

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

Dr. Uwe Kern

v2.12 (11/05/2016) *

Résumé

L’extension `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}`. Elle permet de sélectionner un modèle de couleur cible 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	Terminologie des couleurs pour ce document	6
1.3	Modèles colorimétriques	6
1.4	Cercles chromatiques et accords de couleurs	7
2	L’interface utilisateur	9
2.1	Préparation	9
2.1.1	Installation de l’extension	9
2.1.2	Options de l’extension	9
2.1.3	Exécution de commandes additionnelles à l’initialisation . .	10
2.2	Modèles colorimétriques	10
2.2.1	Modèles colorimétriques supportés	10
2.2.2	Substitution de modèles colorimétriques individuels	14
2.2.3	Changement du modèle colorimétrique cible dans un document	14
2.3	Arguments et terminologie	14

* 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 .	16
2.3.2	Signification des expressions de couleur standards	18
2.3.3	Signification des expressions de couleur étendues	19
2.3.4	Fonctions de couleur	19
2.4	Couleurs prédéfinies	20
2.4.1	Couleurs qui sont toujours disponibles	20
2.4.2	Ensembles additionnels de couleurs	20
2.5	Définition de couleur	21
2.5.1	Couleurs ordinaires et nommées	21
2.5.2	Définition de couleur dans <code>xcolor</code>	23
2.5.3	Définition d'ensembles de couleur	23
2.5.4	Définitions immédiates et différées	24
2.5.5	Définitions de couleur globales	25
2.6	Utilisation de la couleur	26
2.6.1	Commandes de couleur standards	26
2.6.2	Boîtes colorées	26
2.6.3	Utilisation de la couleur courante	27
2.6.4	Test de couleurs	28
2.7	Glissement de couleur	28
2.8	Masques de couleur et séparation	29
2.9	Séries de couleurs	30
2.9.1	Définition d'une série de couleurs	30
2.9.2	Initialisation d'une série de couleurs	31
2.9.3	Utilisation d'une série de couleurs	31
2.9.4	Différences entre couleurs et séries de couleurs	31
2.10	Couleur d'encadrement d'hyperliens	32
2.11	Spécifications de couleurs additionnelles le monde de <code>pstricks</code> . . .	33
2.12	Couleur dans des tableaux	33
2.13	Information sur la couleur	34
2.14	Conversion de couleur	34
2.15	Problèmes et solutions	35
2.15.1	Name clashes between <code>dvipsnames</code> and <code>svgnames</code>	35
2.15.2	Page breaks and <code>pdfTeX</code>	35
2.15.3	Change color of included <code>.eps</code> file	35
3	Exemples	37
4	Colors by Name	44
4.1	Base colors (always available)	44
4.2	Colors via <code>dvipsnames</code> option	44
4.3	Colors via <code>svgnames</code> option	44
4.4	Colors via <code>x11names</code> option	45

5	Technical Supplement	47
5.1	Color models supported by drivers	47
5.2	How xcolor handles driver-specific color models	47
5.3	Behind the scenes : internal color representation	48
5.4	A remark on accuracy	48
6	The Formulas	50
6.1	Color mixing	50
6.2	Conversion between integer and real models	50
6.2.1	Real to integer conversion	52
6.2.2	Integer to real conversion	52
6.3	Color conversion and complements	53
6.3.1	The rgb model	54
6.3.2	The cmY model	56
6.3.3	The cmYk model	57
6.3.4	The hsb model	58
6.3.5	The Hsb model	60
6.3.6	The tHsb model	60
6.3.7	The gray model	60
6.3.8	The RGB model	61
6.3.9	The HTML model	61
6.3.10	The HSB model	61
6.3.11	The Gray model	61
6.3.12	The wave model	62
	Références	64
	Annexes	65
	Remerciements	65
	Marques déposées	65
	Problèmes connus	65
	Historique	65
	Index	71

Liste des tableaux

1	Options de l'extension	11
2	Ordre de chargement des extensions	12
3	Modèles colorimétriques supportés	12
4	Arguments et terminologie associée	15
5	Drivers and color models	47
6	Driver-dependent internal color representation	49
7	Color constants	51
8	Color conversion pairs	51

Table des figures










1	Color spectrum	37
2	Color testing	37
3	Progressing from one to another color	38
4	Target color model	39
5	Standard color expressions	39
6	Standard color expressions	39
7	Current color	39
8	Color series	40
9	Color masking	41
10	Alternating row colors in tables : \rowcolors vs. \rowcolors* . .	41
11	Hsb and tHsb : <i>hue</i> [°] in 15° steps	42
12	Color harmony	43

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)L^AT_EX de façon cohérente, indépendamment des pilotes graphiques, tout en supportant différents modèles colorimétriques (de manière un peu plus dépendante 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 impose d'habitude l'utilisation d'un autre programme qui calcule les paramètres souhaités, paramètres alors copiés dans une commande `\definecolor` dans L^AT_EX. Assez fréquemment, une calculatrice de poche est également impliquée 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 L^AT_EX ?
(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 pdfL^AT_EX favorite. Que faire alors ?
(Réponse : utiliser tout simplement les définitions **hsb**, xcolor fera les calculs nécessaires.)
- Qu'affiche le mélange de 40% de vert (*green*) et de 60% de jaune (*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 (*red*). Qu'est-ce que cela donne ?
(Réponse : $3 \times$  $+ 2 \times$  $+ 1 \times$  = , cette couleur étant accessible avec `\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 **cm^yk**¹. 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 recourir à la copie de 50 commandes `\rowcolor` ? Ce motif

1. NdT : CMJN en français.

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 Terminologie des couleurs pour ce document

Sur la base de [15] nous définissons les termes suivants² :

- **couleur éclaircie** : une couleur à laquelle est ajoutée du blanc (*white*);
- **couleur assombrie** : une couleur à laquelle est ajoutée du noir (*black*);
- **couleur assourdie** : 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

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

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

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

où `white`, `black`, et `gray` sont des constantes dépendantes des modèles, comme présentées en table 7 page 51. Par la suite, nous définissons le terme :

- **couleur complémentaire** : une couleur C^* qui génère du blanc (*white*) 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 53 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

2. N.D.T. : ces termes ne sont pas l'exacte traduction des termes anglais car les notions de teinte (*tint*), nuance (*shade*) et ton (*tone*) utilisées par l'auteur sont bien trop polysémiques en français. Nous conservons ici juste la notion de « teinte » à laquelle nous adjoignons des adjectifs bien moins sujets à discussion, en évitant ici les adjectifs « rompu », « rabattu » ou « désaturé ».

construit par combinaison linéaire des couleurs de base rouge (*red*), vert (*green*) et bleu (*blue*). En contrepartie, un ton direct ne peut souvent être qu'*approximé* par des valeurs de paramètres dans les modèles comme **cm**yk ou **rg**b ; les couleurs originales se mélangent physiquement et dépendent aussi du type de papier retenu. Enfin, il existe certaines couleurs comme l'or (*gold*) ou l'argent (*silver*) qui sont difficilement reproductibles par des modèles paramétriques avec des encres standards ou des imprimantes laser.

1.4 Cercles chromatiques et accords de couleurs

Il s'est développé une longue histoire du placement de couleurs (les teintes saturées) 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 la couleur de l'autre — tendent à utiliser des termes identiques pour décrire des 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 (*red*), jaune (*yellow*), bleu (*blue*), tandis que les scientifiques spécialistes de la couleur préféreront le triplet rouge (*red*), vert (*green*), bleu (*blue*). 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 **triades** correspondent à trois couleurs séparées par 120° ;
- les **tétrades** 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 $\pm \frac{2}{2n} \cdot 360^\circ$.

Compte tenu des 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 affecte le résultat visuel. Des exemples sont donnés en figure 12 page 43.

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 `.def` dans un répertoire où (pdf)LaTeX les trouvera. Un emplacement classique selon la structure des répertoires de TeX³ 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 12 montre dans quel ordre certaines extensions doivent alors être 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`, `luatex`, `pdftex`, `dvipsone`, `dviwindo`, `emtex`, `dviwin`, `oztex`, `textures`, `pctexps`, `pctexwin`, `pctexhp`, `pctex32`, `truetex`, `tcidvi`, `vtex`, `xetex`;
- les options qui déterminent le modèle colorimétrique cible⁴ (`natural`, `rgb`, `cmv`, `cmv`, `hsb`, `gray`, `RGB`, `HTML`, `HSB`, `Gray`) ou qui génèrent 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 11. Afin de faciliter la coopération avec l'extension `hyperref`, il existe une commande `\GetGinDriver`⁵ 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

3. N.D.T. : *TeX Directory Structure* (TDS)

4. La section 2.2.3 page 14 explique comment ce paramétrage peut être annulé n'importe où dans un document.

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

extension), voir l'exemple de code en page 13. S'il n'y a pas d'option `hyperref` correspondante, l'option `hypertex` sera prise par défaut.

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` doit toujours recevoir 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 moyen 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 (*black*) remplacées par du rouge (*red*), 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 12. 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 vous renvoyons 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 56 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 à **rgb**. En terme d'affichage, **cmy** est traité comme **cmyk** avec $k = 0$.

