, 1]$ permettent de faire plus aisément tout type de calculs, tandis que le but des modèles entiers est principalement de faciliter la saisie des paramètres en entrée (transformés ensuite en ceux d'un des modèles centraux). Enfin, les modèles décimaux **Hsb** et **tHsb** sont des versions de **hsb** pensés pour des buts spécifiques, tandis que **wave** et le pseudo-modèle `'named'` ont un statut spécial dans la mesure où ils ne sont pas pensés pour des calculs : s'il est normalement possible de convertir une couleur de ces modèles en une d'un autre modèle, l'inverse ne l'est pas. La situation est bien pire pour le pseudo-modèle `'ps'` : ces couleurs contenant du code PostScript **✕ ne sont pas transparentes ✕** pour TeX.

<spéc> **Spécifications de couleur** L'argument *<spéc>* — qui spécifie les paramètres d'une couleur — dépend évidemment du modèle colorimétrique sous-jacent. Une différence est faite entre les spécifications *explicite* et *implicite*, la première faisant référence à des paramètres numériques comme expliqué en table 3 page 11, la seconde — idéalement — faisant référence à des noms définis par le pilote graphique. Exemples : `' .1, .2, .3'`, `' .1 .2 .3'`, `'0.56789'`, `'89ABCD'`, `'ForestGreen'`.

<liste-modèle>
<liste-spéc> **Modèles et spécifications multiples** Ces arguments apparaissent toujours par paires (explicites ou implicites) dans les commandes de définition de couleur

suivantes : `\definecolor`, `\providecolor`, `\definecolorset`, `\providecolorset`.
 Tout d'abord, $\langle \text{modèle-spéc} \rangle$ est réconcilié avec le modèle cible courant (fixé par exemple avec une option de l'extension ou la commande `\selectcolormodel` ; dans le cas où il n'existe de modèle correspondant, le premier modèle de la liste est choisi. Ensuite, la spécification de couleur correspondante sera sélectionnée dans $\langle \text{liste-spéc} \rangle$, de telle façon à ce que le traitement aboutisse à une paire ($\langle \text{modèle} \rangle$, $\langle \text{spéc} \rangle$) cohérente. Ceci explique pourquoi il n'y a plus d'ambiguïté possible dans la définition de couleur réellement suivie. La forme étendue $\langle \text{modèle central} \rangle : \langle \text{modèle} \rangle_1 / \langle \text{modèle} \rangle_2 / \dots / \langle \text{modèle} \rangle_m$ provoque la conversion immédiate de la $\langle \text{spéc} \rangle$ adéquate au $\langle \text{modèle central} \rangle$; un modèle inconnu sera tout simplement ignoré ici, sans aucun commentaire.

Exemples : `'rgb/cmyk/named/gray'`, `'0,0,0/0,0,0,1/Black/0'`, `'rgb:cmy/hsb'`.

$\langle \text{type} \rangle$ **L'argument de type** Ceci est utilisé uniquement dans le contexte de commandes de définition de couleur, voir la description de `\definecolor` et assimilées.

$\langle \text{expr} \rangle$
 $\langle \text{préfixe} \rangle$
 $\langle \text{expr de mélange} \rangle$
 $\langle \text{suffixe} \rangle$

Expressions standards de couleur Ces expressions servent d'outils pour spécifier facilement une certaine forme de mélange de couleur en cascade, par ailleurs décrit en détail en section 2.3.2. L'argument $\langle \text{préfixe} \rangle$ détermine si la couleur à retenir est celle qui suit ou sa complémentaire : un nombre impair de signes négatifs indique que la couleur résultant de l'expression préfixée doit être convertie en sa couleur complémentaire. Une *expression de mélange incomplète* est une juste une abbréviation d'une *expression de mélange complète* avec $\langle \text{nom} \rangle_n = \text{white}$, afin d'éviter quelques saisies dans le cas des teintes. La chaîne $\langle \text{suffixe} \rangle$ est généralement vide mais elle offre quelques fonctionnalités additionnelles dans le cas de **✖ color series ✖** : les cas où la chaîne n'est pas vide demandent à ce que

- le $\langle \text{nom} \rangle$ indique le nom d'une **✖ color series ✖** ;
- l' $\langle \text{expr de mélange} \rangle$ est *complète*.

Exemples : `'red'`, `'-red'`, `'--red!50!green!12.345'`, `'red!50!green!20!blue'`, `'truc!!'`, `'truc!![7]'`, `'truc!25!red!!++'`, `'truc!25!red!70!green!![7]'`.

$\langle \text{expr étendue} \rangle$ **Expressions de couleur étendues** Ces expressions fournissent une autre méthode pour mélanger des couleurs, voir section 2.3.3 page 18 pour plus d'informations. La forme raccourcie

$$\langle \text{modèle central} \rangle : \langle \text{expr} \rangle_1, \langle \text{déc} \rangle_1 ; \langle \text{expr} \rangle_2, \langle \text{déc} \rangle_2 ; \dots ; \langle \text{expr} \rangle_k ! \langle \text{déc} \rangle_k$$

est une abbréviation pour le cas spécial (et probablement plus courant)

$$\langle \text{modèle central} \rangle, \langle \text{div} \rangle : \langle \text{expr} \rangle_1, \langle \text{déc} \rangle_1 ; \langle \text{expr} \rangle_2, \langle \text{déc} \rangle_2 ; \dots ; \langle \text{expr} \rangle_k ! \langle \text{déc} \rangle_k$$

avec la définition suivante (impliquant une somme non nulle de tous les coefficients $\langle \text{déc} \rangle_k$) :

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

Exemples : `'rgb:red,1'`, `'cmyk:red,1;-green!25!blue!60,11.25;blue,-2'`.

$\langle \text{expr fonctionnelle} \rangle$ $\langle \text{fonction} \rangle$	Expressions fonctionnelles Ces expressions étendent les fonctionnalités des expressions <i>standards</i> ou <i>étendues</i> en récupérant le résultat de ces expressions pour effectuer des calculs complémentaires. Le nombre d'arguments peut varier entre les différentes fonctions, voir section 2.3.4 page suivante pour plus d'informations. Exemples : <code>>wheel,30</code> , <code>>wheel,30,'</code> , <code>>twheel,1,12</code> , <code>>twheel,-11,12</code> .
$\langle \text{couleur} \rangle$ $\langle \text{expr de couleur} \rangle$	Couleurs Au final, $\langle \text{couleur} \rangle$ est un argument générique recouvrant les différents concepts de spécification des couleurs. Ceci signifie qu'à chaque fois qu'un argument $\langle \text{couleur} \rangle$ est utilisable, la totalité des noms et expressions vues ci-dessus peuvent être utilisées.

