Élargissement des capacités de LaTeX en matière de couleur : l'extension xcolor

Dr. Uwe Kern

v2.11 (2007/01/21) *

Résumé

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

Table des matières

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

Liste des tableaux

Table des figures

1 Introduction

1.1 Objectif de cette extension

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

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

— Ma société a défini une couleur d'entreprise et le service des impressions m'a dit combien il est coûteux d'utiliser plus de deux couleurs dans notre nouveau brochure, alors même que toutes les teintes de notre couleur (par exemple, une version à 75%) peuvent être utilisées sans aucun surcoût. Comment accéder à ces variations de couleur avec LaTeX ?
(Réponse : `\color{CouleurEntreprise!75}` etc.)

— Mon ami utilise une belle couleur que je souhaiterais appliquer à mes propres documents ; malheureusement, elle est définie avec le modèle **hsb** qui n'est pas accepté par mon application pdfLaTeX favorite. Que faire alors ?
(Réponse : utiliser tout simplement les définitions **hsb**, xcolor fera les calculs nécessaires.)

— Qu'affiche le mélange de 40% de *green* et de 60% de *yellow* ?
(Réponse : 40% + 60% = , soit `\color{green!40!yellow}`)

— Et quelle est l'aspect de sa couleur complémentaire ?
(Réponse : , accessible via `\color{-green!40!yellow}`)

— Maintenant, je souhaite mélanger trois mesures de la dernière couleur avec deux mesures de sa complémentaire et une mesure de *rouge*. Qu'est-ce que cela donne ?
(Réponse : 3× +2× +1× = , cette couleur étant accessible via `\color{rgb:-green!40!yellow,3;green!40!yellow,2;red,1}`)

— Je sais qu'un rayonnement de longueur d'onde de 485nm appartient au spectre visible. Mais quelle couleur a-t-il ?
(Réponse : approximativement , via `\color[wave]{485}`)

— Mon service des impressions souhaite que toutes les définitions de couleur dans mon document soit basculées en modèle **cmyk** (CMJN en français). Comment puis-je faire ces calculs efficacement ?
(Réponse : `\usepackage[cmyk]{xcolor}` ou `\selectcolormodel{cmyk}`)

— J'ai un tableau de 50 lignes. Comment puis-je obtenir des lignes de tableau avec deux couleurs alternées (A pour les lignes paires et B pour les lignes impaires) sans recours à la copie de 50 commandes `\rowcolor` ? Ce motif alterné devrait d'ailleurs commencer à partir de la troisième ligne.

(Réponse : *grosso modo* `\rowcolors{3}{CouleurA}{CouleurB}`)
Ceci répresente quelques uns des problème résolus par l'extension xcolor. Son objectif peut être résumé en la conservation des caractéristiques de color, tout en apportant des fonctionnalités additionnelles et de la flexibilité avec des interfaces simples d'utilisation (avec un peu de chance).

1.2 Teintes, nuances, tons et couleurs complémentaires

Sur la base de [15] nous définissons les termes suivants :
- **teinte** : une couleur à laquelle est ajoutée du *blanc* (white) ;
- **nuance** : une couleur à laquelle est ajoutée du *noir* (black) ;
- **ton** : une couleur à laquelle est ajoutée du *gris* (gray).

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

$$\text{teinte}(C, p) := \text{mélange}(C, \texttt{white}, p) \tag{1}$$
$$\text{nuance}(C, p) := \text{mélange}(C, \texttt{black}, p) \tag{2}$$
$$\text{ton}(C, p) := \text{mélange}(C, \texttt{gray}, p) \tag{3}$$

où `white`, `black`, and `gray` sont des constantes dépendantes des modèles, comme présentées en table 7 page 48. Par la suite, nous définissons le terme :
- **complémentaire** : une couleur C^* qui génère du *blanc* si elle est mélangée avec la couleur d'origine C,

sachant qu'il existe également différents concepts de complémentarité (par exemple des couleurs opposées sur les *cercles chromatiques*). Voir la section 6.3 page 50 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 **cmyk**, dont le but est de représenter de larges ensembles finis ou même infinis (théoriquement) de couleurs, construits sur de très petits sous-ensembles de couleurs de base et de règles permettant de construire d'autres couleurs à partir des couleurs de base. Par exemple, un large ensemble de couleurs peut être construit par combinaison linéaire des couleurs de base *rouge*, *vert* et *bleu*. En contrepartie, un ton direct ne peut souvent être qu'*approximé* par des valeurs de paramètres dans les modèles comme **cmyk** ou **rgb** ; les couleurs originales se mélangent physiquement et dépendent aussi du type de papier retenu. Enfin, il existe certaines couleurs comme l'*or* ou l'*argent* qui sont difficilement reproductibles par des modèles paramétriques avec des encres standard ou des imprimantes laser.

1.4 Cercles chromatiques et accords de couleurs

Il s'est développé une longue histoire du placement de couleurs (✖ **teintes** ✖) sur des cercles pour discuter de problèmes théoriques ou pratiques sur les couleurs (par exemple Isaac Newton, Johann Wolfgang von Goethe). Une explication de ceci pourrait être que le cercle lui-même est un outil tout naturel pour illustrer des relations communes aussi bien que des propriétés opposées.
De nos jours, une certaine confusion portant sur les notions associées aux couleurs existe, dans la mesure où deux grands domaines qui y sont liés — l'art et le design graphique d'une part, la théorie scientifique de couleur de l'autre — tendent à utiliser les même mots pour décrire les propriétés de la couleur bien qu'en décrivant parfois des faits très différents ! Ainsi, l'apparence des cercles chromatiques diffère autant que la signification de concepts comme couleurs 'primaires' ou 'complémentaires'.

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

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

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

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

— les **couleurs complémentaires adjacentes** d'une couleur donnée sont les deux voisines immédiates de la couleur complémentaire, caractérisées par les positions $\frac{n\pm1}{2n} \cdot 360°$,
— les **couleurs analogues** d'une couleur donnée sont les deux ou quatre voisines, caractérisées par les positions $\pm\frac{1}{2n} \cdot 360°$ and $\pm\frac{2}{2n} \cdot 360°$.

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

2 L'interface utilisateur

2.1 Préparation

2.1.1 Installation de l'extension

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

2.1.2 Options de l'extension

En général, plusieurs types d'options existent :
- les options qui déterminent le pilote graphique comme expliqué dans [5] et [6], soit actuellement : `dvips`, `xdvi`, `dvipdf`, `dvipdfm`, `dvipdfmx`, `pdftex`, `dvipsone`, `dviwindo`, `emtex`, `dviwin`, `oztex`, `textures`, `pctexps`, `pctexwin`, `pctexhp`, `pctex32`, `truetex`, `tcidvi`, `vtex`, `xetex` ;
- les options qui déterminent le modèle colorimétrique cible [1] (`natural`, `rgb`, `cmy`, `cmyk`, `hsb`, `gray`, `RGB`, `HTML`, `HSB`, `Gray`) ou une sortie avec des couleurs désactivées (`monochrome`) ;
- les options qui contrôlent si certains ensembles de couleurs prédéfinies sont chargés et comment : `dvipsnames`, `dvipsnames*`, `svgnames`, `svgnames*`, `x11names`, `x11names*` ;
- les options qui déterminent quelles autres extensions doivent être chargées ou supportées : `table`, `fixpdftex`, `hyperref` ;
- les options qui influencent le comportement d'autres commandes : `prologue`, `kernelfbox`, `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
`\GetGinDriver` obsolètes) sont listées en table 1 page 10. Afin de faciliter la coopération avec
`\GinDriver` l'extension hyperref, il existe une commande `\GetGinDriver` [2] qui récupère le nom du pilote effectivement utilisé et qui le place dans la commande `\GinDriver`.

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

1. La section 2.2.3 page 13 explique comment ce paramétrage peut être annulé n'importe où dans un document.
2. Cette commande est exécutée automatiquement si l'option d'extension `hyperref` est sélectionnée.

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

2.1.3 Éxecution de commandes additionnelles à l'initialisation

`\xcolorcmd` Voici un interface simple pour passer des commandes devant être exécutées à la fin de l'extension 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 LaTeX 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 occurences de *noir* remplacées par du *rouge*, sans besoin de modifier le fichier source lui-même. (La ligne de commande peut varier selon les systèmes d'exploitation et les distributions de TeX).

2.2 Modèles colorimétriques

2.2.1 Modèles colorimétriques supportés

La liste des modèles colorimétriques et les plages de valeur de leurs paramètres sont données en table 3 page 11. Notez bien que le support de ces couleurs est indépendant du pilote graphique choisi.
Ce support permet d'ailleurs de spécifier des couleurs directement avec leurs paramètres, par exemple avec `\textcolor[cmy]{0.7,0.5,0.3}{toto}` (toto) ou `\textcolor[HTML]{AFFE90}{toto}` (toto).

rgb, cmyk, hsb, gray Ce sont les modèles supportés directement par PostScript. C'est pourquoi ✖ **nous nous référons** ✖ à [1] pour une description de leurs propriétés et relations. Il existe une commande spécifique pour régler finement les mécanismes de ✖ *undercolor-removal* ✖ et ✖ *black-generation]* ✖ durant la conversion vers le modèle **cmyk**, voir section 6.3.2 page 53 pour plus de détails. `\adjustUCRBG`

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

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

TABLE 1 – Options de l'extension

Option	*Description*
`natural`	(valeur par défaut.) Garde toutes les couleurs dans leur modèle, à l'expcetion de **RGB** (converti en **rgb**), **HSB** (converti en **hsb**), et **Gray** (converti en **gray**).
`rgb`	Convertit toutes les couleurs en modèle **rgb**.
`cmy`	Convertit toutes les couleurs en modèle **cmy**.
`cmyk`	Convertit toutes les couleurs en modèle **cmyk**.
`hsb`	Convertit toutes les couleurs en modèle **hsb**.
`gray`	Convertit toutes les couleurs en modèle **gray**. Particulièrement utile pour simuler un rendu en noir et blanc d'une imprimante monochrome.
`RGB`	Convertit toutes les couleurs en modèle **RGB** (et ensuite en **rgb**).
`HTML`	Convertit toutes les couleurs en modèle **HTML** (et ensuite en **rgb**).
`HSB`	Convertit toutes les couleurs en modèle **HSB** (et ensuite en **hsb**).
`Gray`	Convertit toutes les couleurs en modèle **Gray** (et ensuite en **gray**).
`dvipsnames`, `dvipsnames*`	Charge un ensemble de couleurs prédéfinies.[1]
`svgnames`, `svgnames*`	Charge un ensemble de couleurs prédéfinies selon la norme SVG 1.1.[1]
`x11names`, `x11names*`	Charge un ensemble de couleurs prédéfinies selon la norme Unix/X11.[1]
`table`	Charge l'extension colortbl contenant les outils de colorisation des lignes, colonnes et cellules dans des tables.
`fixpdftex`	Charge l'extension pdfcolmk permettant d'améliorer la gestion des couleurs dee `pdftex` (voir section 2.15.2 page 32).
`hyperref`	Permet de prendre en charge l'extension hyperref pour les ✖ **expressions de couleur** ✖ en définissant des ✖ **clés** ✖ additionnelles (voir section 2.10 page 29).
`prologue`	Écrit des informations en début de fichier `.xcp` pour chaque définition de couleur (comme décrit en section 2.5.1 page 20).
`kernelfbox`	Utilise la méthode du noyau LaTeX pour dessiner des boîtes `\f(rame)box`[2].
`xcdraw`	Use driver-specific commands to draw frames and color boxes.[2]
`noxcdraw`	(Valeur par défaut.) Utilise un code générique pour dessiner les encadrements et boîets de couleur.[2]
`fixinclude`	Empêche la réinitialisation de couleur de `dvips` avant l'inclusion de fichier `.eps` (voir section 2.15.3 page 33).
`showerrors`	(Valeur par défaut.) Affiche un message d'erreur si une couleur non définie est utilisée (comportement similaire à celui de l'extension color originale).
`hideerrors`	Affiche seulement une alerte en cas d'utilisation d'une couleur non définie et remplace cette couleur par du *noir*.