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 .
cmv	Convertit toutes les couleurs en modèle cmv .
cmv	Convertit toutes les couleurs en modèle cmv .
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 spécification 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 35).
hyperref	Permet de prendre en charge l’extension hyperref pour les expressions de couleur en définissant des clés additionnelles (voir section 2.10 page 32).
prologue	Écrit des informations en début de fichier .xcp pour chaque définition de couleur (comme décrit en section 2.5.1 page 21).
kernelbbox	Utilise la méthode du noyau L^AT_EX pour dessiner des boîtes \f(rame)box ² .
xcdraw	Utilise des commandes propres aux pilotes pour dessiner les encadrements et boîtes de couleur 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 35).
showerrors	(Valeur par défaut.) Affiche un message d’erreur si une couleur non définie est utilisée (comportement identique 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 noir (<i>black</i>).

¹ Voir section 2.4.2 page 20. ² Voir section 2.6.2 page 26.

TABLE 2 – Ordre de chargement des extensions

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

TABLE 3 – Modèles colorimétriques supportés

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

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

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 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°* $\in [0, H]$, où H est donné par `\def\rangeHsb{<H>}`. Aussi, si $H = 360$, nous pouvons imaginer un cercle ou une roue pour décrire le paramètre *teinte°*. Deuxièmement, **Hsb** est la base du **tHsb**, également nommé **Hsb transformé**, qui permet à l'utilisateur d'appliquer une transformation linéaire par morceaux sur *teinte°* en déplaçant la *teinte°* sélectionnée en avant ou en arrière sur

`\rangetHsb` le cercle. 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 (*yellow*) est placé à 60° dans le cercle **Hsb** (le rouge (*red*) étant à 0°); cependant dans la plus plupart des cercles chromatiques servant aux artistes, le jaune (*yellow*) 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 60 pour la formule exacte de la transformation et les restrictions détaillées, et la section 1.4 page 7 pour les cercles chromatiques et les accords de couleur. La figure 11 page 42 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 convertir les longueurs d'onde dans des modèles colorimétrique standards 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 37 illustre seulement l'intervalle de valeurs mentionné 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; ceci 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>}`, et `\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` $\{\langle\text{modèle source}\rangle\}\{\langle\text{liste de modèles cibles}\rangle\}$

Cette commande substitue le $\langle\text{modèle source}\rangle$ par le premier modèle disponible apparaissant dans la $\langle\text{liste de modèles cibles}\rangle$. Seuls les modèles de type $\langle\text{modèle numérique}\rangle$ 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

`\selectcolormodel` $\{\langle\text{modèle numérique}\rangle\}$

Cette commande définit le modèle cible comme étant le $\langle\text{modèle numérique}\rangle$ indiqué, ce dernier appartenant à la liste des noms de modèles autorisés comme option de l'extension (soit **natural**, **rgb**, **cm**y, **cm**yk, **hsb**, **gray**, **RGB**, **HTML**, **HSB**, **Gray**), voir figure 4 page 39 pour un exemple. Il y a deux possibilités pour effectuer la conversion au modèle cible :

- `\ifconvertcolorsD` — au moment de la *définition* des couleurs⁶ (avec `\definecolor` et assimilées) ; ceci est contrôlé par la bascule `\ifconvertcolorsD` ;
- `\ifconvertcolorsU` — au moment de l'*utilisation* des couleurs (immédiatement avant que l'affichage de la couleur, ce qui traite les couleurs 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 `\ifconvertcolorsU`.

Les deux bascules valent « vrai » en sélectionnant n'importe quel modèle, à l'exception de **natural** qui leur donne la valeur « faux ». D'autres choix sont autorisés grâce à une option d'extension ou à `\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}` puis `\convertcolorsUfalse`, ce qui correspond en fait à 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.

6. Ceci signifie que toute couleur *nouvellement* définie sera d'abord convertie dans le modèle cible, puis sauvegardée.

TABLE 4 – Arguments et terminologie associée

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)
$\langle \text{expr de mélange} \rangle$	→ $! \langle \text{pct} \rangle_1 ! \langle \text{nom} \rangle_1 ! \langle \text{pct} \rangle_2 ! \langle \text{nom} \rangle_2 ! \dots ! \langle \text{pct} \rangle_n ! \langle \text{nom} \rangle_n$ → $! \langle \text{pct} \rangle_1 ! \langle \text{nom} \rangle_1 ! \langle \text{pct} \rangle_2 ! \langle \text{nom} \rangle_2 ! \dots ! \langle \text{pct} \rangle_n$	(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 ✖)
$\langle \text{expr étendue} \rangle$	→ $\langle \text{modèle central} \rangle, \langle \text{div} \rangle : \langle \text{expr} \rangle_1, \langle \text{déc} \rangle_1 ; \langle \text{expr} \rangle_2, \langle \text{déc} \rangle_2 ; \dots ; \langle \text{expr} \rangle_k, \langle \text{déc} \rangle_k$ → $\langle \text{modèle central} \rangle : \langle \text{expr} \rangle_1, \langle \text{déc} \rangle_1 ; \langle \text{expr} \rangle_2, \langle \text{déc} \rangle_2 ; \dots ; \langle \text{expr} \rangle_k, \langle \text{déc} \rangle_k$	
$\langle \text{expr fonctionnelle} \rangle$	→ $> \langle \text{fonction} \rangle, \langle \text{arg} \rangle_1, \langle \text{arg} \rangle_2, \dots, \langle \text{arg} \rangle_j$	(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, $k, l, m, n > 0, m \leq 8$.	

2.3.1 Remarques additionnelles et restrictions sur les arguments

<vide> **Chaînes basiques et nombres** Ces arguments ne nécessitent pas beaucoup 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> **Modèles colorimétriques**
<modèle numérique>
<modèle> La différence faite entre les *modèles centraux* (**rgb**, **cmy**, **cmk**, **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 12, la seconde — idéalement — faisant référence à des noms définis par le pilote graphique. Exemples : '.1,.2,.3', '.1 .2 .3', '0.56789', '89ABCD', 'ForestGreen'.

$\langle \text{liste-modèle} \rangle$
 $\langle \text{liste-spéc} \rangle$ **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 19 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 *standards* ou *étendues* 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 : ‘>wheel,30’, ‘>wheel,30,’ , ‘>twheel,1,12’, ‘>twheel,-11,12’.

$\langle \text{couleur} \rangle$
 $\langle \text{expr de couleur} \rangle$ **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 ‘!![$\langle \text{num} \rangle$]’, 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. $(\langle \text{pct} \rangle_2, \langle \text{nom} \rangle_2), \dots, (\langle \text{pct} \rangle_n, \langle \text{nom} \rangle_n)$.
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,

$\langle temp \rangle$, puis le mélange est calculé dans le modèle⁷. Ainsi, $\langle temp \rangle ! \langle pct \rangle_\nu ! \langle nom \rangle_\nu$ et $\langle nom \rangle_\nu ! \langle 100 - pct \rangle_\nu ! \langle temp \rangle$ qui devraient être théoriquement équivalents, peuvent ne pas avoir des résultats visuels identiques.

Les figures 5 à 6 page 39 montrent de premières applications des couleurs et expressions. D'autres exemples sont donnés en figure 3 page 38. 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 *modèle central* 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 (*rouge*), 2 parts de  vert (*green*), et une part de  jaune (*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 jaune (*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 dé-
twheel

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




















terminer 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 $\langle cercle\ complet \rangle$ 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 43.

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 14.

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 24, avant de pouvoir être appliqués dans un document.

- `dvipsnames/dvipsnames*` charge un ensemble de 68 noms de couleur **cmymk** 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 couleur⁸ **rgb** respectant la spécification SVG 1.1 [17]⁹, augmenté de 4 noms tirés du fi-

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

9. 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.

chier `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 44.

- `x11names/x11names*` charge un ensemble de 317 noms de couleur¹⁰ **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/X11¹¹. 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 : `DarkSlateGray ()` 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 44. La section 2.15.1 page 35 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.

10. Ces noms représentent 315 couleurs différentes.

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

- **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 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 couleur¹² (**\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 bleu (*blue*) dans la logique usuelle de **xcolor**, bien que le fichier **.ps** va afficher rouge (*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.

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

2.5.2 Définition de couleur dans xcolor

`\definecolor` [*type*]{*nom*}{*liste-modèle*}{*liste-spéc*}¹³

Il s'agit d'une des commandes qui peut être utilisée pour assigner un *nom* 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 expression*, comme expliquée en section 2.3 page 14. La commande remplace à la fois `\DefineNamedColor` et `\definecolor` de `color`. Notez que la définition d'un *nom* 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 34 pour plus d'informations). Les arguments sont décrits en section 2.3 page 14. 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 rouge (*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 33.

`\providecolor` [*type*]{*nom*}{*liste-modèle*}{*liste-spéc*}

Cette commande est similaire à `\definecolor` à ceci près que la couleur *nom* est définie seulement si elle n'existe pas déjà.