2.3.2 Signification des expressions de couleur standards

Est expliquée maintenant comme l'expression

$$\langle \text{préfixe} \rangle \langle \text{name} \rangle ! \langle \text{pct} \rangle_1 ! \langle \text{name} \rangle_1 ! \langle \text{pct} \rangle_2 ! \dots ! \langle \text{pct} \rangle_n ! \langle \text{name} \rangle_n \langle \text{suffixe} \rangle$$

est interprétée et traitée :

1. Tout d'abord, le modèle et les paramètres de couleur de $\langle \text{nom} \rangle$ sont extraits pour définir une couleur temporaire $\langle \text{temp} \rangle$. Si $\langle \text{suffixe} \rangle$ est de la forme `'!![<num>]'`, alors $\langle \text{temp} \rangle$ sera la couleur correspondante $\langle \text{num} \rangle$ (en accès direct) de la série de couleur $\langle \text{nom} \rangle$.
2. Alors un mélange de couleur, consistant en $\langle \text{pct} \rangle_1\%$ de la couleur $\langle \text{temp} \rangle$ et $(100 - \langle \text{pct} \rangle_1)\%$ de la couleur $\langle \text{nom} \rangle_1$ est calculé; ce dernier devient la nouvelle couleur temporaire $\langle \text{temp} \rangle$.
3. L'étape précédente est répétée pour toutes les paires de paramètres restantes. ($\langle \text{pct} \rangle_2, \langle \text{nom} \rangle_2$), ..., ($\langle \text{pct} \rangle_n, \langle \text{nom} \rangle_n$).
4. Si $\langle \text{préfixe} \rangle$ contient un nombre impair de signes négatifs '-', alors $\langle \text{temp} \rangle$ sera changée en sa couleur complémentaire.
5. Si $\langle \text{suffixe} \rangle$ est de la forme `'!!+', '!!++', '!!+++', etc.` un nombre de **✖ step commands** ✖ (= nombre de signes '+') sont effectuées sur la série de couleur sous-jacente $\langle \text{nom} \rangle$. Ceci est sans conséquence pour la couleur $\langle \text{temp} \rangle$.
6. La couleur $\langle \text{temp} \rangle$ est enfin affichée ou sert comme donnée en entrée pour d'autres opérations, selon la commande utilisée.

Notez que, dans une expression $\langle \text{temp} \rangle ! \langle \text{pct} \rangle_\nu ! \langle \text{nom} \rangle_\nu$ typique de l'étape 2, si $\langle \text{pct} \rangle_\nu = 100$, la couleur $\langle \text{temp} \rangle$ est directement utilisée sans plus de transformation. Si $\langle \text{pct} \rangle_\nu = 0$, c'est alors la couleur $\langle \text{nom} \rangle_\nu$ qui est utilisée. Dans les cas de véritables mélanges ($0 < \langle \text{pct} \rangle_\nu < 100$), les deux couleurs impliquées peuvent être définies dans des modèles colorimétriques différents, par exemple `\definecolor{foo}{rgb}{...}` et `\definecolor{bar}{cmyk}{...}`. En général, la seconde couleur, $\langle \text{nom} \rangle_\nu$, est convertie dans le modèle de la première couleur, $\langle \text{temp} \rangle$, puis le mélange est calculé dans le modèle ⁴ Ainsi, $\langle \text{temp} \rangle ! \langle \text{pct} \rangle_\nu ! \langle \text{nom} \rangle_\nu$ et

4. Exception : afin d'éviter des résultats inattendus, cette règle est inversée si $\langle \text{temp} \rangle$ est issue du modèle **gray**; dans ce cas, elle est convertie dans le modèle associé à $\langle \text{nom} \rangle_\nu$.

$\langle nom \rangle_\nu ! \langle 100 - pct \rangle_\nu ! \langle temp \rangle$ qui devraient être théoriquement équivalents, peuvent ne pas avoir des résultats visuels identiques.

Les figures 5 à 6 page 37 montrent de premières applications des couleurs et expressions. D'autres exemples sont donnés en figure 3 page 36. Par ailleurs, un grand nombre d'exemples peuvent être trouvés dans [9].

2.3.3 Signification des expressions de couleur étendues






Une *expression de couleur étendue*

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


imite la manière dont les peintres mélangent les couleurs : en indiquant une liste de couleurs, chaque couleur étant associée à un facteur $\langle dec \rangle$. Dans une telle *expr étendue*, chaque expression standard de couleur $\langle expr \rangle_\kappa$ sera convertie dans le $\langle modèle\ central \rangle$ et le vecteur résultant est multiplié par $\langle dec \rangle_\kappa / \langle div \rangle$, où

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

Ensuite, la somme de tous ces vecteurs est calculée.

Exemple : mélanger 4 parts de  *red*, 2 parts de  *green*, et une part de  *yellow* permet d'obtenir  par le biais de `\color{rgb:red,4;green,2;yellow,1}`. Essayer le même mélange en mettant -1 part de *yellow* au lieu d'une fait obtenir . Notez que ce mécanisme peut être aussi utilisé pour afficher une (expression de) couleur individuelle dans un certain modèle colorimétrique : `\color{rgb:yellow,1}` permet une telle conversion. La forme générale

$$\langle modèle\ central \rangle, \langle div \rangle : \langle expr \rangle_1, \langle dec \rangle_1 ; \langle expr \rangle_2, \langle dec \rangle_2 ; \dots ; \langle expr \rangle_k, \langle dec \rangle_k$$

exécute la même opération avec pour seule différence que le diviseur $\langle div \rangle$ est spécifié au lieu d'être calculé. Dans l'exemple ci-dessus, nous obtenons une version plus sombre  par le biais de `\color{rgb,9:red,4;green,2;yellow,1}`. Notez qu'il n'est pas interdit de spécifier un argument $\langle div \rangle$ qui soit plus petit que la somme de tous les $\langle dec \rangle_\kappa$, de telle façon à ce que certains des paramètres de spécification des couleurs puissent être hors de l'intervalle $[0, 1]$. Le traitement de l'équation 7 gère ce type de cas.

2.3.4 Fonctions de couleur

Les fonctions de couleur utilisent une liste d'arguments séparés par des virgules et elles servent à transformer la *couleur donnée* (autrement dit le résultat des calculs précédant l'appel de la fonction) en une nouvelle couleur.

wheel **Calculs associés aux cercles chromatiques** Ces fonctions permettent de déterminer des couleurs liées par des relations harmoniques basées sur les cercles chromatiques (voir section 1.4 page 7). Les arguments sont ici $\langle angle \rangle$ ou $\langle angle \rangle, \langle cercle\ complet \rangle$, le premier servant d'abréviation à $\langle angle \rangle, \backslash rangeHsb$.




















Le second argument (*cercle complet*) indique de combien d'unités un cercle complet est constitué tandis que le premier argument indique de combien d'unités doit être faite la rotation à appliquer à la couleur donnée. Pour cela, la couleur donnée est tout d'abord convertie en **Hsb** (dans le cas de *wheel*), ce qui génère les paramètres *teinte*[°], *saturation*, et *luminosité*. Ensuite,

$$teinte^{\circ} := teinte^{\circ} + \frac{\langle angle \rangle}{\langle cercle\ complet \rangle} \cdot H, \quad teinte := u\left(\frac{teinte^{\circ}}{H}\right) \quad (4)$$

où u est la fonction de réduction d'intervalle de l'équation 7 et $H = \backslash rangeHsb$. La *saturation* et la *luminosité* étant laissées inchangées, le modèle final est **hsb**. La fonction *twheel* fonctionne de façon similaire, mais ces arguments se basent sur **tHsb** au lieu de **Hsb**. Des exemples sont présentés en figure 12 page 41.

2.4 Couleurs prédéfinies

2.4.1 Couleurs qui sont toujours disponibles

Dans le fichier `xcolor.sty`, les noms de couleur suivants sont définis :  *red*,  *green*,  *blue*,  *cyan*,  *magenta*,  *yellow*,  *black*,  *gray*,  *white*,  *darkgray*,  *lightgray*,  *brown*,  *lime*,  *olive*,  *orange*,  *pink*,  *purple*,  *teal*,  *violet*.