[1] Voir section 2.4.2 page 19. [2] Voir section 2.6.2 page 24.

Table 2 – Ordre de chargement des extensions

Chargement/Extension	colortbl	pdfcolmk	hyperref	pstricks	color	pstcol
Avant xcolor	non	non	permis	permis[1]	non	non
Avec l'option xcolor	`table`	`fixpdftex`	—	—	—	—
Après xcolor	non	permis	permis	permis	non	non

[1] Les versions récentes de pstricks chargent xcolor par défaut.

Table 3 – Modèles colorimétriques supportés

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

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

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

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

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

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

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

est placé à 60° dans le cercle **Hsb** (le *rouge* étant à 0°) ; cependant dans la plus plupart des cercles chromatiques servant aux artistes, le *jaune* est à 120°. Ainsi, une entrée « `120,60` » ferait sens si nous avions décidé de répliquer un cercle chromatique d'artiste par le biais de **tHsb**. Voir la section 6.3.6 page 57 pour la formule exacte de la transformation et les restrictions avancées, et la section 1.4 page 7 pour les cercles chromatiques et les accords de couleur. La figure 11 page 39 peut servir pour effectuer des comparaisons.
Exemple : '`\def\rangetHsb{60,30;120,60;180,120;210,180;240,240}`' correspond en fait au paramétrage par défaut de xcolor.

wave Avec ce modèle nous essayons de transformer les longueurs d'onde en un modèle de colorimétrique standard afin de réaliser une approximation de l'apparence visuelle des ondes lumineuses. Tandis que le spectre visible couvre un intervalle de valeur de 400 à 750 nm, l'implémentation dans xcolor permet de traiter toutes les longueurs d'onde qui ont une valeur absolue inférieur à 16383.99998 (le plus grand nombre que TEX 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 34 illustre seulement l'intervalle de valeur mention ci-dessus. Notez qu'il n'est pas possible de convertir fidèlement les autres modèles en **wave** puisque ce dernier ne couvre qu'un ensemble limité de couleurs.

RGB, HSB, Gray Ce sont des modèles dérivés, transformant la plage de valeurs continue $[0, 1]$ des paramètres de **rgb**, **hsb** et **gray** en un ensemble de valeurs finies ; ce qui nous 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⟩`}`, `\def\rangeHSB{`⟨M⟩`}`, and `\def\rangeGray{`⟨N⟩`}`. La mo-
`\rangeHSB` dification de ces constantes peut être fait *avant* ou *après* que l'extension xcolor ait
`\rangeGray` 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 modéles colorimétriques individuels

`\substitutecolormodel` `{`⟨*modèle source*⟩`}{`⟨*liste de modèles cibles*⟩`}`

Substitue le ⟨*modèle source*⟩ par le premier modèle disponible apparaissant dans la ⟨*liste de modèles cibles*⟩. Seuls les modèles de type ⟨*modèle numérique*⟩ sont possibles ; tous les changements sont locaux au groupe courant, mais un `\xglobal` préfixé est respecté.
Exemple : supposons que le pilote actuel a une implémentation incorrecte de **hsb** tandis que **rgb** paraît correct. Alors `\substitutecolormodel{hsb}{rgb}` pourrait être un bon choix puisqu'il convertit — à partir de ce point — toutes les définitions des couleurs **hsb** en spécifications du modèle **rgb** par le biais des algorithmes de xcolor, sans toucher aux autres modèles.

2.2.3 Changement du modèle colorimétrique cible dans un document

`\selectcolormodel` `{`⟨*num model*⟩`}`
Définit le modèle cible au ⟨*modèle numérique*⟩, où ce dernier est un des noms de modèles autorisés comme option de l'extension (autrement dit, `natural`, `rgb`, `cmy`, `cmyk`, `hsb`, `gray`, `RGB`, `HTML`, `HSB`, `Gray`), voir figure 4 page 36 pour un exemple. Il y a deux possibilités pour rendre possible la conversion au modèle cible :

`\ifconvertcolorsD`
- au moment de la *définition* des couleurs [3] (autrement dit dans `\definecolor` et ses assimilées) ; ceci est contrôlé par la bascule `\ifconvertcolorsD` ;

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

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

2.3 Arguments et terminologie

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

2.3.1 Remarques additionnelles et restrictions sur les arguments

⟨*vide*⟩ ⟨*moins*⟩ ⟨*plus*⟩ ⟨*ent*⟩ ⟨*num*⟩ ⟨*déc*⟩ ⟨*pct*⟩ ⟨*div*⟩

Chaînes basiques et nombres Ces arguments ne nécessitent pas beaucoup

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

Table 4 – Arguments et terminologie

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

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

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

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

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

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

suivantes : `\definecolor`, `\providecolor`, `\definecolorset`, `\providecolorset`. Tout d'abord, ⟨*modèle-spéc*⟩ 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 ⟨*liste-spéc*⟩, de telle façon à ce que le traitement aboutisse à une paire (⟨*modèle*⟩, ⟨*spéc*⟩) cohérente. Ceci explique pourquoi il n'y a plus d'ambiguité possible dans la définition de couleur réellement suivie. La forme étendue ⟨*modèle central*⟩`:`$\langle$*modèle*$\rangle_1$`/`$\langle$*modèle*$\rangle_2$`/.../`$\langle$*modèle*$\rangle_m$ provoque la conversion immédiate de la ⟨*spéc*⟩ adéquate au ⟨*modèle central*⟩ ; un modèle inconnu sera tout simplment ignoré ici, sans aucun commentaire.
Exemples : '`rgb/cmyk/named/gray`', '`0,0,0/0,0,0,1/Black/0`', '`rgb:cmy/hsb`'.

⟨*type*⟩ **L'argument de type** Ceci est utilisé uniquement dans le contexte de commandes de définition de couleur, voir la description de `\definecolor` et assimilées.

⟨*expr*⟩
⟨*préfixe*⟩
⟨*expr de mélange*⟩
⟨*suffixe*⟩

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 ⟨*préfixe*⟩ 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 complère* aveec $\langle$*nom*$\rangle_n$ = `white`, afin d'éviter quelques saisies dans le cas des teintes. La chaîne ⟨*suffixe*⟩ 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 ⟨*nom*⟩ indique le nom d'une ✖ ***color series*** ✖ ;
— l'⟨*expr de mélange*⟩ 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]`'.

⟨*expr étendue*⟩ **Expressions de couleur étendues** Ces expressions fournissent une autre méthode pour mélanger des couleurs, voir section 2.3.3 page 18 pour plus d'informations. La forme raccourcie

$$\langle \textit{modèle central}\rangle\texttt{:}\langle \textit{expr}\rangle_1\texttt{,}\langle \textit{déc}\rangle_1\texttt{;}\langle \textit{expr}\rangle_2\texttt{,}\ \langle \textit{déc}\rangle_2\texttt{;}\ldots\texttt{;}\langle \textit{expr}\rangle_k\texttt{!}\langle \textit{déc}\rangle_k$$

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

$$\langle \textit{modèle central}\rangle\texttt{,}\langle \textit{div}\rangle\texttt{:}\langle \textit{expr}\rangle_1\texttt{,}\langle \textit{déc}\rangle_1\texttt{;}\langle \textit{expr}\rangle_2\texttt{,}\langle \textit{déc}\rangle_2\texttt{;}\ldots\texttt{;}\langle \textit{expr}\rangle_k\texttt{!}\langle \textit{déc}\rangle_k$$

avec la définition suivante (impliquant une somme non nulle de tous les coefficients $\langle$*déc*$\rangle_\kappa$) :

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

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

⟨*expr fonctionnelle*⟩ ⟨*fonction*⟩ **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`'.

⟨*couleur*⟩ ⟨*expr de couleur*⟩ **Couleurs** Au final, ⟨*couleur*⟩ est un argument générique recouvrant les différents concepts de spécification des couleurs. Ceci signifie qu'à chaque fois qu'un argument ⟨*couleur*⟩ est utilisable, la totalité des noms et expressions vues ci-dessus peuvent être utilisées.

2.3.2 Signification des expressions de couleur standards

✖We explain now how an expression

$$\langle \textit{préfixe}\rangle\langle name\rangle\texttt{!}\langle pct\rangle_1\texttt{!}\langle name\rangle_1\texttt{!}\langle pct\rangle_2\texttt{!}\ldots\texttt{!}\langle pct\rangle_n\texttt{!}\langle name\rangle_n\langle postfix\rangle$$

is being interpreted and processed :

1. First of all, the model and color parameters of ⟨*name*⟩ are extracted to define a temporary color ⟨*temp*⟩. If ⟨*postfix*⟩ has the form '`!![`⟨*num*⟩`]`', then ⟨*temp*⟩ will be the corresponding (direct-accessed) color ⟨*num*⟩ from the series ⟨*name*⟩.
2. Then a color mix, consisting of $\langle pct\rangle_1$% of color ⟨*temp*⟩ and $(100-\langle pct\rangle_1)$% of color $\langle name\rangle_1$ is computed ; this is the new temporary color ⟨*temp*⟩.
3. The previous step is being repeated for all remaining parameter pairs $(\langle pct\rangle_2,\langle name\rangle_2)$, ..., $(\langle pct\rangle_n,\langle name\rangle_n)$.
4. If ⟨*préfixe*⟩ consists of an odd number of minus signs '-', then ⟨*temp*⟩ will be changed into its complementary color.
5. If ⟨*postfix*⟩ has the form '`!!+`', '`!!++`', '`!!+++`', etc., a number of step commands (= number of '+' signs) are performed on the underlying color series ⟨*name*⟩. This has no consequences for the color ⟨*temp*⟩.
6. Now the color ⟨*temp*⟩ is being displayed or serves as an input for other operations, depending on the invoking command.

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

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

Figures 5 à 6 page 36 show some first applications of colors and expressions. More examples are given in figure 3 page 35. Over and above that, a large set of color examples can be found in [9].