`\colorlet` [*type*]{*nom*}[*modèle numérique*]{*couleur*}

Cette commande copie la couleur obtenue avec *couleur* dans *name*. Si *modèle numérique* n'est pas vide, *couleur* est tout d'abord converti dans le modèle spécifié avant que *name* ne soit défini. Le pseudo-modèle 'named' n'est pas autorisé ici mais il peut cependant être spécifié dans l'argument *type*. Notez qu'une couleur nommée *nom* 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` [*type*]{*liste-modèle*}{*tête*}{*queue*}{*ensemble-spéc*}

Cette commande facilite la construction d'un *ensemble de couleurs*, autrement dit un ensemble (potentiellement vaste) de couleurs individuelles ayant en commun une même *liste-modèle* et un même *type*. Ici, *ensemble-spéc* = $\langle nom \rangle_1, \langle liste-spéc \rangle_1 ; \dots ; \langle nom \rangle_l, \langle liste-spéc \rangle_l$ ($l \geq 1$ nom/paires de liste de spécification). Les couleurs individuelles sont construites par des commandes

¹³. Avant la version 2.00, cette commande était appelée `\xdefinecolor`, cette dernière restant disponible pour des raisons de compatibilité.

`\definecolor` [*<type>*] {*<tête>*} *<nom>* _{λ} *<queue>* {*<liste-modèle>*} {*<liste-spéc>* _{λ} }

où $\lambda = 1, \dots, l$. Par exemple,

- `\definecolorset{rgb}{-}{-}{red,1,0,0;green,0,1,0;blue,0,0,1}`
peut être utilisé pour définir les couleurs de base rouge (*red*), vert (*green*),
et bleu (*blue*);¹⁴
- `\definecolorset{rgb}{x}{10}{red,1,0,0;green,0,1,0;blue,0,0,1}`
définit les couleurs `xred10 ()`, `xgreen10 ()` et `xblue10 ()`.

`\providecolorset` [*<type>*] {*<liste-modèle>*} {*<tête>*} {*<queue>*} {*<ensemble-spéc>*}

Cette commande, similaire à `\definecolorset`, se base sur `\providecolor`; ainsi les couleurs individuelles ne sont définies que si elles n'existent pas déjà.

2.5.4 Définitions immédiates et différées

Traditionnellement, la définition d'une couleur comme décrit ci-dessus conduit à la construction immédiate d'une commande contenant au moins l'information nécessaire au pilote pour afficher la couleur souhaitée. Ainsi, définir par exemple 300 couleurs en chargeant un large ensemble de couleurs prédéfinies va créer 300 nouvelles commandes, bien que la plupart d'entre elles — sauf dans des documents listant délibérément les couleurs — ne soient pas utilisées. Avec le développement de la mémoire des ordinateurs — augmentation en taille, diminution du prix — les récentes implémentations de T_EX ont augmenté leur capacité de mémoire interne disponible pour les chaînes, commandes, etc. Cependant, la mémoire restant finie, il peut être toujours utile (ou occasionnellement nécessaire) de disposer d'une méthode permettant de réduire un peu les besoins de mémoire. C'est ici qu'intervient la *définition de couleur différée*. Son principe est simple : pour chaque définition de ce type (par exemple avec `\preparecolor`), tout l'information nécessaire est sauvée dans une *pile de définitions* globale dédiée où elle peut être récupérée par la suite (par exemple avec `\definecolors`) afin de construire la commande souhaitée.

Notez que les commandes suivantes doivent être utilisées uniquement dans le préambule du document car la pile de définitions de couleur pour les définitions différées est supprimée au début du corps du document — afin d'économiser de la mémoire.

`\preparecolor` [*<type>*] {*<nom>*} {*<liste-modèle>*} {*<liste-spéc>*}

Similaire à `\definecolor`, cette commande ne définit pas cependant la couleur *<nom>* : les arguments *<liste-modèle>* et *<liste-spéc>* sont évalués immédiatement puis tous les paramètres nécessaires (*<type>*, *<nom>*, *<modèle>* et *<spéc>*) sont mis sur la *pile de définitions* pour un usage ultérieur.

`\preparecolorset` [*<type>*] {*<liste-modèle>*} {*<tête>*} {*<queue>*} {*<ensemble-spéc>*}

`\ifdefinecolors` Cette commande est similaire à `\definecolorset` mais dépend de la bascule

¹⁴. En fait, xcolor utilise une variante plus complexe pour fournir les couleurs de base pour les différents modèles sous-jacents (voir le code source pour observer la commande intégrale) :

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

`\ifdefinecolors` : si elle vaut « true », la commande `\definecolor` est appliquée à chaque élément de l'ensemble (ce qui revient à une définition immédiate); sinon la commande `\preparecolor` est appliquée (ce qui donne une définition différée). Par exemple, l'option d'extension `svgnames` réalise quelque chose comme `\definecolorstrue\preparecolorset`, tandis que `svgnames*` agit comme `\definecolorsfalse\preparecolorset`. Les deux options imposent à leur fin `\definecolorstrue`, de façon à ce que les autres commandes disposent d'une situation initiale propre.

`\DefineNamedColor` $\{\langle type \rangle\}\{\langle nom \rangle\}\{\langle liste-modèle \rangle\}\{\langle liste-spéc \rangle\}$ Cette commande est principalement fournie pour des raisons de compatibilité, particulièrement pour permettre le support de couleurs prédéfinies dans `dvipsnam.def`. Elle est équivalente à $\langle commande \rangle[\langle type \rangle]\{\langle nom \rangle\}\{\langle modèle \rangle\}\{\langle spéc \rangle\}$, où $\langle commande \rangle$ est soit `\definecolor` soit `\preparecolor`, selon l'état de `\ifdefinecolors`. Notez que les restrictions de `color` pour l'utilisation de `\DefineNamedColor` dans le seul préambule du document ont été abolies dans `xcolor`.

`\definecolors` $\{\langle list-id \rangle\}$
Il faut ici se rappeler que $\langle liste-id \rangle$ est de la forme $\langle id-étendu \rangle_1, \dots, \langle id-étendu \rangle_l$ où chaque $\langle id-étendu \rangle_\lambda$ est soit un identifiant $\langle id \rangle_\lambda$ ou une équivalence $\langle id \rangle_{\lambda'} = \langle id \rangle_\lambda$. Le premier cas est considéré comme un raccourci pour $\langle id \rangle_\lambda = \langle id \rangle_\lambda$, ce qui amène à la description générale suivante : la pile de définitions est parcourue pour trouver le nom $\langle id \rangle_\lambda$ et ses paramètres de couleur associés; s'il n'y a pas de correspondance, rien ne se passe; si le nom $\langle id \rangle_\lambda$ est dans la pile et que ses paramètres de couleur sont $\langle type \rangle_\lambda$, $\langle modèle \rangle_\lambda$, et $\langle spéc \rangle_\lambda$, alors la commande `\definecolor` $[\langle type \rangle_\lambda]\{\langle id \rangle_\lambda\}\{\langle modèle \rangle_\lambda\}\{\langle spéc \rangle_\lambda\}$ est exécutée. Ainsi, l'utilisateur peut contrôler sous quels noms les couleurs *préparées* peuvent être appelées dans le document. Notez que l'entrée $\langle id \rangle_\lambda$ n'est pas retranchée de la pile de façon à ce qu'elle puisse être utilisée plusieurs fois (y compris dans une même commande `\definecolors`).

`\providecolors` $\{\langle list-id \rangle\}$
Similaire à `\definecolors`, cette commande se base sur `\providecolor` ce qui fait que les couleurs sont définies si elles n'existent pas déjà.

2.5.5 Définitions de couleur globales

`\ifglobalcolors` Par défaut, les définitions faites avec `\definecolor`, `\providecolor`, ... sont seulement disponibles dans le groupe courant. En utilisant `\globalcolorstrue`, toutes ces définitions deviennent disponibles globalement — jusqu'à ce que le groupe courant prenne fin¹⁵. Une autre méthode pour indiquer qu'une définition de couleur individuelle doit être globale revient à la préfixer avec `\xglobal`, soit, par exemple, `\xglobal\definecolor{toto}...`

¹⁵. Cette bascule peut être placée aussi dans le préambule pour contrôler le document dans son ensemble.

2.6 Utilisation de la couleur

2.6.1 Commandes de couleur standards

Voici la liste des commandes appliquant les couleurs qui se retrouvent dans l'extension `coloret` qui bénéficient ici de la syntaxe étendue pour les couleurs comme vu ci-dessus :

`\color` $\{\langle\text{couleur}\rangle\}$
 $[\langle\text{liste-modèle}\rangle][\langle\text{liste-spéc}\rangle]$

Cette commande fait passer à la couleur donnée soit par nom/expression, soit par modèle/spécification. La couleur restera active jusqu'à la fin du groupe courant.

`\textcolor` $\{\langle\text{couleur}\rangle\}\{\langle\text{texte}\rangle\}$
 $[\langle\text{liste-modèle}\rangle][\langle\text{liste-spéc}\rangle]\{\langle\text{texte}\rangle\}$

La commande est ici juste une syntaxe alternative à `\color`, cette syntaxe précisant le groupe de façon explicite. Ainsi, le $\langle\text{texte}\rangle$ apparaît dans la couleur spécifiée puis la couleur reprend sa valeur précédente. Par ailleurs, elle fait appel à `\leavevmode` pour garantir le démarrage du mode horizontal.