Cet ensemble de base de couleurs peut être utilisé sans aucune restriction dans tout type d'expression de couleur, comme expliqué en section 2.3 page 13.

2.4.2 Ensembles additionnels de couleurs

Il existe également des ensembles de noms de couleur qui peuvent être chargés par le biais d'options d'extension, toujours disponibles en deux variantes : une version « normale » (par exemple, `dvipsnames`) et une version « étoilée » (par exemple, `dvipsnames*`). La première variante définit toutes les couleurs *immédiatement*, tandis que la seconde applique le mécanisme de la définition *différée*. Dans ce dernier cas, les noms de couleur individuels doivent être activés par les commandes `\definecolors` ou `\providecolors`, comme décrit dans la section 2.5.4 page 23, avant de pouvoir être appliqués dans un document.

- `dvipsnames/dvipsnames*` charge un ensemble de 68 noms de couleur **cmky** telles que définies par le pilote `dvips`. Cependant, ces couleurs peuvent être utilisées avec tous les pilotes supportés.
- `svgnames/svgnames*` charge un ensemble de 151 noms de couleur⁵ **rgb** respectant la spécification SVG 1.1 [17]⁶, augmenté de 4 noms tirés du fichier `rgb.txt` appartenant aux distributions Unix/X11. Notez que HTML4 accepte un sous-ensemble de 16 noms de couleur (en utilisant des spécifications identiques), voir [16] et la section 4 page 42.

5. En fait, les noms chargés représentent 141 couleurs différentes.

6. Plus exactement, la spécification indiquée liste uniquement les noms écrits en minuscules. De plus, les définitions originales sont données en paramètres **RGB** et ont été converties en **rgb** par l'auteur.

- `x11names/x11names*` charge un ensemble de 317 noms de couleur⁷ **rgb** qui sont une simple variation sur un sous-ensemble de l'ensemble SVG mentionné précédemment, respectant le fichier `rgb.txt` appartenant aux distributions Unix/X11⁸. Pour obtenir les 752 noms de couleur de `rgb.txt` sans trop d'effort :
 - chargez `x11names` ainsi que `svgnames` ;
 - mettez les initiales en majuscule et supprimez les espaces : *DarkSlate-Gray* au lieu de *dark slate gray* par exemple ;
 - les noms X11 sans les nombres sont identiques aux couleurs SVG sauf dans 5 cas : utilisez *Gray0*, *Grey0*, *Green0*, *Maroon0*, *Purple0* au lieu de *Gray*, *Grey*, *Green*, *Maroon*, *Purple* pour obtenir les couleurs X11 d'origine ;
 - pour $N = 0, 1, \dots, 100$, utilisez '`[gray]{N/100}`' ou '`black!100 - N`' au lieu de *grayN* ou *greyN*.

Les noms des couleurs ainsi que leur visualisation sont présentés en section 4 page 42. La section 2.15.1 page 32 décrit comment traiter les doublons de noms lors de l'utilisation conjointe de `svgnames` et `dvipsnames` dans un même document. Voir également [9] avec son ensemble systématique de couleurs et des exemples de mélange.

2.5 Définition de couleur

2.5.1 Couleurs ordinaires et nommées

Dans l'extension `color` il y a une distinction entre les « couleurs » (définies par la commande `\definecolor`) et les « couleurs nommées » (définies par `\DefineNamedColor`, ce qui est autorisé uniquement dans le préambule). Chaque fois qu'une couleur ordinaire est utilisée dans un document, elle est déposée dans une commande `\special` qui contient une description numérique de la couleur — dépendante du pilote — et qui est écrite dans le fichier `.dvi`. Les couleurs nommées, elles, présentent l'opportunité de stocker les valeurs numériques à une place centrale tandis que, pendant leur utilisation, les couleurs peuvent être identifiées par leur nom, ce qui permet des traitements ultérieurs si nécessaire par le périphérique de sortie.

Tous les pilotes livrés avec l'extension standard `graphics` supportent le *formalisme* de la définition et de l'appel de « couleurs nommées ». Cependant, le support réel du *concept* derrière cela, autrement dit employer des noms au lieu des paramètres, va d'« inexistant » à « total ». Voici une illustration de la situation actuelle avec trois pilotes différents.

- `dvips` traite très bien le concept de « couleur nommée » ; les équivalents PostScript des noms de couleur définis par `dvipsnames` sont chargés — à moins qu'ils ne soient désactivés — automatiquement par `dvips`. Cependant les noms additionnels doivent être défini à l'interpréteur PostScript par une

7. Ces noms représentent 315 couleurs différentes.

8. Une nouvelle fois, les définitions originales sont données en paramètres **RGB** et ont été converties en **rgb** par l'auteur.

sorte de fichier de préambule. Depuis la version 2.01, xcolor propose une solution intégrée pour effectuer cette tâche : en utilisant l'option d'extension **prologue**, un fichier de préambule PostScript **xcolor.pro** est chargé par **dvips**. En complément, avec cette option, chaque commande de définition de couleur⁹ (**\definecolor**, **\colorlet**, etc.) génère un code PostScript enregistré dans un fichier auxiliaire d'extension **.xcp** (abréviation de « **xcolor prologue** »). Ce fichier est également chargé par **dvips** comme préambule, rendant ainsi disponibles tous les noms de couleur à l'interpréteur PostScript. Bien entendu, le fichier **.xcp** peut être édité avant que **dvips** ne l'utilise, ce qui permet de changer les paramètres de couleur spécifiques au pilote à un endroit central. Notez que le code PostScript est constitué de façon similaire à **color.pro** : seuls les *nouveaux* noms sont définis. Ceci permet de précharger d'autres fichiers de préambule avec des définitions de couleur qui ne sont pas détruites par xcolor. En contrepartie, ceci impose à l'utilisateur de prendre soin de la redéfinition des noms de couleur.

Exemple : **\colorlet{foo}{red}\colorlet{foo}{blue}\color{foo}** va basculer la couleur à *blue* dans la logique usuelle de xcolor, bien que le fichier **.ps** va afficher *red* (à moins que *foo* n'ait été défini différemment avant).

Il faut souligner que ce mécanisme est employé uniquement avec l'option **prologue**. Sans cela, les couleurs « nommées » prédéfinies activées par l'option **dvipsnames** (sans employer aucune teinte, nuance, expression de couleur, etc.) peuvent être utilisées de cette manière, toutes les autres couleurs « nommées » sont inconnues de PostScript.

- **dvipdfm** supporte seulement les couleurs **dvipsnames** standard car elles sont décrites dans le programme **dvipdfm** lui-même ; il ne semble pas y avoir de façon de charger un fichier de préambule défini par l'utilisateur.
- **pdftex** ne permet pas le support conceptuel, toutes les couleurs « nommées » étant converties immédiatement en leur représentation numérique. En conséquence, ceci permet d'utiliser des définitions et un usage des couleurs nommées sans restriction (même si cela n'offre aucune valeur ajoutée ici).