2.3.3 Meaning of extended color expressions

An *extended color expression*

$$\langle core\ model\rangle\texttt{:}\langle expr\rangle_1\texttt{,}\langle dec\rangle_1\texttt{;}\langle expr\rangle_2\texttt{,}\langle dec\rangle_2\texttt{;}\ldots\texttt{;}\langle expr\rangle_k\texttt{,}\langle dec\rangle_k$$

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

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

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

$$\langle core\ model\rangle\texttt{,}\langle div\rangle\texttt{:}\langle expr\rangle_1\texttt{,}\langle dec\rangle_1\texttt{;}\langle expr\rangle_2\texttt{,}\langle dec\rangle_2\texttt{;}\ldots\texttt{;}\langle expr\rangle_k\texttt{,}\langle dec\rangle_k$$

does the same operation with the only difference that the divisor $\langle div\rangle$ is being specified instead of calculated. In the above example, we get a shaded version via `\color{rgb,9:red,4;green,2;yellow,1}`. Note that it is not forbidden to specify a $\langle div\rangle$ argument which is smaller than the sum of all $\langle dec\rangle_\kappa$, such that one or more of the final color specification parameters could be outside the interval $[0,1]$. However, the mapping of equation (7) takes care of such cases.

2.3.4 Color functions

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

wheel
twheel

Color wheel calculations Arguments : $\langle angle\rangle$ or $\langle angle\rangle$`,`$\langle full\ circle\rangle$, the former being an abbreviataion of $\langle angle\rangle$`,\rangeHsb`. These functions allow to calculate related colors by harmonic relations based on color wheels (cf. section 1.4 page 7). The second argument $\langle full\ circle\rangle$ declares how many units a full circle consists of, the first argument states by how many units the given color has to be rotated. To this end, the given color is first converted to **Hsb** (in case of *wheel*), yielding *hue*°, *saturation*, and *brightness*, respectively. Then

$$hue^\circ := hue^\circ + \frac{\langle angle\rangle}{\langle full\ circle\rangle}\cdot H, \qquad hue := u\Big(\frac{hue^\circ}{H}\Big) \tag{4}$$

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

2.4 Predefined colors

2.4.1 Colors that are always available

Within `xcolor.sty`, the following color names are defined : *red*, *green*, *blue*, *cyan*, *magenta*, *yellow*, *black*, *gray*, *white*, *darkgray*, *lightgray*, *brown*, *lime*, *olive*, *orange*, *pink*, *purple*, *teal*, *violet*.
This base set of colors can be used without restrictions in all kinds of color expressions, as explained in section 2.3 page 13.

2.4.2 Additional sets of colors

There are also sets of color names that may be loaded by xcolor via package options, available in two variants : a 'normal' version (e.g., `dvipsnames`) and a 'starred' version (e.g., `dvipsnames*`). The first variant simply defines all the colors *immediately*, whereas the second applies the mechanism of *deferred* definition. In the latter case, individual color names have to be activated by `\definecolors` or `\providecolors` commands, as described in section 2.5.4 page 22, before they can be applied in a document.

— `dvipsnames`/`dvipsnames*` loads a set of 68 **cmyk** colors as defined in the `dvips` driver. However, these colors may be used in all supported drivers.
— `svgnames`/`svgnames*` loads a set of 151 **rgb** color names [5] according to the SVG 1.1 specification [17] [6], enhanced by 4 names taken from the file `rgb.txt` which is part of Unix/X11 distributions. Note that HTML4 accepts a subset of 16 color keywords (using identical specifications), see [16] and section 4 page 41.
— `x11names`/`x11names*` loads a set of 317 **rgb** color names [7] that are basically variations of a subset of the SVG set mentioned before, according to the file `rgb.txt` which is part of Unix/X11 distributions [8]. We describe now how to access all 752 color names of `rgb.txt` without much effort :
  — Load `x11namees` as well as `svgnames`.
  — Capitalise initials and skip blanks : *DarkSlateGray* instead of *dark slate gray*.
  — X11 names without numbers are identical to the corresponding SVG colors, except in 5 cases : use *Gray0*, *Grey0*, *Green0*, *Maroon0*, *Purple0*

5. In fact, these names represent 141 different colors.
6. Actually, the cited specification lists only lowercase names, and the original definitions are given in **RGB** parameters, converted to **rgb** by the author.
7. These names represent 315 different colors.
8. Again, the original definitions are given in **RGB** parameters, converted to **rgb** by the author.

instead of *Gray*, *Grey*, *Green*, *Maroon*, *Purple* to obtain the original X11 colors.
— For $N = 0, 1, \ldots, 100$ use '`[gray]{`$N/100$`}`' or '`black!`$100 - N$' instead of *grayN* or *greyN*.

The color names and corresponding displays are listed in section 4 page 41. Section 2.15.1 page 32 describes how to deal with name clashs while using both `svgnames` and `dvipsnames` in the same document. See also [9] for a systematic set of color and mix examples.

2.5 Color definition

2.5.1 Ordinary and named colors

In the extension color there is a distinction between 'colors' (defined by the command `\definecolor`) and 'named colors' (defined by `\DefineNamedColor`, which is allowed only in the preamble). Whenever an ordinary color is being used in a document, it will be translated into a `\special` command that contains a — driver-specific — numerical description of the color which is written to the `.dvi` file. On the other hand, named colors offer the opportunity to store numerical values at a central place whereas during usage, colors may be identified by their names, thus enabling post-processing if required by the output device.
All drivers delivered with the standard graphics package support the *formalism* of defining and invoking 'named colors'. However, real support for the *concept* behind that, i.e. employing names instead of parameters, ranges from 'none' to 'complete'. We demonstrate the current situation for three different drivers :

— `dvips` has very good support for the 'named' concept ; the PostScript equivalents to the color names defined by `dvipsnames` are being loaded – unless switched off – by *dvips* automatically. However, additional names have to be made known to the PostScript interpreter by some kind of header file. Since version 2.01, xcolor offers an integrated solution for this task : by invoking the package option `prologue`, a PostScript header file `xcolor.pro` is loaded by *dvips*. Additionally, under this option every color definition command [9] (`\definecolor`, `\colorlet`, etc.) will generate some PostScript code that is written to an auxiliary file with the extension `.xcp` (shortcut for **xc**olor **p**rologue). This file is as well loaded by *dvips* as a prologue, thus making all color names available to the PostScript interpreter. Of course, the `.xcp` file may be edited before *dvips* is applied, making it easy to change device-specific color parameters at a central place. Note that the PostScript code is designed similar to `color.pro` : only *new* names are defined. This allows to preload other prologue files with color definitions that are not being destroyed by xcolor. On the other hand, it requires the user to take care about redefining color names.
Example : `\colorlet{foo}{red}\colorlet{foo}{blue}\color{foo}` will switch to *blue* in the usual xcolor logic, however the `.ps` file would display *red* (unless *foo* had been defined differently before).

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

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

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

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

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

2.5.2 Color definition in xcolor

`\definecolor` [⟨*type*⟩]{⟨*name*⟩}{⟨*model-list*⟩}{⟨*spec-list*⟩}[10]

This is one of the commands that may be used to assign a ⟨*name*⟩ to a specific color. Afterwards, this color is known to the system (in the current group) and may be used in *color expressions*, as explained in section 2.3 page 13. It replaces both color's `\DefineNamedColor` and `\definecolor`. Note that an already existing color ⟨*name*⟩ will be overwritten. The variable `\tracingcolors` controls whether such an overwriting will be logged or not (see section 2.13 page 31 for details). The arguments are described in section 2.3 page 13. Hence, valid expressions for color definitions are

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

where the last command is equivalent to `\colorlet{myblack}{Black}` (see below); the second command defines *red* in the **rgb** or **cmyk** model, depending on the current setting of the *target model*, whereas the third will additionally transform the color to **hsb** prior to saving. Note that there is a special pstricks-related version as described in section 2.11 page 30.

`\providecolor` [⟨*type*⟩]{⟨*name*⟩}{⟨*model-list*⟩}{⟨*spec-list*⟩}

Similar to `\definecolor`, but the color ⟨*name*⟩ is only defined if it does not exist already.

`\colorlet` [⟨*type*⟩]{⟨*name*⟩}[⟨*num model*⟩]{⟨*color*⟩}

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

Copies the actual color which results from ⟨*color*⟩ to ⟨*name*⟩. If ⟨*num model*⟩ is non-empty, ⟨*color*⟩ is first transformed to the specified model, before ⟨*name*⟩ is being defined. The pseudo model 'named' is *not* allowed here, it may, however, be specified in the ⟨*type*⟩ argument. Note that an already existing color ⟨*name*⟩ will be overwritten.
Example : we said `\colorlet{tableheadcolor}{gray!25}` in the preamble of this document. In most of the tables we then formatted the first row by using the command `\rowcolor{tableheadcolor}`.

2.5.3 Defining sets of colors

`\definecolorset` `[`⟨*type*⟩`]{`⟨*model-list*⟩`}{`⟨*head*⟩`}{`⟨*tail*⟩`}{`⟨*set spec*⟩`}`
This command facilitates the construction of *color sets*, i.e. (possibly large) sets of individual colors with common underlying ⟨*model-list*⟩ and ⟨*type*⟩. Here, ⟨*set spec*⟩ = ⟨*name*⟩$_1$,⟨*spec-list*⟩$_1$;...;⟨*name*⟩$_l$,⟨*spec-list*⟩$_l$ ($l \geq 1$ name/specification-list pairs). Individual colors are being constructed by single

`\definecolor[`⟨*type*⟩`]{`⟨*head*⟩⟨*name*⟩$_\lambda$⟨*tail*⟩`}{`⟨*model-list*⟩`}{`⟨*spec-list*⟩$_\lambda$`}`

commands, $\lambda = 1, \ldots, l$. For example,
- `\definecolorset{rgb}{}{}{red,1,0,0;green,0,1,0;blue,0,0,1}` could be used to define the basic colors *red*, *green*, and *blue* ;[11]
- `\definecolorset{rgb}{x}{10}{red,1,0,0;green,0,1,0;blue,0,0,1}` would define the colors *xred10*, *xgreen10*, and *xblue10*.

`\providecolorset` `[`⟨*type*⟩`]{`⟨*model-list*⟩`}{`⟨*head*⟩`}{`⟨*tail*⟩`}{`⟨*set spec*⟩`}`
Similar to `\definecolorset`, but based on `\providecolor`, thus the individual colors are defined only if they do not exist already.

2.5.4 Immediate and deferred definitions

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

11. Actually, xcolor uses a more complicated variant to provide the basic colors for different underlying models (see the source code for the full command) :
`\definecolorset{rgb/hsb/cmyk/gray}{}{}{red,1,0,0/0,1,1/0,1,1,0/.3;green,...}`.

Note that the following commands are only to be used in the document preamble, since the definition stack of colors for deferred definitions is deleted at the begin of the document body — in order to save memory.