`\pagecolor` $\{\langle\text{couleur}\rangle\}$
 $[\langle\text{liste-modèle}\rangle][\langle\text{liste-spéc}\rangle]$

Cette commande spécifie la couleur de fond pour la page courante comme les suivantes. Il s'agit d'une déclaration globale qui ne tient pas compte des groupes \TeX .

Note : toutes ces commandes, à l'exception de `\color` demandent à ce que les arguments $\langle\text{color}\rangle$ ou $\langle\text{spéc}\rangle$ soient mis dans accolades $\{\}$, même s'ils sont enfouis dans les commandes.

Par exemple, une fois posé `\def\toto{red}`, il est possible d'écrire `\color\toto` dans le document mais il vaut mieux toujours écrire `\textcolor{\toto}{truc}` au lieu de `\textcolor{foo}{truc}` pour éviter des résultats étranges.

Notez que les commandes dédiées aux couleurs tirées d'autres extensions peuvent avoir des résultats inattendus si elles sont directement confrontées à des expressions de couleur (par exemple, `\sethlcolor` et similaires issues de l'extension `soul`). Cependant, il est possible de passer l'expression en un nom par le biais de `\colorlet` et d'essayer d'utiliser ce nom à la place.

`\nopagecolor` Contrairement à `\pagecolor`, cette commande retire la couleur de fond de page en restaurant le fond transparent usuel. Elle n'est pas supportée par toutes les options de pilote et génère donc une alerte s'il n'existe pas de définition dans le fichier du pilote.

2.6.2 Boîtes colorées

`\colorbox` $\{\langle\text{couleur}\rangle\}\{\langle\text{texte}\rangle\}$
 $[\langle\text{liste-modèle}\rangle][\langle\text{liste-spéc}\rangle]\{\langle\text{texte}\rangle\}$

La commande `\colorbox` prend les mêmes arguments que `\textcolor`, mais l'indication de couleur sert au *fond* de la boîte.

`\fcolorbox` $\{\langle\text{couleur-cadre}\rangle\}\{\langle\text{couleur-fond}\rangle\}\{\langle\text{texte}\rangle\}$
 $[\langle\text{liste-modèle}\rangle][\langle\text{liste-spéc-cadre}\rangle][\langle\text{liste-spéc-fond}\rangle]\{\langle\text{texte}\rangle\}$
 $[\langle\text{liste-modèle-cadre}\rangle][\langle\text{liste-spéc-cadre}\rangle][\langle\text{liste-modèle-fond}\rangle][\langle\text{liste-spéc-fond}\rangle]\{\langle\text{texte}\rangle\}$

`{\couleur-cadre}{[liste-modèle-fond]}{\liste-spéc-fond}{\texte}`

Cette commande place un cadre de la première couleur autour d'une boîte dont le fond est de la seconde couleur. Si seul le premier argument optionnel est donné, il spécifie le modèle de couleur pour les deux couleurs. Au-delà de la possibilité de spécifier des *expressions de couleur* comme arguments, `\fcolorbox` offre maintenant plus de flexibilité pour ses arguments que sa version de l'extension `color` :

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

Qui est plus, `\fcolorbox` recourt maintenant à une nouvelle manière de d'encadrer un dessin, un développement de la suggestion de Donald Arseneau dans le rapport d'erreur latex/3655 [2]. La principale différence avec l'implémentation de \LaTeX est que la construction de boîte et le tracé du cadre sont divisés en opérations distinctes, de telle façon à ce que le cadre soit tracé *après* que contenu de boîte soit construit. Ceci garantit que la cadre est toujours au-dessus de la boîte. Donald Arseneau a amélioré la rapidité ainsi que les besoins en mémoire de cette approche. Par ailleurs, une nouvelle commande est introduite :

`\boxframe` `{\largeur}{\hauteur}{\profondeur}`

Elle trace un cadre avec une épaisseur de trait de `\boxrule`. La boîte `\hbox` qu'elle restitue présente les dimensions extérieures `\largeur`, `\hauteur` et `\profondeur`. Avec cette approche, une primitive de cadre peut également être fournie par un fichier de pilote afin d'exploiter les spécificités de tracé des pilotes (voir ci-dessous). Là encore, la commande a été optimisée par Donald Arseneau.

La nouvelle approche de l'encadrement est utilisée pour `\fcolorbox` que pour les commandes `\fbox` et `\framebox` de \LaTeX , à moins que l'option `kernel\fbbox` ne soit spécifiée, ce qui restitue alors les définitions originales de \LaTeX pour `\framebox`. L'option `xcdraw` utilise les commandes PostScript pour tracer les cadres et boîtes colorées en cas de choix du pilote `dvips` et elle utilise le code PDF pour tracer les cadres en cas de choix des pilotes `pdftex` et `dvipdfm`. Ceci reste un code expérimental qui peut perturber les visualisateurs `.dvi`. L'option opposée, `noxcdraw`, impose l'utilisation d'un code générique (indépendant du pilote).

2.6.3 Utilisation de la couleur courante

Dans une expression de couleur, « . » sert de synonyme pour la couleur courante. Voir la figure 7 page 39 pour un exemple.

Il est également possible de sauvegarder la couleur courante pour un usage ultérieur, par exemple, par le biais de la commande `\colorlet{foo}{.}`.

Notez que, dans certains cas, la couleur courante est d'un intérêt relativement limité. Par exemple, la construction d'une boîte `\fcolorbox` implique qu'au moment où la *couleur-fond* est évaluée, la couleur courante est celle de *couleur-cadre* ; dans ce cas, « . » ne fait pas référence à la couleur courante à l'*extérieur* de la boîte.

2.6.4 Test de couleurs

`testcolors` [*modèles-numériques*]

Il s'agit ici d'un simple environnement de table (`tabular`) utilisé pour afficher des couleurs dans différents modèles et montrant à la fois le résultat visuel et les paramètres spécifiques au modèle. L'argument optionnel *modèles-numériques* est une liste de modèles de couleur *numériques* (comme d'habitude sans espaces) séparés par des virgules et qui forment les colonnes de la table ; la liste par défaut est `rgb,cmyk,hsb,HTML`.

`\testcolor` {*couleur*} [*liste-modèle*] {*liste-spéc*}

Chaque commande `\testcolor` génère une rangée de la table, contenant un exemple d'affichage et les paramètres propres à chaque modèle. Si le modèle associé à la colonne correspond au modèle de la couleur en question, les paramètres sont alors soulignés. Notez que cette commande n'est disponible que dans l'environnement `testcolors`.

Pour des exemples, voir la figure 2 page 37 ainsi que les figures 11 et 12.

2.7 Glissement de couleur

L'objectif du *glissement de couleur* est d'ajouter une expression de mélange de couleur à toutes les couleurs explicites qui suivent. Il devient ainsi possible d'effectuer une même opération de mélange pour plusieurs couleurs sans toucher à leur définition individuelle.

`\blendcolors` {*expr de mélange*}

`\blendcolors*` {*expr de mélange*}

Cette commande initialise tous les paramètres nécessaires pour le glissement de couleur. L'expression complète de glissement de couleur à proprement parler est stockée dans `\colorblend`. Dans la version étoilée, l'argument sera ajouté à l'expression de glissement existante. Un argument *expr de mélange* vide annule l'effet du glissement à sa suite.

Exemple : après `\blendcolors{!50!yellow}`, les couleurs  sont transformées en  et après un `\blendcolors*{!50}` complémentaire en . Afin d'obtenir un effet global, `\blendcolors` peut être préfixé avec `\xglobal`.

`\xglobal`

Remarque : le glissement de couleur peut être appliqué uniquement à des commandes de couleur *explicites*, autrement dit `\color`, `\fcolorbox` et similaires. Dans l'exemple précédent, les cadres ne sont pas touchés par le glissement car leur couleur est fixée par une commande interne au pilote (revenant à la « couleur courante »). Aussi, pour influencer également ces couleurs *implicites*, il faut fixer la couleur courante *après* le glissement : `\blendcolors{!50!yellow}\color{black}` conduit à , et un `\blendcolors*{!50}\color{black}` complémentaire à .

2.8 Masques de couleur et séparation

Le but de la *séparation de couleur* est de représenter toutes les couleurs qui apparaissent dans un document comme une combinaison d'un sous-ensemble fini de couleurs de base et de leurs variantes assourdies. La séparation la plus importante est **cm_yk** où les couleurs de base sont le cyan (*cyan*), le magenta (*magenta*), le jaune (*yellow*) et le noir (*black*), comme demandé par les imprimeurs. Ceci peut se faire en choisissant l'option d'extension **cm_yk**, de telle manière à ce que toutes les couleurs soient converties dans ce modèle, et de retraiter alors le fichier de sortie pour en extraire les compositions des couleurs. ✖We describe now another — and more general — solution : *color masking*. How does it work? Color masking is based on a specified color model $\langle m\text{-}model \rangle$ and a parameter vector $\langle m\text{-}spec \rangle$. Whenever a color is to be displayed in the document, it will first be converted to $\langle m\text{-}model \rangle$, afterwards each component of the resulting color vector will be multiplied by the corresponding component of $\langle m\text{-}spec \rangle$. For example, let's assume that $\langle m\text{-}model \rangle$ equals **cm_yk**, and $\langle m\text{-}spec \rangle$ equals $(\mu_c, \mu_m, \mu_y, \mu_k)$. Then an arbitrary color foo will be transformed according to

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

Obviously, color separation is a special case of masking by the vectors $(1, 0, 0, 0)$, $(0, 1, 0, 0)$, etc. An interesting application is to shade or tint all colors by masking them with (x, x, x) in the **rgb** or **cm_yk** model, see the last two rows in figure 9 page 41.