Typiquement, un visualisateur **.dvi** aura des difficultés à afficher les couleurs « nommées » définies par l'utilisateur. Par exemple, le visualisateur de MiKTeX, **Yap**, affiche actuellement uniquement les couleurs « nommées » de l'ensemble **dvipsnames**. Aussi, à chaque fois que l'option **prologue** est utilisée en lien avec l'option **dvips**, *toutes* les autres couleurs sont restituées en noir. Cependant, après le traitement par **dvips**, un visualisateur PostScript affichera les bonnes couleurs.

2.5.2 Définition de couleur dans xcolor

\definecolor [*type*]{*nom*}{*liste-modèle*}{*liste-spéc*}¹⁰

9. Ceci est vrai non seulement pour le préambule du document mais aussi pour tout le corps du document.

Il s'agit d'une des commandes qui peut être utilisée pour assigner un $\langle nom \rangle$ à une couleur spécifique. La couleur est ensuite connue du système (dans le groupe où elle est définie) et peut être utilisée dans toute *expression de couleur*, comme expliquée en section 2.3 page 13. La commande remplace à la fois `\DefineNamedColor` et `\definecolor` de `color`. Notez que la définition d'un $\langle nom \rangle$ de couleur existant déjà est écrasée. La variable `\tracingcolors` contrôle si cet écrasement est indiqué dans le fichier « log » ou pas (voir section 2.13 page 32 pour plus d'informations). Les arguments sont décrits en section 2.3 page 13. Aussi, des expressions valides définissant des couleurs sont, par exemple :

```
— \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},
```

où la dernière commande est équivalente à `\colorlet{myblack}{Black}` (voir ci-dessous) ; la deuxième commande définit *red* dans le modèle **rgb** ou **cmyk** selon la paramètre actuel du *modèle cible*, tandis que la troisième convertit la couleur au modèle **hsb** avant d'être enregistré. Notez qu'il existe une version spéciale associée à `pstricks`, comme décrit en section 2.11 page 31.

`\providecolor` [$\langle type \rangle$]{ $\langle nom \rangle$ }{ $\langle liste-modèle \rangle$ }{ $\langle liste-spéc \rangle$ }
 Cette commande est similaire à `\definecolor` à ceci près que la couleur $\langle nom \rangle$ est définie seulement si elle n'existe pas déjà.

`\colorlet` [$\langle type \rangle$]{ $\langle nom \rangle$ }[$\langle modèle numérique \rangle$]{ $\langle couleur \rangle$ }
 Cette commande copie la couleur obtenue avec $\langle couleur \rangle$ dans $\langle name \rangle$. Si $\langle modèle numérique \rangle$ n'est pas vide, $\langle couleur \rangle$ est tout d'abord converti dans le modèle spécifié avant que $\langle name \rangle$ ne soit défini. Le pseudo-modèle 'named' n'est pas autorisé ici mais il peut cependant être spécifié dans l'argument $\langle type \rangle$. Notez qu'une couleur nommée $\langle nom \rangle$ définie auparavant sera écrasée.
 Exemple : `\colorlet{tableheadcolor}{gray!25}` a été utilisé dans le préambule du document. Dans la plupart des tables, la première ligne est mise en forme en utilisant la commande `\rowcolor{tableheadcolor}`.

2.5.3 Définition d'ensembles de couleur

`\definecolorset` [$\langle type \rangle$]{ $\langle liste-modèle \rangle$ }{ $\langle tête \rangle$ }{ $\langle queue \rangle$ }{ $\langle ensemble-spéc \rangle$ }
 ✖ 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. Avant la version 2.00, cette commande était appelée `\xdefinecolor`, cette dernière restant disponible pour des raisons de compatibilité.

— `\definecolorset{rgb}{x}{10}{red,1,0,0;green,0,1,0;blue,0,0,1}`
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}{x}{10}{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 `<cmd>[<type>]{<name>}{<model>}{<spec>}`, where `<cmd>` 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 `{<id-list>}`
Recall that `<id-list>` has the form `<ext id>1, ..., <ext id>l` where each `<ext id>λ` is either an identifier `<id>λ` or an assignment `<id>λ'=<id>λ`. We consider the first case to be an abbreviation for `<id>λ=<id>λ` and describe the general case: the definition stack is searched for the name `<id>λ` and its corresponding color parameters; if there is no match, nothing happens; if the name `<id>λ` is on the stack and its color parameters are `<type>λ`, `<model>λ`, and `<spec>λ`, then the command `\definecolor[<type>λ]{<id>λ'}{<model>λ}{<spec>λ}` is executed. Thus, the user may control by which names the *prepared* colors are to be used in the document. Note that the entry `<id>λ` is not removed from the stack, such that it can be used several times (even within the same `\definecolors` command).

\providecolors `{<id-list>}`
Similar to `\definecolors`, but based on `\providecolor`, thus the individual colors are defined only if they do not exist already.

2.5.5 Global color definitions

\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 `{<color>}`
`[<model-list>]{<spec-list>}`
Switches to the color given either by name/expression or by model/specification. This color will stay in effect until the end of the current `TeX` group.

\textcolor `{<color>}{<text>}`
`[<model-list>]{<spec-list>}{<text>}`
are just alternative syntax for `\color`, in which the groups are added implicitly. Thus `<text>` appears in the specified color, but then the color reverts to its previous value. Additionally, it calls `\leavevmode` to ensure the start of horizontal mode.

\pagecolor `{<color>}`

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 :

-  `\fcolorbox[gray]{yellow}{test}`,
-  `\fcolorbox[cmk]{0,0,0,0.5}{0,0,1,0}{test}`,
-  `\fcolorbox[gray]{0.5}[wave]{580}{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 page 37 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,cmyk,hsb,HTML`.

`\testcolor` {*color*}
[*model-list*][*spec-list*]

Each `\testcolor` command generates a table row, containing a display sample plus the respective parameters for each of the models. If the column-model matches the model of the color in question, its parameters are underlined. Note that this command is only available within the `testcolors` environment.




For applications see figure 2 page 35 and figures 11, 12.

2.7 Color blending

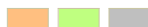

The purpose of *color blending* is to add some mixing color (expression) to all subsequent explicit color commands. Thus, it is possible to perform such a mix (or blend) operation for many colors without touching the individual commands.

`\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 **rg**b or **cm**y model, see the last two rows in figure 9 page 39.

`\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 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 page 38 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 page 39. 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 page 38.