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

`\preparecolorset` `[`⟨*type*⟩`]{`⟨*model-list*⟩`}{`⟨*head*⟩`}{`⟨*tail*⟩`}{`⟨*set spec*⟩`}`
`\ifdefinecolors` Similar to `\definecolorset`, but depending on the `\ifdefinecolors` switch : if set to 'true', to each element of the set the command `\definecolor` (i.e. immediate definition) is applied ; if set to 'false', `\preparecolor` (i.e. deferred definition) is applied. For example, the package option `svgnames` performs something like `\definecolorstrue\preparecolorset`, whereas `svgnames*` acts like `\definecolorsfalse\preparecolorset`. Both options set `\definecolorstrue` at the end, in order to have a proper starting point for other color sets.

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

`\definecolors` `{`⟨*id-list*⟩`}`
Recall that ⟨*id-list*⟩ has the form ⟨*ext id*⟩$_1$`,...,`⟨*ext id*⟩$_l$ where each ⟨*ext id*⟩$_\lambda$ is either an identifier ⟨*id*⟩$_\lambda$ or an assignment ⟨*id*⟩$_{\lambda'}$`=`⟨*id*⟩$_\lambda$. We consider the first case to be an abbreviation for ⟨*id*⟩$_\lambda$`=`⟨*id*⟩$_\lambda$ and describe the general case : the definition stack is searched for the name ⟨*id*⟩$_\lambda$ and its corresponding color parameters ; if there is no match, nothing happens ; if the name ⟨*id*⟩$_\lambda$ is on the stack and its color parameters are ⟨*type*⟩$_\lambda$, ⟨*model*⟩$_\lambda$, and ⟨*spec*⟩$_\lambda$, then the command `\definecolor[`⟨*type*⟩$_\lambda$`]{`⟨*id*⟩$_{\lambda'}$`}{`⟨*model*⟩$_\lambda$`}{`⟨*spec*⟩$_\lambda$`}` is executed. Thus, the user may control by which names the *prepared* colors are to be used in the document. Note that the entry ⟨*id*⟩$_\lambda$ is not removed from the stack, such that it can be used several times (even within the same `\definecolors` command).

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

2.5.5 Global color definitions

`\ifglobalcolors` By default, definitions via `\definecolor`, `\providecolor`, ... are available only within the current group. By setting `\globalcolorstrue`, all such definitions are
`\xglobal` being made globally available — until the current group ends.[12] Another method

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

to specify that an individual color definition is to be made global is to prefix it by `\xglobal`, e.g., `\xglobal\definecolor{foo}`....

2.6 Color application

2.6.1 Standard color commands

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

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

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

`\pagecolor` `{`⟨*color*⟩`}`
`[`⟨*model-list*⟩`]{`⟨*spec-list*⟩`}`
Specifies the background color for the current, and all following, pages. It is a global declaration which does not respect TEX groups.
Remark : all of these commands except `\color` require that the ⟨*color*⟩ resp. ⟨*spec*⟩ arguments are put into curly braces `{}`, even if they are buried in macros.
For example, after `\def\foo{red}`, one may say `\color\foo`, but one should always write `\textcolor{\foo}{bar}` instead of `\textcolor\foo{bar}` in order to avoid strange results.
Note that color-specific commands from other packages may give unexpected results if directly confronted with color expressions (e.g., soul's `\sethlcolor` and friends). However, one can turn the expression into a name via `\colorlet` and try to use that name instead.

2.6.2 Colored boxes

`\colorbox` `{`⟨*color*⟩`}{`⟨*text*⟩`}`
`[`⟨*model-list*⟩`]{`⟨*spec-list*⟩`}{`⟨*text*⟩`}`
Takes the same argument forms as `\textcolor`, but the color specifies the *background* color of the box.

`\fcolorbox` `{`⟨*frame color*⟩`}{`⟨*background color*⟩`}{`⟨*text*⟩`}`
`[`⟨*model-list*⟩`]{`⟨*frame spec-list*⟩`}{`⟨*background spec-list*⟩`}{`⟨*text*⟩`}`
`[`⟨*fr. model-list*⟩`]{`⟨*fr. spec-list*⟩`}[`⟨*backgr. model-list*⟩`]{`⟨*backgr. spec-list*⟩`}{`⟨*text*⟩`}`
`{`⟨*frame color*⟩`}[`⟨*background model-list*⟩`]{`⟨*background spec-list*⟩`}{`⟨*text*⟩`}`
Puts a frame of the first color around a box with a background specified by the second color. If only the first optional argument is given, it specifies the color model for both colors. Besides the possibility to specify color *expressions* as arguments, `\fcolorbox` now offers more flexibility for its arguments than the color version :

— 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}`.

Additionally, `\fcolorbox` uses a new approach to frame drawing, which is an extension of Donald Arseneau's suggestion in bug report latex/3655 [2]. The main difference to LaTeX's implementation is that box construction and frame drawing are split into separate operations, such that the frame is drawn *after* the box contents has been constructed. This ensures that the frame is always on top of the box. Donald Arseneau improved speed as well as memory requirements of this approach. Furthermore, a new macro is introduced :

`\boxframe` `{`⟨*width*⟩`}{`⟨*height*⟩`}{`⟨*depth*⟩`}`

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

The new frame approach is used for `\fcolorbox` as well as LaTeX's `\fbox` and `\framebox` commands, unless the `kernelfbox` option is specified, which returns to LaTeX's original definitions of `\f(rame)box`.

Option `xcdraw` uses PostScript commands to draw frames and color boxes in case of the `dvips` driver and PDF code to draw frames in case of the `pdftex` and `dvipdfm` drivers. This is still experimental code that may confuse `.dvi` viewers. The opposite option `noxcdraw` forces usage of the generic (driver-independent) code.

2.6.3 Using the current color

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

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

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

2.6.4 Color testing

`testcolors` `[`⟨*num models*⟩`]`

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

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

Each `\testcolor` command generates a table row, containing a display sample plus the respective parameters for each of the models. If the column-model matches the model of the color in question, its parameters are underlined. Note that this command is only available within the `testcolors` environment.
For applications see figure 2 page 34 and figures 11, 12.

2.7 Color blending

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

`\blendcolors` `{`⟨*mix expr*⟩`}`
`\blendcolors*` `{`⟨*mix expr*⟩`}`

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

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

`\xglobal` In order to achieve global scope, `\blendcolors` may be prefixed by `\xglobal`.

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

2.8 Color masks and separation

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

$$\mathit{foo} \mapsto (c, m, y, k) \mapsto (\mu_c \cdot c, \mu_m \cdot m, \mu_y \cdot y, \mu_k \cdot k) \tag{5}$$

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 **cmy** model, see the last two rows in figure 9 page 38.

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

`\ifmaskcolors`

`\xglobal`

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

`\colormask`

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

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

2.9 Color series

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

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

2.9.1 Definition of a color series

`\definecolorseries` `{`⟨*name*⟩`}{`⟨*core model*⟩`}{`⟨*method*⟩`}[`⟨*b-model*⟩`]{`⟨*b-spec*⟩`}[`⟨*s-model*⟩`]{`⟨*s-spec*⟩`}`
Defines a color series called ⟨*name*⟩, whose calculations are performed within the color model ⟨*core model*⟩, where ⟨*method*⟩ selects the algorithm (one of `step`, `grad`, `last`, see below). The method details are determined by the remaining arguments :

— `[`⟨*b-model*⟩`]{`⟨*b-spec*⟩`}` specifies the *base* (= first) color in the algorithm, either directly, e.g., `[rgb]{1,0.5,0.5}`, or as a ⟨*color*⟩, e.g., `{-yellow!50}`, if the optional argument is missing.

— `[`⟨*s-model*⟩`]{`⟨*s-spec*⟩`}` specifies how the *step* vector is calculated in the algorithm, according to the chosen ⟨*method*⟩ :

  — `step`, `grad` : the optional argument is meaningless, and ⟨*s-spec*⟩ is a parameter vector whose dimension is determined by ⟨*core model*⟩, e.g., `{0.1,-0.2,0.3}` in case of `rgb`, `cmy`, or `hsb`.

  — `last` : the last color is specified either directly, e.g., `[rgb]{1,0.5,0.5}`, or as a ⟨*color*⟩, e.g., `{-yellow!50}`, if the optional argument is missing.

This is the general scheme :

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

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

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

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, `{`⟨*s-spec*⟩`}`, defines the directional vector *grad*.

— `last` : `{`⟨*s-spec*⟩`}` resp. `[`⟨*s-model*⟩`]{`⟨*s-spec*⟩`}` defines the color parameter vector *last*.

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

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

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 TEX's `\let` primitive, the *current* definition of the current color at the time of execution is used, there is no relation to current colors in any later stage of the document.

2.9.2 Initialisation of a color series

`\resetcolorseries` `[`⟨*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

`\colorseriescycle` 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 Application of a color series

There are two ways to display the current color of a color series : any of the *color expressions* in section 2.3 page 13 used within a `\color`, `\textcolor`, … command will display this color according to the usual syntax of such expressions. However, in the cases when ⟨*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*⟩ $= 0, 1, 2, \ldots$, starting with 0 for the base color. See figure 8 page 37 for a demonstration of different methods.

2.9.4 Differences between colors and color series

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

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

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

2.10 Border colors for hyperlinks

The hyperref package offers all kinds of support for hyperlinks, pdfmarks etc. There are two standard ways to make hyperlinks visible (see the package documentation [14] for additional information on how to set up these features) :

— print hyperlinks in a different color than normal text, using the keys *citecolor*, *filecolor*, *linkcolor*, *menucolor*, *pagecolor*, *runcolor*, *urlcolor* with color expressions, e.g., `\hypersetup{urlcolor=-green!50}` ;

— display a colored border around hyperlinks, using the keys *citebordercolor*, *filebordercolor*, *linkbordercolor*, *menubordercolor*, *pagebordercolor*, *runbordercolor*, *urlbordercolor* with explicit numerical **rgb** parameter specification, e.g., `\hypersetup{urlbordercolor={1 0.5 0.25}}`.

Obviously, the second method is somewhat inconvenient since it does not allow for color names or even color expressions. Therefore, xcolor provides — via the package option `hyperref` — a set of extended keys *xcitebordercolor*, *xfilebordercolor*, *xlinkbordercolor*, *xmenubordercolor*, *xpagebordercolor*, *xrunbordercolor*, *xurlbordercolor* which are being used in conjunction with color expressions, e.g., `\hypersetup{xurlbordercolor=-green!50}`.