<code>\maskcolors</code>	<code>[\langle num model \rangle]{\langle color \rangle}</code> 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, <code>\maskcolorstrue</code> is performed. Color masking can be switched off temporarily by <code>\maskcolorsfalse</code> , or — in a more radical way — by <code>\maskcolors{}</code> , which in addition clears the initialisation parameters. In general, the scope of <code>\maskcolors</code> is the current group (unless it is prefixed by the <code>\xglobal</code> command), but it may be used in the document preamble as well. The final remark of the color blending section applies here similarly.
<code>\ifmaskcolors</code>	Now it is easy to separate a complete document without touching the source code : <code>\latex \def\xcolorcmd{\maskcolors[cm_yk]{cyan}}\input{a}</code> will do the (<i>cyan</i>) part of the job for <code>a.tex</code> .
<code>\xglobal</code>	
<code>\colormask</code>	Caution : xcolor has no idea about colors in files that are included via the command <code>\includegraphics</code> , e.g., images of type <code>.eps</code> , <code>.pdf</code> , <code>.jpg</code> , or <code>.png</code> . Such files have to be separated separately. Nevertheless, xcolor offers some basic support by storing the mask color in <code>\colormask</code> , which can be used to decide which file is to be included :
	<pre> \def\temp{cyan}\ifx\colormask\temp \includegraphics{foo_c}\else \def\temp{magenta}\ifx\colormask\temp \includegraphics{foo_m}\else ... \fi\fi </pre>

2.9 Séries de couleurs

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 Définition d'une série de couleurs

✕

`\definecolorseries` $\{\langle name \rangle\}\{\langle core\ model \rangle\}\{\langle method \rangle\}[\langle b\text{-}model \rangle]\{\langle b\text{-}spec \rangle\}[\langle s\text{-}model \rangle]\{\langle s\text{-}spec \rangle\}$
 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 :

- $[\langle b\text{-}model \rangle]\{\langle b\text{-}spec \rangle\}$ 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.
- $[\langle s\text{-}model \rangle]\{\langle s\text{-}spec \rangle\}$ 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 $\langle s\text{-}spec \rangle$ 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 d’une série de couleurs

✖

`\resetcolorseries` [*div*]{*name*}

This command has to be applied at least once, in order to make use of the color series *name*. It resets the current color of the series to the base color and calculates the actual step vector according to the chosen *div*, a non-zero real number, for the methods `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{div}`, applied *before* the extension `xcolor` is loaded (similar to `\rangeRGB` and friends). The optional argument is ignored in case of the `step` method.

✖

2.9.3 Utilisation d’une série de couleurs

✖

There are two ways to display the current color of a color series : any of the *color expressions* in section 2.3 page 14 used within a `\color`, `\textcolor`, ... command will display this color according to the usual syntax of such expressions. However, in the cases when *postfix* equals ‘`!!+`’, `\color{name}!!+` 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{name}!![num]` enables direct access to an element of a series, where *num* = 0, 1, 2, ..., starting with 0 for the base color. See figure 8 page 40 for a demonstration of different methods.

✖

2.9.4 Différences entre couleurs et séries de couleurs

✖

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 Couleur d'encadrement d'hyperliens

✕

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 Spécifications de couleurs additionnelles le monde de pstricks

✖

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 liste-modèle>}{<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 Couleur dans des tableaux

✖

`\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 41. 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 40.

✖

2.13 Information sur la couleur

✖

`\extractcolorspec` $\{\langle color \rangle\}\{\langle cmd \rangle\}$
 Extracts the color specification of $\langle color \rangle$ and puts it into $\langle cmd \rangle$; equivalent to `\def\cmd{\{\langle model \rangle\}\{\langle spec \rangle\}}`.

`\extractcolorspecs` $\{\langle color \rangle\}\{\langle model-cmd \rangle\}\{\langle color-cmd \rangle\}$
 Extracts the color specification of $\langle color \rangle$ and puts it into $\langle model-cmd \rangle$ and $\langle color-cmd \rangle$, respectively.

`\tracingcolors` $=\langle int \rangle$

Controls the amount of information that is written into the log file :

- $\langle int \rangle \leq 0$: no specific color logging.
- $\langle int \rangle \geq 1$: ignored color definitions due to `\providecolor` 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 TeX's `\tracing...` commands, this command may be used globally (in the document preamble) or locally/block-wise. The package sets `\tracingcolors=0` as default. Remark : since registers are limited and valuable, no counter is wasted for this issue.

Note that whenever a color is used that has been defined via `color`'s `\definecolor` command rather than `xcolor`'s new `\definecolor` and friends, a warning message 'Incompatible color definition' will be issued.¹⁶

✖

2.14 Conversion de couleur

✖

`\convertcolorspec` $\{\langle model \rangle\}\{\langle spec \rangle\}\{\langle target model \rangle\}\{\langle cmd \rangle\}$

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 : `\convertcolorspec{cmyk}{0.81,1,0,0.07}{HTML}\tmp` acts like `\def\tmp{1F00ED}`.



✖

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

2.15 Problèmes et 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 :

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

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

2.15.2 Page breaks and pdfTeX

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

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

This works as expected with *dvips*, i.e., ‘red1’ and ‘red2’ being *(red)*, however, with *pdftex*, ‘red2’ is displayed in *(black)*. The problem may be solved by using the `fixpdftex` option which simply loads Heiko Oberdiek’s `pdfcolmk` package [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 Exemples

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

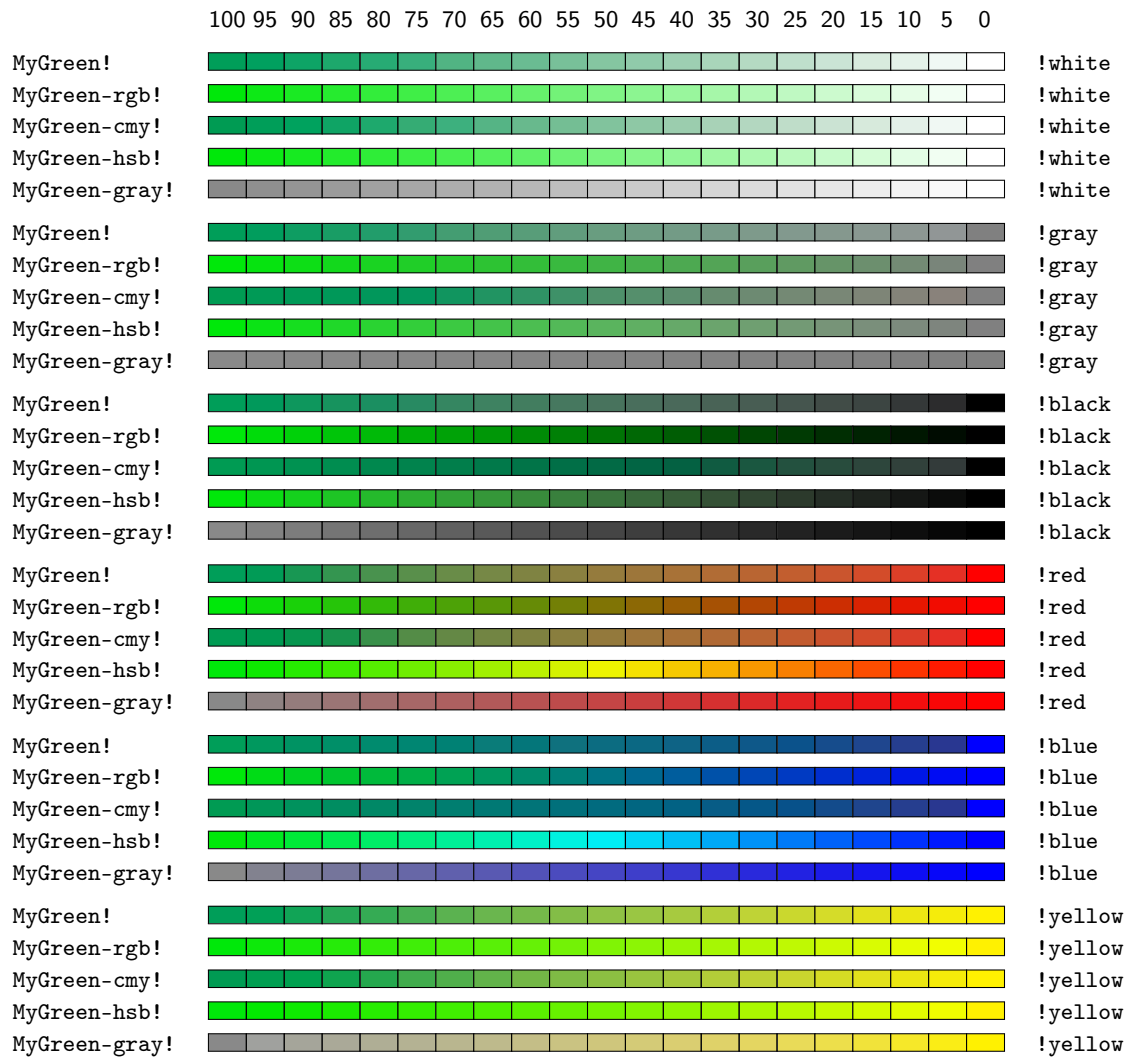
color	rgb	cmYk	hsb	HTML	gray
olive	 0.5 0.5 0	 <u>0 0 1 0.5</u>	 0.16667 1 0.5	 808000	 0.39
red!50!green	 <u>0.5 0.5 0</u>	 0 0 0.5 0.5	 0.16667 1 0.5	 808000	 0.445
-cyan!50!magenta	 0.5 0.5 0	 <u>0 0 0.5 0.5</u>	 0.16667 1 0.5	 808000	 0.445
[cmYk]0,0,1,0.5	 0.5 0.5 0	 <u>0 0 1 0.5</u>	 0.16667 1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