2.13 Color information

<code>\extractcolorspec</code>	$\{\langle color \rangle\}\{\langle cmd \rangle\}$ Extracts the color specification of $\langle color \rangle$ and puts it into $\langle cmd \rangle$; equivalent to <code>\def\cmd{\{\langle model \rangle\}\{\langle spec \rangle\}}</code> .
<code>\extractcolorspecs</code>	$\{\langle color \rangle\}\{\langle model-cmd \rangle\}\{\langle color-cmd \rangle\}$ 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>	$=\langle int \rangle$ 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>	$\{\langle model \rangle\}\{\langle spec \rangle\}\{\langle target model \rangle\}\{\langle cmd \rangle\}$ 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 $\langle cmd \rangle$. $\langle target model \rangle$ 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 pdf \TeX

Since pdf \TeX 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 [12]. 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* :

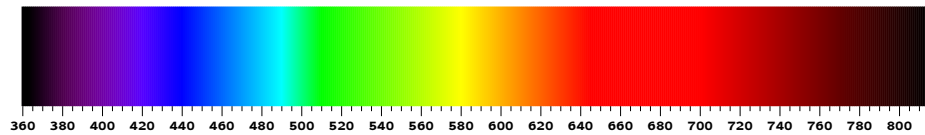
```
\documentclass{minimal}
\usepackage[fixinclude]{xcolor}
\usepackage{graphics}
```

```
\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

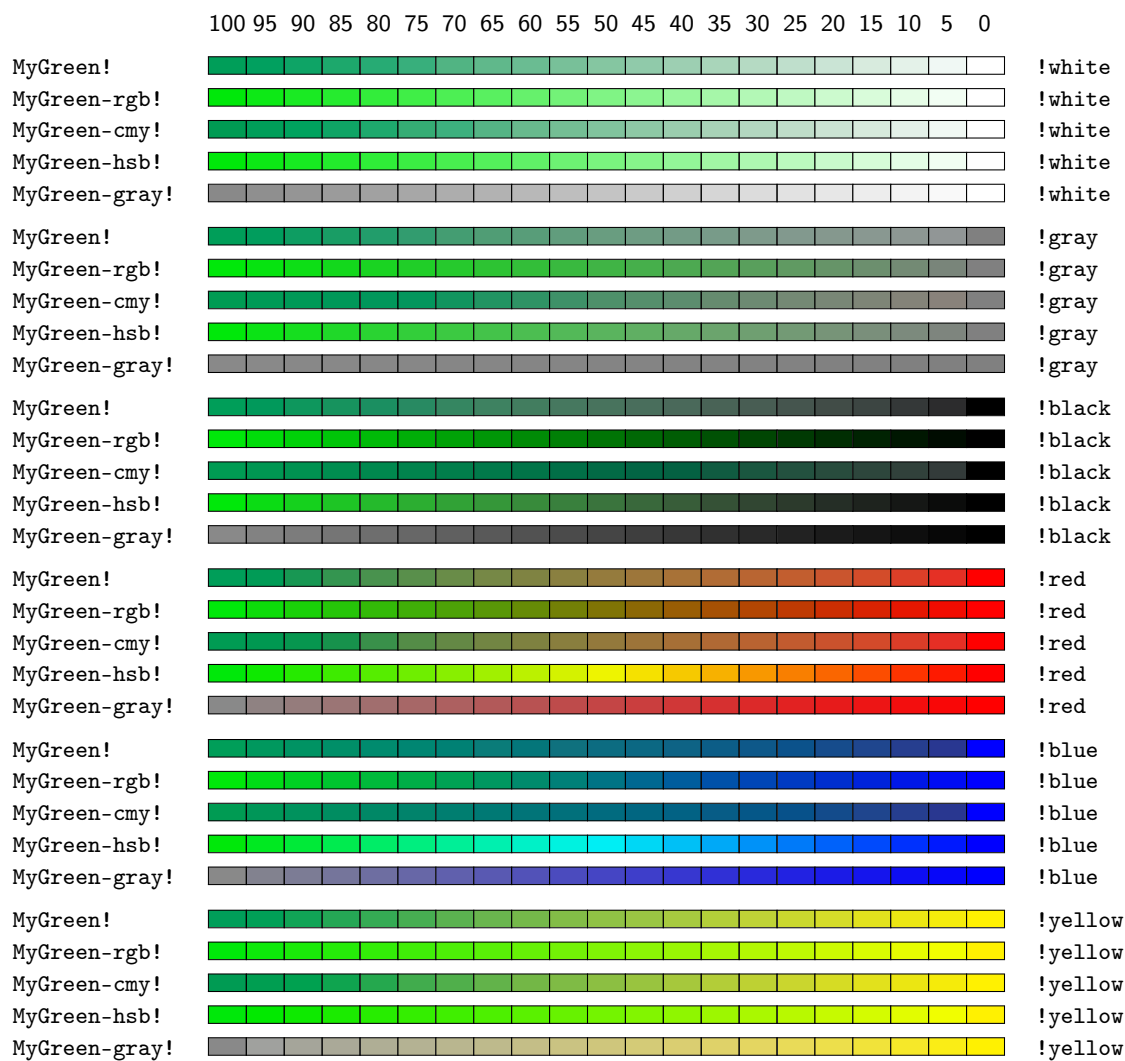
color	rgb	cmYk	hsb	HTML	gray
olive	 0.5 0.5 0	 <u>0 0 1 0.5</u>	 0.16667 1 0.5	 808000	 0.39
red!50!green	 <u>0.5 0.5 0</u>	 0 0 0.5 0.5	 0.16667 1 0.5	 808000	 0.445
-cyan!50!magenta	 0.5 0.5 0	 <u>0 0 0.5 0.5</u>	 0.16667 1 0.5	 808000	 0.445
[cmYk]0,0,1,0.5	 0.5 0.5 0	 <u>0 0 1 0.5</u>	 0.16667 1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

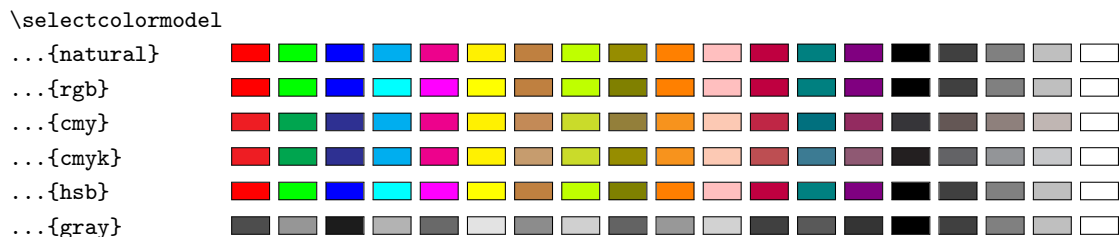


FIGURE 5 – Standard color expressions

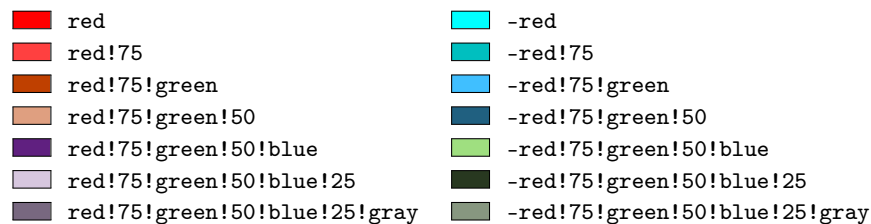


FIGURE 6 – Standard color expressions

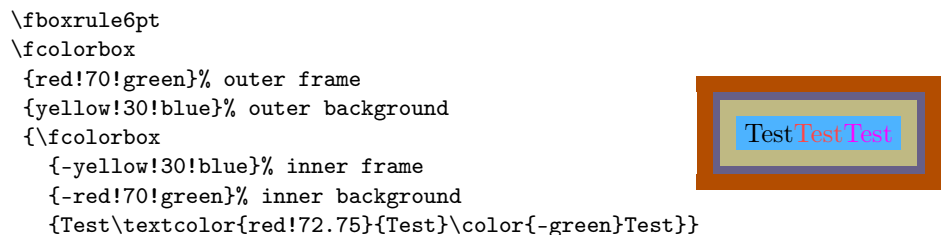


FIGURE 7 – Current color

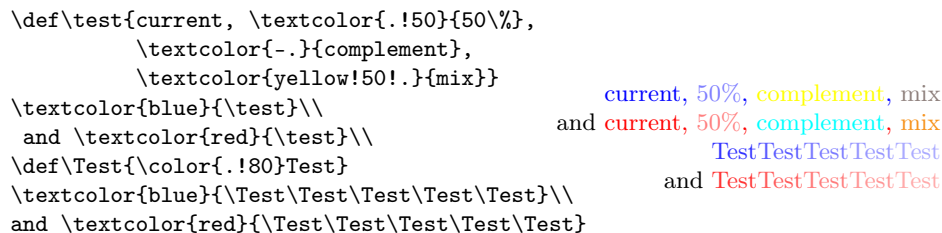


FIGURE 8 – Color series

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\\
\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 [13] 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	myk	hsb	gray	RGB	HTML	HSB	Gray
dvipdf	1999/02/16 v3.0i	d	n	d	n	d	i	n	n	n
dvips	1999/02/16 v3.0i	d	n	d	d	d	i	n	n	n
dvipsone	1999/02/16 v3.0i	d	n	d	d	d	i	n	n	n
pctex32	1999/02/16 v3.0i	d	n	d	d	d	i	n	n	n
pctexps	1999/02/16 v3.0i	d	n	d	d	d	i	n	n	n
pdftex	2006/03/02 v0.03p	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
dvipdfm	1999/9/6 vx.x ²	d	n	d	a	d	i	n	n	n
dvipdfmx	?	d	n	d	f	d	i	n	n	n
textures	1997/5/28 v0.3	d	n	d	a	i	n	n	n	n
vtex	1999/01/14 v6.3	d	n	d	n	i	i	n	n	n
xetex	2004/05/09 v0.7	i	n	i	i	i	i	d	n	n
tcidvi	1999/02/16 v3.0i	i	n	i	n	i	d	n	n	n
truotex	1999/02/16 v3.0i	i	n	i	n	i	d	n	n	n
dviwin	1999/02/16 v3.0i	n	n	n	n	n	n	n	n	n
emtex	1999/02/16 v3.0i	n	n	n	n	n	n	n	n	n
pctexhp	1999/02/16 v3.0i	n	n	n	n	n	n	n	n	n
pctexwin	1999/02/16 v3.0i	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 [11] 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 page suivante shows some examples for the two most prominent drivers. See also figure 3 page 36 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 $\text{T}_{\text{E}}\text{X}$'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

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

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

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

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 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 page 49 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 cmy 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 page 50.

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

Conversion rgb to HTML As described in section 6.2.1 page 50. 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 page 52.

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

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.

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

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

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