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

2.11 Additional color specification in the pstricks world

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

— `\psset{linecolor=green!50}`

— `\psset{linecolor=[rgb]{0.5,1,0.5}}`

— `\psframebox[linecolor={[rgb]{0.5,1,0.5}}]{foo}`

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

`\definecolor` `[ps]{`⟨*name*⟩`}{`⟨*core model-list*⟩`}{`⟨*code*⟩`}`

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

2.12 Color in tables

`\rowcolors` `[`⟨*commands*⟩`]{`⟨*row*⟩`}{`⟨*odd-row color*⟩`}{`⟨*even-row color*⟩`}`
`\rowcolors*` `[`⟨*commands*⟩`]{`⟨*row*⟩`}{`⟨*odd-row color*⟩`}{`⟨*even-row color*⟩`}`

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

`\showrowcolors`
`\hiderowcolors`
`\rownum`

The *rowcolors status* is activated (i.e., use coloring scheme) by default and/or `\showrowcolors`, it is inactivated (i.e., ignore coloring scheme) by the command `\hiderowcolors`. The counter `\rownum` may be used within such a table to access the current row number. An example is given in figure 10 page 38. These commands require the `table` option (which loads the colortbl package).

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

2.13 Color information

`\extractcolorspec` `{`⟨*color*⟩`}{`⟨*cmd*⟩`}`

Extracts the color specification of ⟨*color*⟩ and puts it into ⟨*cmd*⟩; equivalent to `\def\cmd{{`⟨*model*⟩`}{`⟨*spec*⟩`}}`.

`\extractcolorspecs` `{`⟨*color*⟩`}{`⟨*model-cmd*⟩`}{`⟨*color-cmd*⟩`}`

Extracts the color specification of ⟨*color*⟩ and puts it into ⟨*model-cmd*⟩ and ⟨*color-cmd*⟩, respectively.

`\tracingcolors` `=`⟨*int*⟩

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

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

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

Note that whenever a color is used that has been defined via color's `\definecolor` command rather than xcolor's new `\definecolor` and friends, a warning message 'Incompatible color definition' will be issued. [13]

2.14 Color conversion

`\convertcolorspec` `{`⟨*model*⟩`}{`⟨*spec*⟩`}{`⟨*target model*⟩`}{`⟨*cmd*⟩`}`

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

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

2.15 Problems and solutions

2.15.1 Name clashs between `dvipsnames` and `svgnames`

Due to the fixed option processing order (which does not depend on the order how the options were specified in the `\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 [13]. However, its author also lists some limitations :

- Mark limitations : page breaks in math.
- LaTeX's output routine is redefinded.
  - 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
```

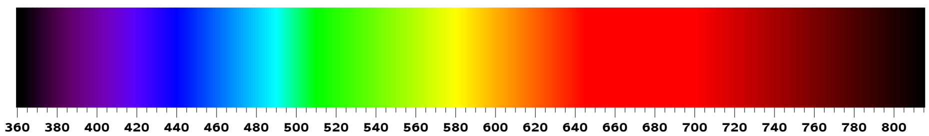
Now run the following code through LaTeX and *dvips* :

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

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

3 Examples

FIGURE 1 – Color spectrum



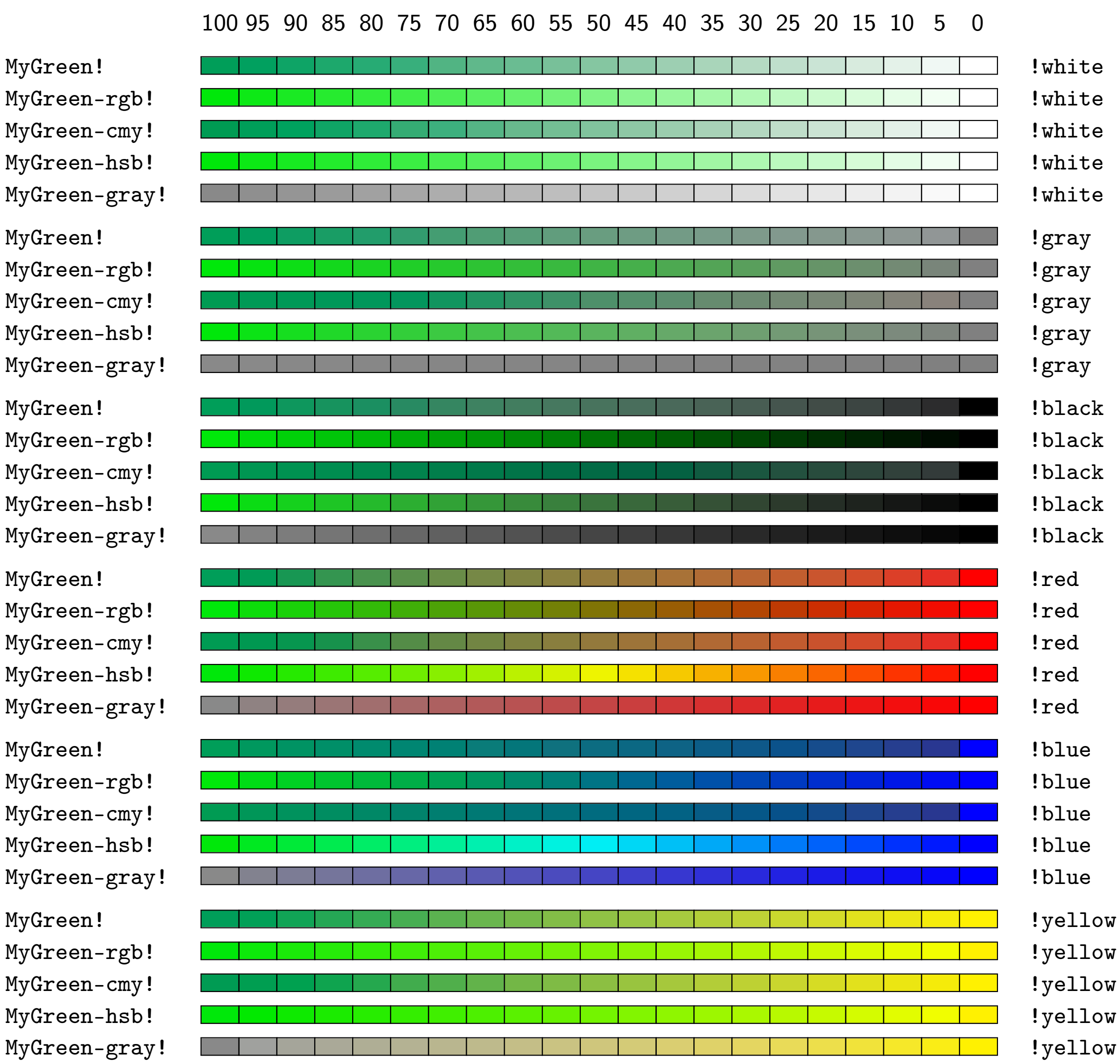

```
\newcount\WL \unitlength.75pt
\begin{picture}(460,60)(355,-10)
\sffamily \tiny \linethickness{1.25\unitlength} \WL=360
\multiput(360,0)(1,0){456}%
 {{\color[wave]{\the\WL}\line(0,1){50}}\global\advance\WL1}
\linethickness{0.25\unitlength}\WL=360
\multiput(360,0)(20,0){23}%
 {\picture(0,0)
  \line(0,-1){5}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

FIGURE 2 – Color testing

color	**rgb**	**cmyk**	**hsb**	**HTML**	**gray**
olive	0.5 0.5 0	0 0 1 0.5	0.16667 1 0.5	808000	0.39
red!50!green	0.5 0.5 0	0 0 0.5 0.5	0.16667 1 0.5	808000	0.445
-cyan!50!magenta	0.5 0.5 0	0 0 0.5 0.5	0.16667 1 0.5	808000	0.445
[cmyk]0,0,1,0.5	0.5 0.5 0	0 0 1 0.5	0.16667 1 0.5	808000	0.39
[cmyk]0,0,.5,.5	0.5 0.5 0	0 0 0.5 0.5	0.16667 1 0.5	808000	0.445
[rgb:cmyk]0,0,.5,.5	0.5 0.5 0	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

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

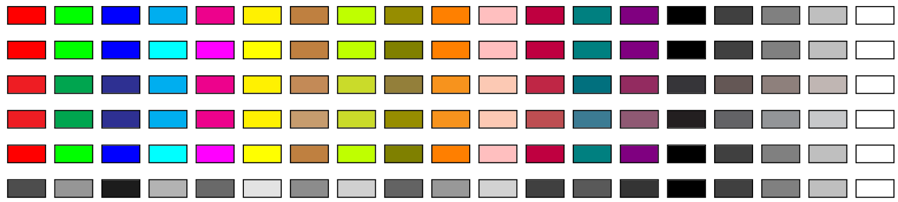


Figure 5 – Standard color expressions

`red`	`-red`
`red!75`	`-red!75`
`red!75!green`	`-red!75!green`
`red!75!green!50`	`-red!75!green!50`
`red!75!green!50!blue`	`-red!75!green!50!blue`
`red!75!green!50!blue!25`	`-red!75!green!50!blue!25`
`red!75!green!50!blue!25!gray`	`-red!75!green!50!blue!25!gray`

Figure 6 – Standard color expressions

```
\fboxrule6pt
\fcolorbox
 {red!70!green}% outer frame
 {yellow!30!blue}% outer background
 {\fcolorbox
   {-yellow!30!blue}% inner frame
   {-red!70!green}% inner background
   {Test\textcolor{red!72.75}{Test}\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	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	`\definecolorseries{test}{rgb}{step}[rgb]{.95,.85,.55}{.17,.47,.37}`
S_2	`\definecolorseries{test}{hsb}{step}[hsb]{.575,1,1}{.11,-.05,0}`
G_1	`\definecolorseries{test}{rgb}{grad}[rgb]{.95,.85,.55}{3,11,17}`
G_2	`\definecolorseries{test}{hsb}{grad}[hsb]{.575,1,1}{.987,-.234,0}`
L_1	`\definecolorseries{test}{rgb}{last}[rgb]{.95,.85,.55}[rgb]{.05,.15,.55}`
L_2	`\definecolorseries{test}{hsb}{last}[hsb]{.575,1,1}[hsb]{-.425,.15,1}`
L_3	`\definecolorseries{test}{rgb}{last}{yellow!50}{blue}`
L_4	`\definecolorseries{test}{hsb}{last}{yellow!50}{blue}`
L_5	`\definecolorseries{test}{cmy}{last}{yellow!50}{blue}`

Common definitions

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

FIGURE 9 – Color masking

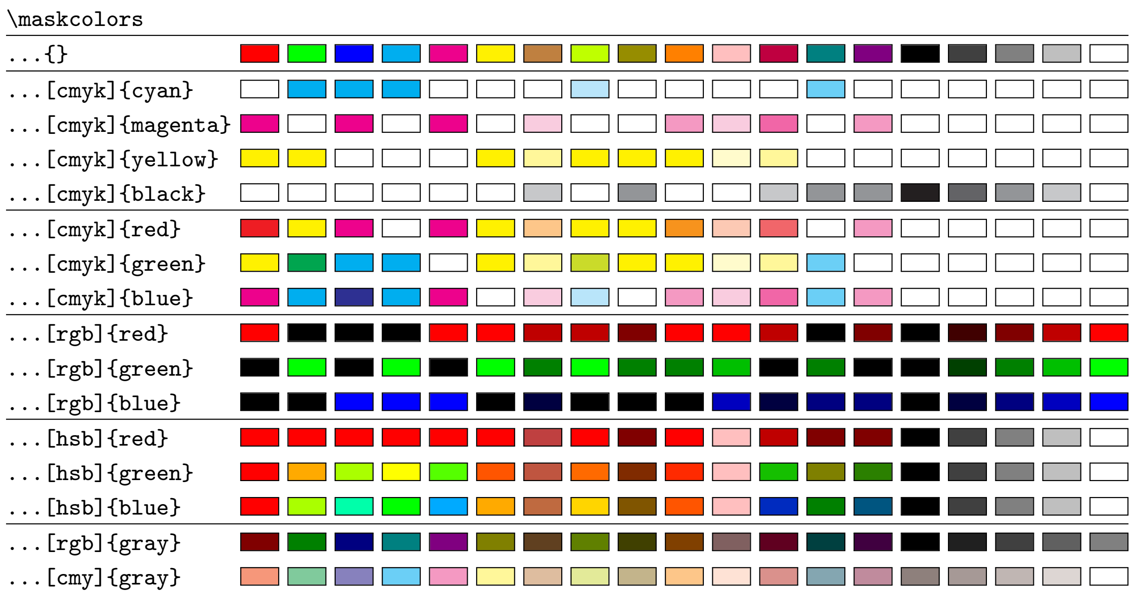


FIGURE 10 – Alternating row colors in tables : `\rowcolors` vs. `\rowcolors*`