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

FIGURE 5 – Standard color expressions

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

FIGURE 6 – Standard color expressions

```

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

```



FIGURE 7 – Current color

```

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

```

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

FIGURE 8 – Color series

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

Individual definitions

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

Common definitions

```

\resetcolorseries[12]{test}
\rowcolors[\hline]{1}{test!!+}{test!!+}
\begin{tabular}{c}
\number\rownum\ \number\rownum\ \number\rownum\ \number\rownum\
\number\rownum\ \number\rownum\ \number\rownum\ \number\rownum\
\number\rownum\ \number\rownum\ \number\rownum\ \number\rownum\
\number\rownum\ \number\rownum\ \number\rownum\ \number\rownum\
\end{tabular}

```


FIGURE 9 – Color masking

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

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

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

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















































































































































































































































































































































































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

FIGURE 12 – Color harmony





































































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

4 Colors by Name

4.1 Base colors (always available)

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

4.2 Colors via dvipsnames option

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

4.3 Colors via svgnames option

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

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
















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

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

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

4.4 Colors via x11names option

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

 Honeydew1 ( LightSteelBlue3 ( PaleVioletRed1 ( SlateBlue3 (
 Honeydew2 ( LightSteelBlue4 ( PaleVioletRed2 ( SlateBlue4 (
 Honeydew3 ( LightYellow1 ( PaleVioletRed3 ( SlateGray1 (
 Honeydew4 ( LightYellow2 ( PaleVioletRed4 ( SlateGray2 (
 HotPink1 ( LightYellow3 ( PeachPuff1 ( SlateGray3 (
 HotPink2 ( LightYellow4 ( PeachPuff2 ( SlateGray4 (
 HotPink3 ( Magenta1 ( PeachPuff3 ( Snow1 (
 HotPink4 ( Magenta2 ( PeachPuff4 ( Snow2 (
 IndianRed1 ( Magenta3 ( Pink1 ( Snow3 (
 IndianRed2 ( Magenta4 ( Pink2 ( Snow4 (
 IndianRed3 ( Maroon1 ( Pink3 ( SpringGreen1 (
 IndianRed4 ( Maroon2 ( Pink4 ( SpringGreen2 (
 Ivory1 ( Maroon3 ( Plum1 ( SpringGreen3 (
 Ivory2 ( Maroon4 ( Plum2 ( SpringGreen4 (
 Ivory3 ( MediumOrchid1 ( Plum3 ( SteelBlue1 (
 Ivory4 ( MediumOrchid2 ( Plum4 ( SteelBlue2 (
 Khaki1 ( MediumOrchid3 ( Purple1 ( SteelBlue3 (
 Khaki2 ( MediumOrchid4 ( Purple2 ( SteelBlue4 (
 Khaki3 ( MediumPurple1 ( Purple3 ( Tan1 (
 Khaki4 ( MediumPurple2 ( Purple4 ( Tan2 (
 LavenderBlush1 ( MediumPurple3 ( Red1 ( Tan3 (
 LavenderBlush2 ( MediumPurple4 ( Red2 ( Tan4 (
 LavenderBlush3 ( MistyRose1 ( Red3 ( Thistle1 (
 LavenderBlush4 ( MistyRose2 ( Red4 ( Thistle2 (
 LemonChiffon1 ( MistyRose3 ( RosyBrown1 ( Thistle3 (
 LemonChiffon2 ( MistyRose4 ( RosyBrown2 ( Thistle4 (
 LemonChiffon3 ( NavajoWhite1 ( RosyBrown3 ( Tomato1 (
 LemonChiffon4 ( NavajoWhite2 ( RosyBrown4 ( Tomato2 (
 LightBlue1 ( NavajoWhite3 ( RoyalBlue1 ( Tomato3 (
 LightBlue2 ( NavajoWhite4 ( RoyalBlue2 ( Tomato4 (
 LightBlue3 ( OliveDrab1 ( RoyalBlue3 ( Turquoise1 (
 LightBlue4 ( OliveDrab2 ( RoyalBlue4 ( Turquoise2 (
 LightCyan1 ( OliveDrab3 ( Salmon1 ( Turquoise3 (
 LightCyan2 ( OliveDrab4 ( Salmon2 ( Turquoise4 (
 LightCyan3 ( Orange1 ( Salmon3 ( VioletRed1 (
 LightCyan4 ( Orange2 ( Salmon4 ( VioletRed2 (
 LightGoldenrod1 ( Orange3 ( SeaGreen1 ( VioletRed3 (
 LightGoldenrod2 ( Orange4 ( SeaGreen2 ( VioletRed4 (
 LightGoldenrod3 ( OrangeRed1 ( SeaGreen3 ( Wheat1 (
 LightGoldenrod4 ( OrangeRed2 ( SeaGreen4 ( Wheat2 (
 LightPink1 ( OrangeRed3 ( Seashell1 ( Wheat3 (
 LightPink2 ( OrangeRed4 ( Seashell2 ( Wheat4 (
 LightPink3 ( Orchid1 ( Seashell3 ( Yellow1 (
 LightPink4 ( Orchid2 ( Seashell4 ( Yellow2 (
 LightSalmon1 ( Orchid3 ( Sienna1 ( Yellow3 (
 LightSalmon2 ( Orchid4 ( Sienna2 ( Yellow4 (
 LightSalmon3 ( PaleGreen1 ( Sienna3 ( Gray0 (
 LightSalmon4 ( PaleGreen2 ( Sienna4 ( Green0 (
 LightSkyBlue1 ( PaleGreen3 ( SkyBlue1 ( Grey0 (
 LightSkyBlue2 ( PaleGreen4 ( SkyBlue2 ( Maroon0 (
 LightSkyBlue3 ( PaleTurquoise1 ( SkyBlue3 ( Purple0 (
 LightSkyBlue4 ( PaleTurquoise2 ( SkyBlue4 (
 LightSteelBlue1 ( PaleTurquoise3 ( SlateBlue1 (
 LightSteelBlue2 ( PaleTurquoise4 ( SlateBlue2 (

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	my	hsb	gray	RGB	HTML	HSB	Gray
dvipdf	2015/12/30 v3.0k	d	n	d	n	d	i	n	n	n
dvips	2015/12/30 v3.0k	d	n	d	d	d	i	n	n	n
dvipsone	2015/12/30 v3.0k	d	n	d	d	d	i	n	n	n
pctex32	2015/12/30 v3.0k	d	n	d	d	d	i	n	n	n
pctexps	2015/12/30 v3.0k	d	n	d	d	d	i	n	n	n
pdftex	2011/05/27 v0.06d	d	n	d	n	d	i	n	n	n
luatex	2016/01/23 v0.01b	d	n	d	n	d	i	n	n	n
dvipdfm	1999/9/6 vx.x	d	n	d	n	d	i	n	n	n
dvipdfmx	2016/04/06 v4.08	d	n	d	?	d	i	n	n	n
textures	1997/5/28 v0.3	d	n	d	?	i	n	n	n	n
vtex	1999/01/14 v6.3	d	n	d	n	i	i	n	n	n
xetex	2016/04/06 v4.08	d	n	d	d	d	i	n	n	n
tcidvi	2015/12/30 v3.0k	i	n	i	n	i	d	n	n	n
truettex	2015/12/30 v3.0k	i	n	i	n	i	d	n	n	n
dviwin	2015/12/30 v3.0k	n	n	n	n	n	n	n	n	n
emtex	2015/12/30 v3.0k	n	n	n	n	n	n	n	n	n
pctexhp	2015/12/30 v3.0k	n	n	n	n	n	n	n	n	n
pctexwin	2015/12/30 v3.0k	n	n	n	n	n	n	n	n	n
dviwindo = dvipsone; oztex = dvips; xdvi = dvips + monochrome										
Driver's color model support : d = direct, i = indirect, n = none										

5.2 How xcolor handles driver-specific color models

Although there is a variety of drivers that implement different approaches to color visualisation, they all have some features in common, as defined by the original 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`¹⁷ 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`¹⁸ 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}{...}{...}` generates¹⁹ 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 38 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.²⁰ 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 fixed-point arithmetic framework, providing at most 5 decimal digits via \TeX 's dimension registers.