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

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

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{\langle\beta_c\rangle,\langle\beta_m\rangle,\langle\beta_y\rangle,\langle\beta_k\rangle}` 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 page 50.

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

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 page 50.

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

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 page 50.

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

6.3.9 The HTML model

Conversion HTML to rgb As described in section 6.2.2 page 50 : 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 page 50.

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

6.3.11 The Gray model

Conversion Gray to gray As described in section 6.2.2 page 50.

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

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, troisième édition, 1999. <http://www.adobe.com/products/postscript/pdfs/PLRM.pdf>
- [2] Donald Arseneau : « Patch so `\fbox` draws frame on top of text ». L^AT_EX bug report, latex/3655, 18/03/2004.
<http://www.latex-project.org/cgi-bin/ltxbugs2html?pr=latex/3655>
- [3] Donald Arseneau : extension url, « 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 L^AT_EX Color ». CTAN/macros/latex/required/graphics/color.dtx
- [7] David P. Carlisle : extension colortbl, « 2001/02/13 v0.1j Color table columns ». CTAN/macros/latex/contrib/colortbl/
- [8] David P. Carlisle, Herbert Voß, Rolf Niepraschk : extension pstcol, « 2005/11/16 v1.2 LaTeX wrapper for ‘PSTricks’ ». CTAN/macros/graphics/pstricks/latex/pstcol.sty
- [9] Uwe Kern : « Chroma : a reference book of L^AT_EX colors ». CTAN/info/colour/chroma/ et <http://www.ukern.de/tex/chroma.html>
- [10] Uwe Kern : extension xcolor, « L^AT_EX color extensions ». CTAN/macros/latex/contrib/xcolor/ et <http://www.ukern.de/tex/xcolor.html>
- [11] Rolf Niepraschk : extension colorinfo, « 2003/05/04 v0.3c Info from defined colors ». CTAN/macros/latex/contrib/colorinfo/
- [12] Heiko Oberdiek : extension pdfcolmk, « 2006/02/20 v0.8 PDFtex COlor MarK ». CTAN/macros/latex/contrib/oberdiek/pdfcolmk.*
- [13] Projet MiK_TE_X : <http://www.miktex.org/>
- [14] Sebastian Rahtz, Heiko Oberdiek : extension hyperref, « 2006/09/06 v6.75e Hypertext links for L^AT_EX ». CTAN/macros/latex/contrib/hyperref/
- [15] Alvy Ray Smith : « Color Gamut Transform Pairs ». *Computer Graphics* (ACM SIGGRAPH), Volume 12, Numéro 3, Août 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>

Annexes

Remerciements

Cette extension se base sur [6] (Copyright (C) 1994–1999 David P. Carlisle) et contient du code de cette extension, cette dernière faisant partie de l’« ensemble graphique » du standard L^AT_EX. Bien que de nombreuses commandes et fonctionnalités ont été ajoutées et que la plupart des commandes originales de `color` ont été réécrites ou adaptées dans `xcolor`, cette dernière n’existerait pas sans `color`. Aussi, l’auteur est reconnaissant à David P. Carlisle d’avoir créé `color` et ses fichiers associés.

Marques déposées

Des Marques déposées apparaissent tout au long de cette documentation sans aucun symbole les dénotant ; elles sont la propriété de leur dépositaire respectif. Il n’y a ici aucune intention d’infraction ; l’utilisation est au bénéfice du dépositaire de la marque.

Problèmes connus

- `\rowcolors[\hline]...` ne fonctionne pas avec `longtable`.

Historique

21/01/2007 v2.11

- Nouvelles fonctionnalités :
 - les noms de couleur *lime* et *teal* sont ajoutés à l’ensemble des couleurs prédéfinies.
- Correction d’erreur :
 - appel incorrect de `\XC@strip@comma` dans les options liées à `hyperref`.

28/11/2006 v2.10

- ✖
- Nouvelles fonctionnalités :

- `fixinclude` option prevents *dvips* from explicitly resetting current color to *black* before actually inserting an `.eps` file via