```
\rowcolors[\hline]{3}{green!25}{yellow!50} \arrayrulecolor{red!75!gray}
\begin{tabular}{ll}
test & row \number\rownum\\
test & row \number\rownum\\
test & row \number\rownum\\
test & row \number\rownum\\
\arrayrulecolor{black}
test & row \number\rownum\\
test & row \number\rownum\\
\rowcolor{blue!25}
test & row \number\rownum\\
test & row \number\rownum\\
\hiderowcolors
test & row \number\rownum\\
test & row \number\rownum\\
\showrowcolors
test & row \number\rownum\\
test & row \number\rownum\\
\multicolumn{1}%
 {>{\columncolor{red!12}}l}{test} & row \number\rownum\\
\end{tabular}
```

test	row 1
test	row 2
test	row 3
test	row 4
test	row 5
test	row 6
test	row 7
test	row 8
test	row 9
test	row 10
test	row 11
test	row 12
test	row 13

test	row 1
test	row 2
test	row 3
test	row 4
test	row 5
test	row 6
test	row 7
test	row 8
test	row 9
test	row 10
test	row 11
test	row 12
test	row 13

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
complementary colors (two-color harmony) :				
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
color triad (three-color harmony) :				
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
color tetrad (four-color harmony) :				
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
split complementary colors :				
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
analogous (adjacent) colors :				
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
MediumAquamarine
MediumBlue
MediumOrchid
MediumPurple
MediumSeaGreen
MediumSlateBlue
MediumSpringGreen
MediumTurquoise
MediumVioletRed
MidnightBlue
MintCream
MistyRose
Moccasin
NavajoWhite
Navy
NavyBlue
OldLace
Olive
OliveDrab
Orange
OrangeRed
Orchid
PaleGoldenrod
PaleGreen
PaleTurquoise
PaleVioletRed
PapayaWhip
PeachPuff
Peru
Pink
Plum
PowderBlue
Purple
Red
RosyBrown
RoyalBlue
SaddleBrown
Salmon
SandyBrown
SeaGreen
Seashell
Sienna
Silver
SkyBlue
SlateBlue
SlateGray
SlateGrey
Snow
SpringGreen
SteelBlue
Tan
Teal
Thistle
Tomato
Turquoise
Violet
VioletRed
Wheat
White
WhiteSmoke
Yellow
YellowGreen

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

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

Duplicate colors : *Gray0* = *Grey0*, *Green0* = *Green1*.

5 Technical Supplement

5.1 Color models supported by drivers

Since some of the drivers only pretend to support the **hsb** model, we included some code to bypass this behaviour. The models actually added by xcolor are shown in the log file. Table 5 lists mainly the drivers that are part of current MiKTEX [11] distributions and their color model support. Probably, other distributions behave similarly.

TABLE 5 – Drivers and color models

Driver	*Version*	rgb	cmy	cmyk	hsb	gray	RGB	HTML	HSB	Gray
`dvipdf`	1999/02/16 v3.0i	d	n	d	n	d	i	n	n	n
`dvips`	1999/02/16 v3.0i	d	n	d	d	d	i	n	n	n
`dvipsone`	1999/02/16 v3.0i	d	n	d	d	d	i	n	n	n
`pctex32`	1999/02/16 v3.0i	d	n	d	d	d	i	n	n	n
`pctexps`	1999/02/16 v3.0i	d	n	d	d	d	i	n	n	n
`pdftex`	2006/03/02 v0.03p	d	n	d	n	d	i	n	n	n
`dvipdfm`	1998/11/24 vx.x [1]	d	n	d	a	d	i	n	n	n
`dvipdfm`	1999/9/6 vx.x [2]	d	n	d	a	d	i	n	n	n
`dvipdfmx`	?	d	n	d	f	d	i	n	n	n
`textures`	1997/5/28 v0.3	d	n	d	a	i	n	n	n	n
`vtex`	1999/01/14 v6.3	d	n	d	n	i	i	n	n	n
`xetex`	2004/05/09 v0.7	i	n	i	i	i	i	d	n	n
`tcidvi`	1999/02/16 v3.0i	i	n	i	n	i	d	n	n	n
`truetex`	1999/02/16 v3.0i	i	n	i	n	i	d	n	n	n
`dviwin`	1999/02/16 v3.0i	n	n	n	n	n	n	n	n	n
`emtex`	1999/02/16 v3.0i	n	n	n	n	n	n	n	n	n
`pctexhp`	1999/02/16 v3.0i	n	n	n	n	n	n	n	n	n
`pctexwin`	1999/02/16 v3.0i	n	n	n	n	n	n	n	n	n

`dviwindo` = `dvipsone` ; `oztex` = `dvips` ; `xdvi` = `dvips` + `monochrome`

[1] part of graphics package [2] additionally distributed with MiKTEX

Driver's color model support : d = direct, i = indirect, a = alleged, n = none, f = faulty

5.2 How xcolor handles driver-specific color models

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

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

5.3 Behind the scenes : internal color representation

Every definition of a color in order to access it by its name requires an internal representation of the color, i.e. a macro that contains some bits of information required by the driver to display the color properly.
color's `\definecolor{foo}{...}{...}` generates a command `\\color@foo` [14] which contains the color definition in a driver-dependent way ; therefore it is possible but non-trivial to access the color model and parameters afterwards (see the colorinfo package [12] for a solution).
color's `\DefineNamedColor{named}{foo}{...}{...}` generates `\col@foo` [15] which again contains some driver-dependent information. In this case, an additional `\\color@foo` will only be defined if the package option `usecolors` is active.
xcolor's `\definecolor{foo}{...}{...}` generates [16] 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 35 which displays the definitions with respect to the driver that was used to process this document.

5.4 A remark on accuracy

Since the macros presented here require some computation, special efforts were made to ensure a maximum of accuracy for conversion and mixing formulas — all within TeX's limited numerical capabilities. [17] We decided to develop and include a small set of commands to improve the quality of division and multiplication results, instead of loading one of the packages that provide multi-digit arithmetic and a lot more, like realcalc or fp. The marginal contribution of the latter packages seems not to justify their usage for our purposes. Thus, we stay within a sort of

---

14. The double backslash is intentional.
15. The single backslash is intentional.
16. This was introduced in version 1.10 ; prior to that, a command `\\xcolor@foo` with a different syntax was generated.
17. For example, applying the 'transformation' `\dimen0=0.`⟨*int*⟩`pt \the\dimen0` to all 5-digit numbers ⟨*int*⟩ of the range 00000. . .99999, exactly 34464 of these 100000 numbers don't survive unchanged. We are not talking about gobbled final zeros here . . .

Table 6 – Driver-dependent internal color representation

`dvips` driver		
`\\color@Plum=macro:` `->rgb .5 0 1.`	(`\definecolor{Plum}{rgb}{.5,0,1}`)	color
`\\color@Plum=macro:` `->\xcolor@ {}{rgb 0.5 0 1}{rgb}{0.5,0,1}.`	(`\definecolor{Plum}{rgb}{.5,0,1}`)	xcolor

`\col@Plum=macro:`
`->\@nil .`
`\\color@Plum=macro:`
`-> Plum.`
	(`\DefineNamedColor{Plum}{rgb}{.5,0,1}`)
	
	(with option `usenames`)
		color

`\\color@Plum=macro:` `->\xcolor@ {named}{ Plum}{rgb}{0.5,0,1}.`	(`\definecolor[named]{Plum}{rgb}{.5,0,1}`)	xcolor
`pdftex` driver		
`\\color@Plum=macro:` `->.5 0 1 rg .5 0 1 RG.`	(`\definecolor{Plum}{rgb}{.5,0,1}`)	color
`\\color@Plum=macro:` `->\xcolor@ {}{0.5 0 1 rg 0.5 0 1 RG}{rgb}{0.5,0,1}.`	(`\definecolor{Plum}{rgb}{.5,0,1}`)	xcolor

`\col@Plum=macro:`
`->.5 0 1 rg .5 0 1 RG.`
`\\color@Plum=macro:`
`->.5 0 1 rg .5 0 1 RG.`
	(`\DefineNamedColor{Plum}{rgb}{.5,0,1}`)
	
	(with option `usenames`)
		color

`\\color@Plum=macro:` `->\xcolor@ {}{0.5 0 1 rg 0.5 0 1 RG}{rgb}{0.5,0,1}.`	(`\definecolor[named]{Plum}{rgb}{.5,0,1}`)	xcolor

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

6 The Formulas

6.1 Color mixing

In general, we use linear interpolation for color mixing :

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

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

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

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

$$\text{ton}_{\textsf{hsb}}(C, p) = p \cdot C + (1-p) \cdot \big(\mathit{hue}, 0, \tfrac{1}{2}\big) \tag{12}$$

where $C = (\mathit{hue}, \mathit{saturation}, \mathit{brightness})$.
From equation (9) and the way how color expressions are being interpreted, as described in section 2.3 page 13, it is an easy proof by induction to verify that a color expression

$$C_0!P_1!C_1!P_2!\ldots!P_n!C_n \tag{13}$$

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

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

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

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

6.2 Conversion between integer and real models

We fix a positive integer n and define the sets $\mathcal{I}_n := \{0, 1, \ldots, n\}$ and $\mathcal{R} := [0, 1]$. The complement of $\nu \in \mathcal{I}_n$ is $n - \nu$, the complement of $x \in \mathcal{R}$ is $1 - x$.

TABLE 7 – Color constants

model/constant	`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

from/to	**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

$$\Gamma_n : \mathcal{R} \to \mathcal{I}_n,\ x \mapsto \mathrm{round}(n \cdot x, 0) = \left\lfloor \tfrac{1}{2} + n \cdot x \right\rfloor \tag{16}$$

where $\mathrm{round}(r, d)$ 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 $x \in \mathcal{R}$,*

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

Démonstration. Let $\nu := \Gamma_n(x)$, then from $-\frac{1}{2} \leq \eta := n \cdot x - \nu < \frac{1}{2}$ we conclude

$$\Gamma_n(1 - x) = \mathrm{round}(n(1 - x), 0) = \mathrm{round}(n - \nu - \eta, 0) = \begin{cases} n - \nu & \text{if } \eta \neq -\frac{1}{2} \\ n - \nu + 1 & \text{if } \eta = -\frac{1}{2} \end{cases}$$

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

Remark : the set $\mathcal{R}_n^\circ$ 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

$$\Delta_n^* : \mathcal{I}_n \to \mathcal{R},\ \nu \mapsto \frac{\nu}{n}. \tag{18}$$

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

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

which is equivalent to

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

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

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

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

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

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

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

which is obeyed by the function

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

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

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

$$\Delta'_n(\nu) + \Delta'_n(n-\nu) = 1 \iff \nu \notin \mathcal{I}^\circ_n := \big\{\tfrac{n-1}{2}, \tfrac{n+1}{2}\big\}\,. \tag{25}$$

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 :

$$\Delta''_n : \mathcal{I}_n \to \mathcal{R},\ \nu \mapsto \operatorname{round}\big(\tfrac{\nu}{n}, d_n(\tfrac{\nu}{n})\big) \tag{26}$$

where

$$d_n : [0,1] \to \mathbb{N},\ x \mapsto \min\big\{d \in \mathbb{N} \;\big|\; \Gamma_n\big(\operatorname{round}(\Delta^*_n(\Gamma_n(x)), d)\big) = \Gamma_n(x)\big\} \tag{27}$$

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 48 gives an overwiew of how each conversion pair is handled. In general, PostScript (as described in [1]) is used as a basis for most of the calculations, since it supports the color models **rgb**, **cmyk**, **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 :