17. The double backslash is intentional.

18. The single backslash is intentional.

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

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

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>		

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

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

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

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

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 14, 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 51 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 52.

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

Conversion rgb to HTML As described in section 6.2.1 page 52. 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 54.

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

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

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

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

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

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

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

Here, four additional functions are required :

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

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

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

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

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

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

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

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

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

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

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

6.3.3 The `cm`yk model

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

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

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

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

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

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

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

6.3.4 The hsb model

Conversion hsb to rgb

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

with

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

and

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

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

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

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

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

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

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

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

it is clear that there is only one way to define F for $i = 6$ in order to get a continuous function, as shown in equation (67). This has been transformed back to equation (71). A similar argument shows that F indeed is a continuous function of hue over the whole range $[0, 1]$.

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

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

Conversion hsb to HSB As described in section 6.2.1 page 52.

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

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 52.

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

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 52.

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

6.3.9 The HTML model

Conversion HTML to rgb As described in section 6.2.2 page 52 : 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 52.

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

6.3.11 The Gray model

Conversion Gray to gray As described in section 6.2.2 page 52.

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

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 ». \LaTeX 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 », 2014.
CTAN/macros/latex/required/graphics/grfguide.*
- [6] David P. Carlisle : extension color, “2016/01/03 v1.1b Standard LaTeX Color (DPC)”. 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 \LaTeX colors ». CTAN/info/colour/chroma/ et <http://www.ukern.de/tex/chroma.html>
- [10] Uwe Kern : extension xcolor, « \LaTeX 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 MiKTeX : <http://www.miktex.org/>
- [14] Sebastian Rahtz, Heiko Oberdiek : extension hyperref, « 2006/09/06 v6.75e Hypertext links for \LaTeX ». CTAN/macros/latex/contrib/hyperref/
- [15] Alvy Ray Smith : « Color Gamut Transform Pairs ». *Computer Graphics* (ACM SIGGRAPH), Volume 12, 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

11/05/2016 v2.12

- Nouvelles fonctionnalités :
 - `\nopagecolor` command as introduced in `color v1.1a` (example added to `xcolor3.tex`);
 - `luatex` driver option (code provided by DPC) to fix incompatibilities due to changes in new Lua_TE_X version.
- Corrections d’erreur :
 - possible name conflict by `\XC@ifxcase` call;
 - incorrect internal `\@hex@@Hex` macro.

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 :
 - l’option `fixinclude` empêche *dvips* de basculer explicitement la couleur courante au noir (*black*) avant d’insérer un fichier `.eps` par le biais de `\color{red}\includegraphics{test}`.
- Changements :
 - `\colorbox` et `\fcolorbox` sont rendues robustes;
 - l’option d’extension obsolète `pst` est retirée;
 - plusieurs changements aux commandes internes.
- Corrections d’erreur :
 - traitement incorrect de la couleur courante « . » de type **cm**yk.

21/12/2005 v2.09

- Nouvelles fonctionnalités :
 - `\definecolor` et `\color` acceptent maintenant des spécifications de couleur avec comme séparateur l’espace, par exemple : `\color[rgb]{1 .5 0}`;
 - l’option expérimentale `xcdraw` est étendue aux pilotes `pdftex` et `dvipdfm`.
- Changements :
 - le fichier de test `xcolor2.tex` est rendu compatible avec les changements récents dans `pstricks`;

- le fichier de test `xcolor3.tex` est étendu ;
- le fichier de test de pilote `xcolor4.tex` est étendu pour illustrer les différentes approches de tracés de cadres ;
- implémentation plus efficace du code spécifique aux pilotes.

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 pdfTeX'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 :
 - `\aftergroupedef` command to reproduce `\aftergroupdef`'s behaviour prior to v2.01;
 - `xcolor`'s homepage www.ukern.de/tex/xcolor.html now provides also a ready-to-run TDS-compliant archive containing all required files.
- 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 :
 - le modèle de couleur **HTML**, une variante **RGB** en hexadécimal 24-bit; ceci permet de spécifier des couleurs comme `\color[HTML]{AFFE90}`;
 - les noms de couleur orange (*orange*), violet (*violet*), pourpre (*purple*) et brun (*brown*) sont ajoutés à l'ensemble des couleurs prédéfinies.
- Nouvelle page web de `xcolor` : www.ukern.de/tex/xcolor.html
- Correction d'erreur : `\xdefinecolor` ne normalisait parfois pas ses paramètres.

- Changements :
 - légères améliorations de la documentation ;
 - le fichier d'exemple `xcolor1.tex` est réorganisé et abrégé.

04/02/2004 v1.08

- Commandes nouvelles :
 - `\selectcolormodel` pour changer le modèle cible au sein d'un document ;
 - `\adjustUCRBG` pour contrôler finement le **✖ undercolor-removal ✖** et le **✖ black-generation ✖** durant la conversion en **cmk**.
- Correction d'erreur : les expressions de couleur ne fonctionnaient pas correctement avec le caractère actif « ! », par exemple, en cas d'utilisation de `\usepackage[frenchb]{babel}`.
- Réorganisation du code :
 - `\XC@xdefinecolor` est intégré à `\xdefinecolor`, rendant la première commande obsolète ;
 - plusieurs commandes internes améliorées/rationalisées.

20/01/2004 v1.07

- Nouvelle fonctionnalité : support de **✖ color masking ✖** et de **✖ color separation ✖**.
- Nouvelles commandes :
 - `\rmultiply` pour multiplier un registre de dimension par un nombre réel ;
 - `\xcolorcmd` pour passer des commandes à exécuter à la fin de l'extension.
- Changements :
 - meilleure gestion de la couleur : les couleurs étendues ont la priorité sur les couleurs standards ;
 - plusieurs commandes sont améliorées en utilisant le code du noyau \LaTeX .
- Documentation : quelques changements mineurs.

- Fichiers d'exemples : exemples `pstricks` complémentaires (fichier `xcolor2.tex`).

15/12/2003 v1.06

- Nouvelle fonctionnalité : expressions de couleur étendues, ce qui permet des opérations de mélange en cascade, par exemple `\color{red!30!green!40!blue}`.
- Documentation : nouvelle section sur les expressions de couleur.
- Correction d'erreur : l'incrémentement des séries de couleur ne fonctionnait pas correctement dans les commandes sans affichage telle `\extractcolorspec{foo!+}` (cette erreur a été introduite en v1.05).
- Commandes renommées : `\ukfileversion` et les constantes internes similaires sont renommées en `\XCfileversion` et ainsi de suite.
- Commandes retranchées : `\ifXCpst` et `\ifXCtable` sont rendues obsolètes par une simple astuce.

21/11/2003 v1.05

- Corrections d'erreur :
 - l'option d'extension `hideerrors` fonctionne maintenant comme attendu ;
 - l'utilisation de « . » dans la première expression de couleur causait une erreur du fait d'une initialisation incorrecte.
- Réorganisation du code :
 - `\extractcolorspec` utilise maintenant `\XC@splitcolor`, ce qui rend `\XC@extract` obsolète.

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 ;
 - `\rowcol@l@rs` : remplacement de `\@ifempty` par `\@ifxempty`.

09/09/2003 v1.00

- Première version publiée.

Index

Les numéros en italique renvoient à la page où se trouve l'entrée correspondante; les numéros soulignés renvoient à la ligne de code de la définition; les numéros en romain renvoient aux lignes de code où l'entrée est utilisée.

A		
<code>\adjustUCRBG</code>	10, 57	
arguments		
<code><couleur></code>	15, 18	
<code><dec></code>	15, 16	
<code><div></code>	15, 16	
<code><ent></code>	15, 16	
<code><expr étendue></code>	15, 17	
<code><expr de couleur></code>	15, 18	
<code><expr de mélange></code>	15, 17	
<code><expr fonctionnelle></code>	15, 18	
<code><expr></code>	15, 17	
<code><fonction></code>	15, 18	
<code><id étendu></code>	15	
<code><id></code>	15	
<code><liste-id></code>	15	
<code><liste-modèle></code>	15, 17	
<code><liste-spéc></code>	15, 17	
<code><modèle central></code>	15, 16	
<code><modèle numérique></code>	15, 16	
		<code><modèle></code> 15, 16
		<code><moins></code> 15, 16
		<code><nom></code> 15, 16
		<code><num></code> 15, 16
		<code><pct></code> 15, 16
		<code><plus></code> 15, 16
		<code><préfixe></code> 15, 17
		<code><spéc></code> 15, 16
		<code><suffixe></code> 15, 17
		<code><type></code> 15, 17
		<code><vide></code> 15, 16
B		
<code>\blendcolors</code>	28	
<code>\blendcolors*</code>	28	
<code>\boxframe</code>	27	
C		
clés		
<code>citebordercolor</code>	32	
<code>citecolor</code>	32	
		<code>filebordercolor</code> 32
		<code>filecolor</code> 32
		<code>linkbordercolor</code> 32
		<code>linkcolor</code> 32
		<code>menubordercolor</code> 32
		<code>menucolor</code> 32
		<code>pagebordercolor</code> 32
		<code>pagecolor</code> 32
		<code>pdfborder</code> 32
		<code>runbordercolor</code> 32
		<code>runcolor</code> 32
		<code>urlbordercolor</code> 32
		<code>urlcolor</code> 32
		<code>xcitebordercolor</code> 32
		<code>xfilebordercolor</code> 32
		<code>xlinkbordercolor</code> 32
		<code>xmenubordercolor</code> 32
		<code>xpagebordercolor</code> 32
		<code>xpdfborder</code> 32, 68
		<code>xrunbordercolor</code> 32