```
\color{red}\includegraphics{foo}.
```

— Changements :

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

— Corrections d’erreur :

- incorrect processing of **cm**yk-type current color ‘.’.

✖

21/12/2005 v2.09

✖

— Nouvelles fonctionnalités :

- `\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.

— Changements :

- 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.

✖

25/11/2005 v2.08

✖

— Nouvelles fonctionnalités :

- 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;
- `kernelbox` 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`.
- Corrections d’erreur :
 - insufficient expression type detection within `\colorlet`;
 - wrong calculation in the unit interval reduction for negative integers (affecting color series and extended color expressions).

✖

12/11/2005 v2.07

✖

- Nouvelles fonctionnalités :
 - 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;

- Changements :
 - enhanced documentation;
 - several changes to internal macros.
- Corrections d’erreur :
 - wrong calculation of color series components in some cases of negative step parameters.

✖

15/10/2005 v2.06

✖

- Nouvelles fonctionnalités :
 - 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`.
- Changements :
 - `\XC@strip@comma` doesn’t generate a trailing space anymore, which improves also the output of the `testcolors` environment.

✖

30/09/2005 v2.05

✖

- Nouvelles fonctionnalités :
 - `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.
- Corrections d'erreur :
 - `\definecolor{foo}{named}{bar}` did not work in v2.04.

✖

23/09/2005 v2.04

✖

- Nouvelles fonctionnalités :
 - 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`.
- Changements :
 - 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.

✖

06/06/2005 v2.03

✖

- Nouvelles fonctionnalités :
 - `fixpdftex` option loads `pdfcolmk` package in order to improve pdf_TE_X's color behaviour during page breaks.
- Changements :
 - some minor changes to internal macros.
- Corrections d'erreur :

- 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.

✖

24/03/2005 v2.02

✖

- Nouvelles fonctionnalités :
 - `\aftergroupdef` 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.
- Changements :
 - `\rowcolors` and friends are solely enabled by the `table` option;
 - `\@ifxempty` changed back to more robust variant of v2.00.
- Corrections d'erreur :
 - `\psset{linecolor=\ifcase\foo red\or green\or blue\fi}` did not work with `pstricks` (error introduced in v2.01).

✖

15/03/2005 v2.01

✖

- Nouvelles fonctionnalités :
 - `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.
- Changements :
 - 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.
- Corrections d’erreur :
 - `\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.

✖

04/07/2004 v2.00

✖

- Nouvelles fonctionnalités :
 - 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.
- Changements :
 - 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.
- Corrections d’erreur :
 - `\definecolor{foo}{named}{bar}` did not work (error introduced in v1.11);
 - more robust behaviour of conditionals within `pstricks` key-values.

✖

09/05/2004 v1.11

✖

- Nouvelles fonctionnalités :
 - 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.
- Changements :
 - `\XC@logcolor` renamed to `\XC@display`, which is now the core color display command;
 - improved interface to `pstricks`.

✖

27/03/2004 v1.10

✖

- Nouvelles fonctionnalités :
 - 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.

- Corrections d’erreur :
 - 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.

✖

16/02/2004 v1.09

✖

- Nouvelles fonctionnalités :
 - 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.
- Nouvelle page web de xcolor : www.ukern.de/tex/xcolor.html
- Correction d’erreur : `\xdefinecolor` sometimes did not normalise its parameters.
- Changements :
 - slight improvements of the documentation;
 - example file `xcolor1.tex` reorganised and abridged.

✖

04/02/2004 v1.08

✖

- New commands :
 - `\selectcolormodel` to change the target model within a document;
 - `\adjustUCRBG` to fine-tune undercolor-removal and

black-generation during conversion to **cmyk**.

- Bugfix : color expressions did not work correctly in connection with active ‘!’ character, e.g., in case of `\usepackage[frenchb]babel`.
- Code re-organisation :
 - `\XC@xdefinecolor` merged into `\xdefinecolor`, making the first command obsolete;
 - several internal commands improved/streamlined.

✖

20/01/2004 v1.07

✖

- Nouvelle fonctionnalité : support for color masking and color separation.
- Nouvelles commandes :
 - `\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.
- Changements :
 - 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.
- Fichiers d'exemples : additional `pstricks` examples (file `xcolor2.tex`).

✖

15/12/2003 v1.06

✖

- New feature : extended color expressions, allowing for cascaded mix operations, e.g., `\color{red!30!green!40!blue}`.
- Documentation : new section on color expressions.
- Bugfix : color series stepping did not work correctly within non-displaying commands

like `\extractcolorspec{foo!!+}` (this bug was introduced in v1.05).

- Renamed commands : `\ukfileversion` and similar internal constants renamed to `\XCfileversion` etc.
- Removed commands : `\ifXCpst` and `\ifXCtable` made obsolete by a simple trick.

✖

21/11/2003 v1.05

✖

- Corrections d'erreur :
 - package option `hideerrors` should now work as expected;
 - usage of ‘.’ in the first color expression in a document caused an error due to incorrect initialisation.
- Réorganisation du code : `\extractcolorspec` now uses `\XC@splitcolor`, making `\XC@extract` obsolete.

✖

09/11/2003 v1.04

- Nouvelle fonctionnalité : accès simplifié à la couleur courante dans les expressions de couleur.
- Nouvelle option : `override` pour remplacer `\definecolor` par `\xdefinecolor`.
- Nouvelle commande : `\tracingcolors` pour afficher dans le fichier journal les informations spécifiques aux couleurs. information.

21/09/2003 v1.03

- Changement : contournement du comportement étrange de certains pilotes.
- Nouvelle fonctionnalité : partage de pilote avec `hyperref`.

19/09/2003 v1.02

- Changement : `\extractcolorspec` et `\colorlet` acceptent maintenant aussi les séries de couleur comme arguments.