$$E := (1,1,1) \tag{28}$$

6.3.1 The rgb model

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

$$(\mathit{cyan}, \mathit{magenta}, \mathit{yellow}) := E - (\mathit{red}, \mathit{green}, \mathit{blue}) \tag{29}$$

Conversion rgb to hsb (1) We set

$$x := \max\{\mathit{red}, \mathit{green}, \mathit{blue}\} \tag{30}$$
$$y := \operatorname{med}\{\mathit{red}, \mathit{green}, \mathit{blue}\} \tag{31}$$
$$z := \min\{\mathit{red}, \mathit{green}, \mathit{blue}\} \tag{32}$$
$$\tag{33}$$

where 'med' denotes the median of the values. Then,

$$\mathit{brightness} := x \tag{34}$$

Case $x = z$:

$$\mathit{saturation} := 0 \tag{35}$$
$$\mathit{hue} := 0 \tag{36}$$

Case $x \neq z$:

$$\mathit{saturation} := \frac{x - z}{x} \tag{37}$$
$$f := \frac{x - y}{x - z} \tag{38}$$

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

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

$$r := \frac{x - red}{x - z}, \qquad g := \frac{x - green}{x - z}, \qquad b := \frac{x - blue}{x - z} \tag{40}$$

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

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

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

and

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

where

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

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

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

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

$$gray := 0.3 \cdot red + 0.59 \cdot green + 0.11 \cdot blue \tag{45}$$

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

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

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

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

Complement of rgb color We simply take the complementary vector :

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

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 51.

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

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.

$$k := \min\{cyan, magenta, yellow\} \tag{50}$$
$$cyan := \min\{1, \max\{0, cyan - UCR_c(k)\}\} \tag{51}$$
$$magenta := \min\{1, \max\{0, magenta - UCR_m(k)\}\} \tag{52}$$
$$yellow := \min\{1, \max\{0, yellow - UCR_y(k)\}\} \tag{53}$$
$$black := BG(k) \tag{54}$$

Here, four additional functions are required :

$$UCR_c, UCR_m, UCR_y : [0,1] \to [-1,1] \qquad \textit{undercolor-removal}$$
$$BG : [0,1] \to [0,1] \qquad \textit{black-generation}$$

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

$$UCR_c(k) := \beta_c \cdot k \tag{55}$$
$$UCR_m(k) := \beta_m \cdot k \tag{56}$$
$$UCR_y(k) := \beta_y \cdot k \tag{57}$$
$$BG(k) := \beta_k \cdot k \tag{58}$$

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

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

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

Complement of cmy color We simply take the complementary vector :

$$(cyan^*, magenta^*, yellow^*) := E - (cyan, magenta, yellow) \tag{60}$$

6.3.3 The cmyk model

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

$$cyan := \min\{1, cyan + black\} \tag{61}$$
$$magenta := \min\{1, magenta + black\} \tag{62}$$
$$yellow := \min\{1, yellow + black\} \tag{63}$$

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

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

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

6.3.4 The hsb model

Conversion hsb to rgb

$$(\mathit{red}, \mathit{green}, \mathit{blue}) := \mathit{brightness} \cdot (E - \mathit{saturation} \cdot F) \tag{65}$$

with

$$i := \lfloor 6 \cdot \mathit{hue} \rfloor, \qquad f := 6 \cdot \mathit{hue} - i \tag{66}$$

and

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

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

$$m := 1 - \mathit{saturation} \tag{68}$$

$$n := 1 - f \cdot \mathit{saturation} \tag{69}$$

$$k := 1 - (1 - f) \cdot \mathit{saturation} \tag{70}$$

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

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

$$\lim_{f \to 1} (0, 1, f) = (0, 1, 1) = \lim_{f \to 0} (0, 1 - f, 1) \tag{72}$$

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.

$$(hue^{\circ}, saturation, brightness) := (H \cdot hue, saturation, brightness) \tag{73}$$

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

$$(Hue, Saturation, Brightness) := \big(\Gamma_M(hue), \Gamma_M(saturation), \Gamma_M(brightness)\big) \tag{74}$$

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 :*

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

$$brightness^* := 1 - brightness \cdot (1 - saturation) \tag{76}$$

$$saturation^* := \begin{cases} 0 & \text{if } brightness^* = 0 \\ \dfrac{brightness \cdot saturation}{brightness^*} & \text{if } brightness^* \neq 0 \end{cases} \tag{77}$$

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

$$C_{\mathsf{rgb}} + C^*_{\mathsf{rgb}} = b \cdot (E - s \cdot F) + b^* \cdot (E - s' \cdot F^*) \stackrel{!}{=} E, \tag{78}$$

which means that C_{rgb} is the complement of C^*_{rgb}. First we note that the parameters of C^* are in the legal range $[0, 1]$. This is obvious for h^*, b^*. From $b^* = 1 - b \cdot (1 - s) = 1 - b + b \cdot s$ we derive $b \cdot s = b^* - (1 - b) \leq b^*$, therefore $s^* \in [0, 1]$, and

$$b^* = 0 \Leftrightarrow s = 0 \text{ and } b = 1.$$

Thus, equation (78) holds in the case $b^* = 0$. Now we assume $b^* \neq 0$, hence

$$\begin{aligned} C_{\mathsf{rgb}} + C^*_{\mathsf{rgb}} &= b \cdot (E - s \cdot F) + b^* \cdot \Big(E - \frac{b \cdot s}{b^*} \cdot F^*\Big) \\ &= b \cdot E - b \cdot s \cdot F + b^* \cdot E - b \cdot s \cdot F^* \\ &= E - b \cdot s \cdot (F + F^* - E) \end{aligned}$$

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

$$F + F^* = E. \tag{79}$$

From

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

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

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

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

6.3.5 The Hsb model

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

$$(\mathit{hue}, \mathit{saturation}, \mathit{brightness}) := (\mathit{hue}^\circ/H, \mathit{saturation}, \mathit{brightness}) \tag{80}$$

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 :

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

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

6.3.6 The tHsb model

`\rangeHsb` `\rangetHsb`

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

$$x_1, y_1; x_2, y_2; \ldots; x_{h-1}, y_{h-1} \tag{82}$$

where

$$x_0 := 0 < x_1 < x_2 < \cdots < x_{h-1} < x_h := H \tag{83}$$

$$y_0 := 0 < y_1 < y_2 < \cdots < y_{h-1} < y_h := H \tag{84}$$

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

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

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

6.3.7 The gray model

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

$$(\mathit{red}, \mathit{green}, \mathit{blue}) := \mathit{gray} \cdot E \tag{86}$$

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

$$(\mathit{cyan}, \mathit{magenta}, \mathit{yellow}) := (1 - \mathit{gray}) \cdot E \tag{87}$$

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

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

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

$$(hue, saturation, brightness) := (0, 0, gray) \tag{89}$$

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

$$(hue^\circ, saturation, brightness) := (0, 0, gray) \tag{90}$$

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

$$Gray := \Gamma_N(gray) \tag{91}$$

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

$$gray^* := 1 - gray \tag{92}$$

6.3.8 The RGB model

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

$$(red, green, blue) := \big(\Delta_L(Red), \Delta_L(Green), \Delta_L(Blue)\big) \tag{93}$$

6.3.9 The HTML model

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

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

6.3.10 The HSB model

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

$$(hue, saturation, brightness) := \big(\Delta_M(Hue), \Delta_M(Saturation), \Delta_M(Brightness)\big) \tag{95}$$

6.3.11 The Gray model

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

$$gray := \Delta_N(Gray) \tag{96}$$

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

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

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

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

and finally

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

The intermediate colors (r, g, b) at the interval borders of equation (97) are well-known : for $\lambda = 380, 440, 490, 510, 580, 645$ we get *magenta*, *blue*, *cyan*, *green*, *yellow*, *red*, respectively. These turn out to be represented in the **hsb** model by $\mathit{hue} = \frac{5}{6}, \frac{4}{6}, \frac{3}{6}, \frac{2}{6}, \frac{1}{6}, \frac{0}{6}$, whereas $\mathit{saturation} = \mathit{brightness} = 1$ 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^γ`!black`, `red!`0.3^γ`!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

$$\varrho : \mathbb{R} \to [0, 1]\,, \;\; x \mapsto \big(\min\{1, \max\{0, x\}\}\big)^\gamma \tag{100}$$

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

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

$$saturation := 1 \tag{102}$$

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

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

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

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

Références

[1] Adobe Systems Incorporated : "PostScript Language Reference Manual". Addison-Wesley, third edition, 1999. `http://www.adobe.com/products/postscript/pdfs/PLRM.pdf`

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

[3] Donald Arseneau : url package, "2005/06/27 ver 3.2 Verb mode for urls, etc.". `CTAN/macros/latex/contrib/misc/url.sty`

[4] Dan Bruton : "Approximate RGB values for Visible Wavelengths", 1996. `http://www.physics.sfasu.edu/astro/color/spectra.html`

[5] David P. Carlisle : "Packages in the 'graphics' bundle", 2005. `CTAN/macros/latex/required/graphics/grfguide.*`

[6] David P. Carlisle : extension color, "2005/11/14 v1.0j Standard LaTeX Color". `CTAN/macros/latex/required/graphics/color.dtx`

[7] David P. Carlisle : colortbl package, "2001/02/13 v0.1j Color table columns". `CTAN/macros/latex/contrib/colortbl/`

[8] David P. Carlisle, Herbert Voß, Rolf Niepraschk : pstcol package, "2005/11/16 v1.2 LaTeX wrapper for 'PSTricks'". `CTAN/macros/graphics/pstricks/latex/pstcol.sty`

[9] Uwe Kern : "Chroma : a reference book of LaTeX colors". `CTAN/info/colour/chroma/` and `http://www.ukern.de/tex/chroma.html`