<i>xurlbordercolor</i>	32	<code>\extractcolorspecs</code>	34	HTML	11, 12, 16, 47, 51, 56, 61, 69
<code>\color</code>	26			Hsb	12, 13, 16, 20, 42, 51, 59–61, 66
color stack	35	F		RGB	11–13, 16, 20, 21, 47, 51, 56, 61, 69
<code>\colorbox</code>	26	<code>\fcolorbox</code>	26	cmyk	5–7, 10– 12, 16, 20, 23, 28, 47, 51, 53, 56, 57, 61, 65, 70
<code>\colorlet</code>	23	fichiers		cmy	10–12, 16, 29, 47, 51, 54, 56, 57, 60
<code>\colormask</code>	29	<code>.def</code>	9, 68	gray	6, 10–13, 16, 19, 47, 51, 53, 56, 57, 60, 61
<code>\colorseriescycle</code>	31	<code>.dvi</code>	21, 22, 27, 32	hsb	5, 10–14, 16, 20, 23, 47, 48, 50, 51, 53–55, 58–62
<code>\convertcolorspec</code>	34	<code>.eps</code>	11, 29, 35, 36, 65	rgb	6, 7, 10–14, 16, 20, 21, 23, 29, 32, 47, 51, 53–62
couleur éclaircie	6	<code>.jpg</code>	29	tHsb	12, 13, 16, 20, 42, 51, 60, 61, 66
couleur assombrie	6	<code>.pdf</code>	29	wave	12, 13, 16, 51, 62, 66
couleur assourdie	6	<code>.png</code>	29	‘named’	16, 23, 69
		<code>.ps</code>	22, 32, 36	‘ps’	16, 66
D		<code>.sty</code>	68		
<code>\definecolor</code>	23, 33	<code>.xcp</code>	11, 22	N	
<code>\definecolors</code>	25	color.pro	22	noms de couleur	
<code>\definecolorseries</code>	30	dvipsnam.def	25	(black)	28, 35, 62
<code>\definecolorset</code>	23	rgb.txt	21, 66	(blue)	62
<code>\DefineNamedColor</code>	25	xcolor.pro	9, 22, 68	(cyan)	28, 29, 62
		xcolor.sty	9, 20	(Fuchsia)	35
E		xcolor2.tex		(green)	62
ensemble de couleurs	23	33, 63, 65–67, 70	(lime)	65
environnements :		xcolor3.tex	65, 66	(magenta)	28, 62
testcolors	28	xcolor4.tex	66	(olive)	67
expression de expression	23	fonctions de couleur		(pink)	67
extensions		twheel	19, 20	(red)	35, 62, 66
colorinfo	48, 64	wheel	19, 20	(teal)	65
colortbl	11, 12, 33, 64			(yellow)	28, 62
color	5, 6, 9–12, 21, 23, 25–27, 31, 34, 47–49, 64, 65, 69	G		argent (silver)	7
doc	71	<code>\GetGinDriver</code>	9	black (.)	20
fp	48, 68	<code>\GinDriver</code>	9	blanc (white)	6
graphics	21, 35			bleu (blue)	7, 22, 24
graphicx	35	H		blue (.)	20
hyperref	9–12, 32, 64, 65, 71	<code>\hiderowcolors</code>	33	brown (.)	20
longtable	65	HKS	6	brun (brown)	69
memoir	67	HTML4	21, 45	cyan (.)	20
msc	67			darkgray (.)	20
pdfcolmk	11, 12, 35, 64, 67	I		gray (.)	20
preview	67	<code>\ifconvertcolorsD</code>	14	Gray0 ()	21
pstcol	12, 64	<code>\ifconvertcolorsU</code>	14	green (.)	20
pstricks	12, 23, 33, 63, 65–70	<code>\ifdefinecolors</code>	24		
realcalc	48	<code>\ifglobalcolors</code>	25		
soul	26, 69	<code>\ifmaskcolors</code>	29		
url	64				
xcolor	1, 5, 6, 9, 10, 12–14, 16, 22–25, 29, 31, 32, 34, 35, 47–49, 63–65, 67, 69	M			
<code>\extractcolorspec</code>	34	<code>\maskcolors</code>	29		
		MiKTeX	22		
		modèles colorimétriques			
		Gray	11–13, 16, 47, 51, 61		
		HSB	11–13, 16, 47, 51, 59, 61		

Green0 ()	21	gray	9, 11, 14	Kern, Uwe	64
Grey0 ()	21	hideerrors	9, 11, 70	Newton, Isaac	7
gris (<i>gray</i>)	6	hsb	9, 11, 14	Niepraschk, Rolf	64
jaune (<i>yellow</i>)	5, 7, 13, 19	hyperref	9, 11, 32, 69	Oberdiek, Heiko	64
lightgray (.)	20	hypertext	10	Rahtz, Sebastian	64
lime (.)	20	kernelfbox	9, 11, 27, 66	Smith, Alvy Ray	53, 64
magenta (.)	20	luatex	9, 47, 65	Voß, Herbert	64
Maroon0 ()	21	monochrome	9, 47	pile de définitions	24, 25
noir (<i>black</i>)	6, 10, 11, 65	natural	9, 11, 14	PostScript	10, 16, 22,
olive (.)	20	noxdraw	9, 11, 27		27, 33, 35, 53, 55, 66, 68
or (<i>gold</i>)	7	oztex	9, 47	\preparecolor	24
orange (.)	20	pctex32	9, 47	\preparecolorset	24
orange (<i>orange</i>)	69	pctexhp	9, 47	programmes	
pink (.)	20	pctexps	9, 47	Yap	22
pourpre (<i>purple</i>)	69	pctexwin	9, 47	dvipdfm	22
purple (.)	20	pdftex		dvips	9,
Purple0 ()	21		9, 22, 27, 38, 47–49, 65		22, 32, 35, 36, 65, 68, 69
red (.)	20	prologue	9, 11, 22, 67, 68	\providecolor	23
red (<i>rouge</i>)	19	rgb	9, 11, 14	\providecolors	25
rouge (<i>red</i>)		showerrors	9, 11	\providecolorset	24
	5, 7, 10, 13, 22–24	svgnames*	9, 11, 20, 25, 68		
teal (.)	20	svgnames	9, 11,	R	
vert (<i>green</i>)	5, 7, 19, 24		20, 21, 25, 34, 35, 44, 66	\rangeGray	13
violet (.)	20	table	9, 11, 12, 33, 67	\rangeHSB	13
violet (<i>violet</i>)	69	tcidvi	9, 47	\rangeHsb	12, 60
white (.)	20	textures	9, 47	\rangeRGB	13
yellow (.)	20	truetex	9, 47	\rangetHsb	13, 60
\nopagecolor	26	usecolors	48	\resetcolorseries	31
O		vtex	9, 47	\rowcolors	33
options d'extension		x11names*	9, 11, 21	\rowcolors*	33
Gray	9, 11, 14	x11names	9, 11, 21, 45	\rownum	33
HSB	9, 11, 14	xcdraw	9, 11, 27, 65, 66	S	
HTML	9, 11, 14	xdvi	9, 47	\selectcolormodel	14
RGB	9, 11, 14	xetex	9, 47, 66	\showrowcolors	33
cmymk	9, 11, 14, 29	options d'extension (obso-		\substitutecolormodel	14
cmym	9, 11, 14	lètes)		SVG	11, 20, 21, 35, 64, 68
dvipdfmx	9, 47, 66	nodvipsnames	9, 69	T	
dvipdfm	9, 22, 27, 47, 48, 65	override	9, 69, 71	\testcolor	28
dvipdf	9, 47	pst	9, 65	testcolors (environnement)	28
dvipsnames*	9, 11, 20, 68	usenames	9, 49, 69	\textcolor	26
dvipsnames	9–	P		ton direct	6, 7
	11, 20–22, 34, 35, 44, 69	\pagecolor	26	\tracingcolors	34
dvipsone	9, 47	Pantone	6	U	
dvips	9, 10, 22,	PDF	27	Unix	11, 21, 45, 66
	27, 33, 47, 49, 66, 68, 69	personnes		X	
dviwindo	9, 47	Arseneau, Donald	27, 64	X11	11, 21, 45, 66
dviwin	9, 47	Bruton, Dan	62, 64	\xcolorcmd	10
emtex	9, 47	Carlisle, David P.	64, 65	\xglobal	25, 28, 29
fixinclude	9, 11, 36, 65	Goethe, Johann Wolfgang			
fixpdftex	9, 11, 12, 35, 67	von	7		