15/09/2003 v1.01

- Nouvelle fonctionnalité : `\definecolorseries` et apparentés.
- Documentation : retrait de certains effets indésirables liés à `doc`.

- Réorganisation du code : tous les outils de calculs sont placés en un seul endroit.
- Corrections d'erreur :
 - `\@rdivide` : ajout d'un `\relax` pour résoudre le problème des numérateurs négatifs ;
 - `\rowc@l@rs` : remplacement de `\@ifempty` par `\@ifxempty`.

09/09/2003 v1.00

- Première version publiée.

Index

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

A		<i><spec-list></i> 14, 15	gray . . . 6, 9–12, 15, 17, 44, 48, 50, 53, 54, 57, 58
<code>\adjustUCRBG</code> 9, 54	<i><spec></i> 14, 15	hsb 5, 9–12, 15, 18, 21, 44, 45, 47, 48, 50–52, 55–59	
arguments	<i><type></i> 14, 15	rgb . . . 6, 9–12, 15, 19, 21, 26, 29, 44, 48, 50–59	
<i><color expr></i> 14, 16	B		tHsb 11, 15, 18, 39, 48, 57, 58, 63
<i><color></i> 14, 16	<code>\blendcolors</code> 25	wave . 11, 12, 15, 48, 59, 63	
<i><core model></i> 14, 15	<code>\blendcolors*</code> 25	'named' 15, 21, 66	
<i><dec></i> 13, 14	<code>\boxframe</code> 24	'ps' 15, 63	
<i><div></i> 13, 14	C		color names
<i><empty></i> 13, 14	<code>\color</code> 23	<i>Fuchsia</i> 31, 32	
<i><expr></i> 14, 15	color expression 21	<i>Gray0</i> 19	
<i><ext expr></i> 14, 16	color functions	<i>Green0</i> 19	
<i><ext id></i> 14	<i>twheel</i> 18	<i>Grey0</i> 19	
<i><func expression></i> 16	<i>wheel</i> 18	<i>Maroon0</i> 19	
<i><func expr></i> 14	color models	<i>Purple0</i> 19	
<i><function></i> 14, 16	Gray . 10–12, 15, 44, 48, 58	argent 6	
<i><id-list></i> 14	HSB 10–12, 15, 44, 48, 56, 58	black 9, 10, 18, 26, 32, 59, 62	
<i><id></i> 14	HTML 9–	blanc 6	
<i><int></i> 13, 14	11, 15, 44, 48, 53, 58, 66	bleu 6, 7	
<i><minus></i> 13, 14	Hsb 11, 15, 18, 39, 48, 56–58, 63	blue 18, 20, 22, 59	
<i><mix expr></i> 14, 15	RGB 10–12, 15, 19, 44, 48, 53, 58, 66	brown 18, 66	
<i><model-list></i> 14, 15	cmymk 5, 6, 9–	cyan 18, 26, 59	
<i><model></i> 14, 15	11, 15, 19, 21, 26, 44,	darkgray 18	
<i><name></i> 13, 14	48, 50, 53, 54, 58, 62, 66	foo 20, 26	
<i><num model></i> 14, 15	cmym 9–11, 15,	gray 18	
<i><num></i> 13, 14	26, 44, 48, 51, 53, 54, 57	green . . . 5, 17, 18, 22, 59	
<i><pct></i> 13, 14			
<i><plus></i> 13, 14			
<i><postfix></i> 14, 15			
<i><prefix></i> 14, 15			

<i>gris</i>	6	.ps	20, 29, 33	xpagebordercolor	29
<i>jaune</i>	7	.sty	65	xpdfborder	29, 30, 65
<i>lightgray</i>	18	.xcp	10, 20	xrunbordercolor	29
<i>lime</i>	18, 62	color.pro	20	xurlbordercolor	29
<i>magenta</i>	18, 26, 59	dvipshnam.def	23		
<i>noir</i>	6	rgb.txt	19, 63	M	
<i>olive</i>	18, 63	xcolor.pro	8, 20, 64	\maskcolors	26
<i>orange</i>	18, 66	xcolor.sty	8, 18	MiKTeX	20
<i>or</i>	6	xcolor2.tex		N	
<i>pink</i>	18, 63		30, 60, 62–64, 66	nuaance	6
<i>purple</i>	18, 66	xcolor3.tex	62	P	
<i>red</i>	9, 11,	xcolor4.tex	62, 63	package options	
	17, 18, 20–22, 32, 59, 63	G		Gray	8, 10, 13
<i>rouge</i>	5–7	\GetGinDriver	8	HSB	8, 10, 13
<i>teal</i>	18, 62	\GinDriver	8	HTML	8, 10, 13
<i>vert</i>	6, 7	H		RGB	8, 10, 13
<i>violet</i>	18, 66	\hiderowcolors	30	cmymk	8, 10, 13, 26
<i>white</i>	18	HKS	6	cmym	8, 10, 13
<i>yellow</i>	5, 11, 17, 18, 26, 59	HTML4	19, 42	dviptdfmx	8, 44, 63
color set	21	I		dviptdfm	8, 20, 25, 44, 45, 62
color stack	32	\ifconvertcolorsD	13	dviptdf	8, 44
\colorbox	24	\ifconvertcolorsU	13	dviptdnames*	
\colorlet	21	\ifdefinecolors	22		8, 10, 18, 19, 64
\colormask	27	\ifglobalcolors	23	dviptdnames	
\colorseriescycle	28	\ifmaskcolors	26		8–10, 18–20, 31, 41, 65
\convertcolorspec	31	K		dviptsone	8, 44
D		keys		dvipt	20, 25, 30, 44, 46, 63–65
\definecolor	20, 30	citebordercolor	29	dviwindo	8, 44
\definecolors	23	citecolor	29	dviwin	8, 44
\definecolorseries	27	filebordercolor	29	emtex	8, 44
\definecolorset	21	filecolor	29	fixinclude	8, 10, 33, 62
\DefineNamedColor	22	linkbordercolor	29	fixpdftex	8, 10, 11, 32, 64
definition stack	22, 23	linkcolor	29	gray	8, 10, 13
E		menubordercolor	29	hideerrors	8, 10, 67
environments :		menucolor	29	hsb	8, 10, 13
testcolors	25	pagebordercolor	29	hyperref	8, 10, 29, 65
\extractcolorspec	31	pagecolor	29	hypertex	8
\extractcolorspecs	31	pdfborder	29	kernelfbox	8, 10, 24, 62
F		runbordercolor	29	monochrome	8, 44
\fcolorbox	24	runcolor	29	natural	8, 10, 13
files		urlbordercolor	29	noxdraw	8, 10, 25
.def	65	urllcolor	29	oztex	8, 44
.dvi	19, 20, 25, 29	xcitebordercolor	29	pctex32	8, 44
.eps	10, 27, 32, 33, 62	xfilebordercolor	29	pctexhp	8, 44
.jpg	27	xlkbordercolor	29	pctexpss	8, 44
.pdf	27	xmenubordercolor	29	pctexwin	8, 44
.png	27				

pdftex	memoir	64	\providecolor	21
. 8, 20, 25, 35, 44–46, 62	msc	64	\providecolors	23
prologue ... 8, 10, 20, 64	pdfcolmk 10, 11, 32, 61, 64		\providecolorset	22
rgb	preview	64		
showerrors	pstcol	11, 61	R	
svgnames* 8, 10, 19, 22, 64	pstricks 11, 21, 30, 60, 62–66		\rangeGray	12
svgnames	realcalc	45	\rangeHSB	12
. 8, 10, 19, 22, 31, 41, 63	soul	24, 66	\rangeHsb	11, 57
table ... 8, 10, 11, 30, 64	url	61	\rangeRGB	12
tcidvi	xcolor	1, 5, 6, 8, 9, 11–13, 18, 20, 22, 23, 27–29, 31, 32, 44–46, 60–62, 64–66	\rangetHsb	11, 57
textures			\resetcolorseries	28
truetex	\pagecolor	23	\rowcolors	30
usecolors	Pantone	6	\rowcolors*	30
vtex	PDF	25	\rownum	30
x11namees	people		S	
x11names*	Arseneau, Donald ..	24, 61	\selectcolormodel	12
x11names ... 8, 10, 19, 42	Bruton, Dan	59, 61	\showrowcolors	30
xcdraw .. 8, 10, 25, 62, 63	Carlisle, David P. ..	61, 62	\substitutecolormodel ..	12
xdvi	Goethe, Johann Wolfgang		SVG	10, 19, 32, 61, 65
xetex	von	7	T	
package options (obsolete)	Kern, Uwe	61	teinte	6
nodvipsnames	Newton, Isaac	7	\testcolor	25
override	Niepraschk, Rolf	61	testcolors (environment) .	25
pst	Oberdiek, Heiko	61	\textcolor	23
usenames	Rahtz, Sebastian	61	ton	6
packages	Smith, Alvy Ray ...	50, 61	ton direct	6
colorinfo	Voß, Herbert	61	\tracingcolors	31
colortbl 10, 11, 30, 61	PostScript	9, 15, 20, 25, 30, 32, 50, 52, 63, 64	U	
color	\preparecolor	22	Unix	10, 19, 42, 63
. 5, 6, 8, 10, 11, 19, 21, 23, 24, 29, 31, 44–46, 61, 62, 65	\preparecolorset	22	X	
doc	programs		X11	10, 19, 42, 63
fp	Yap	20	\xcolorcmd	9
graphics	dvipdfm	20	\xglobal	23, 26
graphicx	dvips	8, 20, 29, 30, 32, 33, 62, 64–66		
hyperref				
. 8, 10, 11, 29, 61, 62, 67				
longtable				
. 62				