[10] Uwe Kern : xcolor package, "LaTeX color extensions". `CTAN/macros/latex/contrib/xcolor/` and `http://www.ukern.de/tex/xcolor.html`

[11] MiKTeX Project : `http://www.miktex.org/`

[12] Rolf Niepraschk : colorinfo package, "2003/05/04 v0.3c Info from defined colors". `CTAN/macros/latex/contrib/colorinfo/`

[13] Heiko Oberdiek : pdfcolmk package, "2006/02/20 v0.8 PDFtex COLor MarK". `CTAN/macros/latex/contrib/oberdiek/pdfcolmk.*`

[14] Sebastian Rahtz, Heiko Oberdiek : hyperref package, "2006/09/06 v6.75e Hypertext links for LaTeX". `CTAN/macros/latex/contrib/hyperref/`

[15] Alvy Ray Smith : "Color Gamut Transform Pairs". *Computer Graphics* (ACM SIGGRAPH), Volume 12, Number 3, August 1978. `http://alvyray.com/Papers/PapersCG.htm`

[16] World Wide Web Consortium : "HTML4 color keywords". `http://www.w3.org/TR/css3-color/#html4`

[17] World Wide Web Consortium : "Scalable Vector Graphics (SVG) 1.1 Specification — Basic Data Types and Interfaces". `http://www.w3.org/TR/SVG11/types.html#ColorKeywords`

Appendix

Acknowledgement

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

Trademarks

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

Known Issues

— `\rowcolors[\hline]...` does not work with longtable.

History

2007/01/21 v2.11

— New features :
  — color names *lime* and *teal* added to the set of predefined colors.
— Bugfix :
  — incorrect `\XC@strip@comma` call within hyperref-related options.

2006/11/28 v2.10

— New features :
  — `fixinclude` option prevents *dvips* from explicitly resetting current color to *black* before actually inserting an `.eps` file via `\color{red}\includegraphics{foo}`.
— Changes :
  — `\colorbox` and `\fcolorbox` made robust ;
  — obsolete package option `pst` removed ;
  — several changes to internal macros.
— Bugfixes :
  — incorrect processing of **cmyk**-type current color '.'.

2005/12/21 v2.09

— New features :
  — `\definecolor` and `\color` now accept space-separated color specifications, e.g., `\color [rgb]{1 .5 0}` ;
  — experimental `xcdraw` option extended to `pdftex` and `dvipdfm` drivers.
— Changes :
  — test file `xcolor2.tex` made compatible with recent changes in pstricks ;
  — test file `xcolor3.tex` extended ;
  — driver test file `xcolor4.tex` extended to demonstrate the different frame drawing approaches ;
  — more efficient implementation of driver-specific code.

2005/11/25 v2.08

— New features :
  — more flexibility for `\fcolorbox` arguments, e.g., `\fcolorbox [gray]{0.5}[wave]{580}{test}` ;
  — `\boxframe` returns a frame of given dimensions ;
  — new implementation of `\f(rame)box` and `\fcolorbox` as an extension of bug report latex/3655 to reduce pixel positioning errors in output devices ;
  — `kernelfbox` option for those who prefer the previous `\f(rame)box`

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

2005/11/12 v2.07

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

2005/10/15 v2.06

- New features :
  - color model **wave** for (approximate) visualisation of light wavelengths, still somewhat experimental ;
  - pseudo-model 'ps' for colors defined by literal PostScript code in conjunction with pstricks and `dvips` ; an illustrative example for a γ-correction approach is given in `xcolor2.tex` ;
  - `\substitutecolormodel` command for replacement of missing or faulty driver-specific color models ;
  - improved detection and handling of driver-specific color models ;
  - `dvipdfmx` and `xetex` options to support these drivers ;
  - generic driver test file `xcolor4.tex`.
- Changes :
  - `\XC@strip@comma` doesn't generate a trailing space anymore, which improves also the output of the `testcolors` environment.

2005/09/30 v2.05

- New features :
  - `testcolors` environment helps to test colors in different models, showing both the visual result and the model-specific parameters ;
  - `\extractcolorspecs` puts model/color specification into two separate commands, as opposed to `\extractcolorspec` ;
  - color names *pink* and *olive* added to the set of predefined colors.
- Bugfixes :
  - `\definecolor{foo}{named}{bar}` did not work in v2.04.

2005/09/23 v2.04

- New features :
  - preparation for usage of additional – driver-provided – color models ;

- pstricks users may now specify explicit color parameters within `\psset` and related commands, e.g., `\psset{linecolor=[rgb]{1,0,0}}` ; an illustrative example is given in `xcolor2.tex`.
- Changes :
  - color model names sanitized (i.e., turned to catcode 12) throughout the package ;
  - `\@namelet` command deprecated because of name clash with memoir — please use `\XC@let@cc` instead (more `\XC@let@..` commands are available as well) ;
  - simplified color conversion code by using the new `\XC@ifxcase` command ;
  - some minor changes to internal macros.

2005/06/06 v2.03

- New features :
  - `fixpdftex` option loads pdfcolmk package in order to improve pdfTEX's color behaviour during page breaks.
- Changes :
  - some minor changes to internal macros.
- Bugfixes :
  - due to an incorrect `\if` statement within `\XC@info`, `\colorlet` caused trouble whenever its second argument started with two identical letters, e.g., `\colorlet{rab}{oof}` ;
  - argument processing of `\XC@getcolor` caused incompatibility with msc package ;
  - `prologue` option caused incompatibility with preview package.

2005/03/24 v2.02

- New features :
  - `\aftergroupedef` command to reproduce `\aftergroupdef`'s behaviour prior to v2.01 ;
  - xcolor's homepage `www.ukern.de/tex/xcolor.html` now provides also a ready-to-run TDS-compliant archive containing all required files.
- Changes :
  - `\rowcolors` and friends are solely enabled by the `table` option ;
  - `\@ifxempty` changed back to more robust variant of v2.00.
- Bugfixes :
  - `\psset{linecolor=\ifcase\foo red\or green\or blue\fi}` did not work with pstricks (error introduced in v2.01).

2005/03/15 v2.01

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

— Changes :
  — higher accuracy : most complement calculations are now exact for all 5-digit decimals ;
  — `\rangeRGB` and similar variables may now be changed at any point in a document ;
  — `\aftergroupdef` now performs only a first-level expansion of its code argument ;
  — `\XCfileversion` and similar internal constants removed from `.sty` and `.def` files ;
  — improved memory management (reduced generation of 'multiletter control sequences' by `\@ifundefined` tests) ;
  — several internal macros improved and/or renamed.
— Bugfixes :
  — `\XC@getcolor` could cause unwanted spaces when `\psset` was used inside `pspicture` environments (pstricks) ;
  — arithmetic overflow could happen when too many decimal digits were used within color parameters, e.g., as a result of fp calculations.

2004/07/04 v2.00

— New features :
  — extended functionality for color expressions : mix colors like a painter ;
  — support for color blending : specify color mix expressions that are being blended with every displayed color ;
  — `\xglobal` command for selective control of globality for color definitions, blends, and masks ;
  — multiple step operations (e.g., `\color{foo!!+++}`) and access to individual members (e.g., `\color{foo!![7]}`) in color series ;
  — `\providecolor` command to define only non-existent colors ;
  — `\definecolorset` and `\providecolorset` commands to facilitate the construction of color sets with common underlying color model ;
  — additional 147 predefined color names according to SVG 1.1 specification ;
  — *xpdfborder* key for setting the width of hyperlink borders in a more driver-independent way if *dvips* is used.
— Changes :
  — extension color now completely integrated within xcolor ;
  — `override`, `usenames`, `nodvipsnames` options and `\xdefinecolor` command no longer needed ;
  — `dvips` and `dvipsnames` options now independent of each other ;
  — `\tracingcolors`'s behaviour changed to make it more versatile and reduce log file size in standard cases ;
  — `\rdivide`'s syntax made more flexible (divide by numbers and/or dimensions) ;
  — code restructured, some internal commands renamed ;
  — documentation rearranged and enhanced.
— Bugfixes :
  — `\definecolor{foo}{named}{bar}` did not work (error introduced in v1.11) ;
  — more robust behaviour of conditionals within pstricks key-values.

2004/05/09 v1.11

— New features :
  — switch `\ifglobalcolors` to control whether color definitions are global or local ;
  — option `hyperref` provides color expression support for the border colors of hyperlinks, e.g., `\hypersetup{xurlbordercolor=red!50!yellow}` ;
  — internal hooks `\XC@bcolor`, `\XC@mcolor`, and `\XC@ecolor` for additional code that has to be executed immediately before/after the

current color is being displayed.
- Changes :
  - `\XC@logcolor` renamed to `\XC@display`, which is now the core color display command ;
  - improved interface to pstricks.

2004/03/27 v1.10

- New features :
  - support for 'named' model ;
  - support for *dvips* colors (may now be used within color expressions) ;
  - internal representation of 'ordinary' and 'named' colors merged into unified data structure ;
  - allow multiple '-' signs at the beginning of color expressions.
- Bugfixes :
  - commands like `\color[named]{foo}` caused errors when color masking or target model conversion were active ;
  - incompatibility with soul package : commands `\hl`, `\ul`, etc. could yield unexpected results.
- Documentation :
  - added formula for general color expressions ;
  - enhanced text and index ;
  - removed dependence of index generation on local configuration file.

2004/02/16 v1.09

- New features :
  - color model **HTML**, a 24-bit hexadecimal **RGB** variant ; allows to specify colors like `\color[HTML]{AFFE90}` ;
  - color names *orange*, *violet*, *purple*, and *brown* added to the set of predefined colors.
- New xcolor homepage : `www.ukern.de/tex/xcolor.html`
- Bugfix : `\xdefinecolor` sometimes did not normalise its parameters.
- Changes :
  - slight improvements of the documentation ;
  - example file `xcolor1.tex` reorganised and abridged.

2004/02/04 v1.08

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

2004/01/20 v1.07

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

2003/12/15 v1.06

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

2003/11/21 v1.05

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

2003/11/09 v1.04

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

2003/09/21 v1.03

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

2003/09/19 v1.02

— Change : `\extractcolorspec` and `\colorlet` now also accept color series as arguments.

2003/09/15 v1.01

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

2003/09/09 v1.00

— First published release.

✖

Index

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