Die TEXnische Komödie

DANTE
Deutschsprachige
Anwendervereinigung TEX e.V.

Uwe Kern: *Farbspielereien in LATEX mit dem* `xcolor`*-Paket*, Die TEXnische Komödie 2/2004, S. 35–53.

Die TEXnische Komödie ist die Mitgliedszeitschrift von DANTE, Deutschsprachige Anwendervereinigung TEX e.V. Einzelne Hefte können von Mitgliedern bei der Geschäftsstelle von DANTE, Deutschsprachige Anwendervereinigung TEX e.V. erworben werden. Mitglieder erhalten Die TEXnische Komödie im Rahmen ihrer Mitgliedschaft.

Farbspielereien in LATEX mit dem `xcolor`-Paket

Uwe Kern

Das `xcolor`-Paket ermöglicht die einfache Nutzung verschiedener Arten von Farbschattierungen und -mischungen sowie die Erzeugung von Farbenfolgen und alternierend gefärbten Tabellenzeilen. Es stellt weiterhin Routinen für die (manuelle oder automatische) Konvertierung zwischen verschiedenen Farbmodellen zur Verfügung. Dieser Artikel erläutert anhand von Beispielen Anwendungen des Paketes.

Einleitung

Das `color`-Paket [2] stellt eine Reihe nützlicher Funktionen zur konsistenten und treiberunabhängigen Handhabung von Farben in (pdf)LATEX zur Verfügung. Es unterstützt dabei, allerdings nicht ganz treiberunabhängig, mehrere Farbmodelle.

Gleichwohl ist der Umgang mit Farben in (pdf)LATEX manchmal etwas umständlich, insbesondere wenn leichte Farbveränderungen, Mischfarben oder gar Konvertierungen ins Spiel kommen: Hier ist man häufig auf andere Programme angewiesen, welche die erforderlichen Parameter berechnen, die an-

schließend in einen `\definecolor`-Befehl hineinkopiert werden. Hin und wieder kommt auch der gute alte Taschenrechner zum Einsatz, wenn es um Aufgaben der folgenden Art geht:

- Beim Umgang mit Druckereien lernt man, dass es wesentlich preisgünstiger sein kann, eine speziell definierte Farbe (*spot color*) – etwa aus einem Farbfächer – in verschiedenen Helligkeitsstufen zu verwenden, als auf die Standarddruckfarben *cyan*, *magenta*, *yellow* und deren Mischungen zurückzugreifen. Wie sieht nun aber die 75%-Version der mühsam ausgesuchten Spezialfarbe aus?
- Man sieht eine gefällige Farbe, die man gerne selbst verwenden möchte. Leider ist sie im hsb-Modell definiert, man selbst arbeitet jedoch mit pdfLaTeX ohne hsb-Unterstützung. Was tun?
- Wie sieht eigentlich eine Mischung aus 40% *green* und 60% *yellow* aus? (Antwort: 40% + 60% =)
- Und wie sieht die Komplementärfarbe davon aus? (Antwort:)
- Die Druckerei verlangt, dass alle Farbdefinitionen im Dokument in das cmyk-Format transformiert werden sollen. Wie kann man die Berechnungen effizient durchführen?
- Die 50 Zeilen einer Tabelle sollen alternierend gefärbt werden. Geht das auch einfacher, als 50 Kopien des `\rowcolor`-Befehls in den Quelltext einzufügen?

Das `xcolor`-Paket [8] bietet Lösungen für diese und andere Fragestellungen. Neben Kompatibilität zu `color` und den zugehörigen Treiberdefinitionen wurde vor allem auf eine möglichst einfache Bedienbarkeit Wert gelegt. Dazu gehört der weitgehende Verzicht auf explizite Farbdefinitionen – die direkte Anwendung steht im Vordergrund. Wie das geht, wird weiter unten beschrieben.

Technisches

Installation

Die Installation besteht darin, die Datei `xcolor.sty` an einen Ort zu kopieren, wo sie von (pdf)LaTeX gefunden wird. Dann kann `xcolor` einfach anstelle von `color` benutzt werden. Der allgemeine Befehl zur Aktivierung

Tabelle 1: Die Paketoptionen von `xcolor`

Option	*Beschreibung*
`natural`	(Default) Belasse alle Farben in ihrem Farbmodell, außer RGB (konvertiert in rgb), HSB (konvertiert in hsb) und Gray (konvertiert in gray).
`rgb`	Konvertiere alle Farben in das rgb-Modell.
`cmy`	Konvertiere alle Farben in das cmy-Modell.
`cmyk`	Konvertiere alle Farben in das cmyk-Modell.
`hsb`	Konvertiere alle Farben in das hsb-Modell.
`gray`	Konvertiere alle Farben in das gray-Modell. Insbesondere nützlich, um die Ausgabe eines Schwarz-Weiß-Druckers zu simulieren.
`RGB`	Konvertiere alle Farben in das RGB-Modell (und hinterher nach rgb).
`HSB`	Konvertiere alle Farben in das HSB-Modell (und hinterher nach hsb).
`Gray`	Konvertiere alle Farben in das Gray-Modell (und hinterher nach gray).
`pst`	Lade das `pstcol`-Paket [5], um die „normalen“ Farbdefinitionen innerhalb von `pstricks`-Makros verwenden zu können.
`table`	Lade das `colortbl`-Paket [4], um Zeilen, Spalten und Zellen innerhalb von Tabellen einzufärben.
`override`	Ersetze den ursprünglichen `\definecolor`-Befehl durch die Definition von `\xdefinecolor`.
`showerrors`	(Default) Zeige eine Fehlermeldung an, falls eine undefinierte Farbe verwendet wird (entspricht dem Verhalten des `color`-Pakets).
`hideerrors`	Zeige eine Warnmeldung an, falls eine undefinierte Farbe verwendet wird und ersetze diese Farbe durch *black*.

lautet demnach `\usepackage[`⟨*options*⟩`]{xcolor}` in der Dokumentenpräambel. Hierbei werden mit ⟨*options*⟩ die im folgenden Abschnitt aufgeführten Paketoptionen bezeichnet.

Paketoptionen

Die Paketoptionen lassen sich in vier Klassen einteilen:

- Optionen, die an das `color`-Paket weitergereicht werden (beispielsweise `dvips`, `pdftex`),
- Optionen, die das dokumentenweite Farbmodell festlegen (`natural`, `rgb`, `cmy`, `cmyk`, `hsb`, `gray`, `RGB`, `HSB`, `Gray`),
- Optionen, die andere Pakete laden (`pst`, `table`),
- Optionen, die das Verhalten bestimmter Befehle beeinflussen (`override`, `showerrors`, `hideerrors`).

Die `xcolor`-spezifischen Paketoptionen sind in Tabelle 1 beschrieben.

Farbmodelle

Eine umfassende Erklärung oder gar Herleitung von Farbmodellen berührt Gebiete wie Physik, Biologie und Drucktechnik und geht weit über das Ziel dieses Artikels hinaus. Daher seien aus der Fülle der vorhandenen Literatur als Einführung die Bücher [1] und [7] empfohlen.

Wir vertreten hier einen schlichten, algorithmisch-technokratischen Standpunkt: Farbmodelle sind ein- oder mehrdimensionale numerische Repräsentationen bestimmter physikalischer Effekte (nämlich Farben), die – mit gewissen Einschränkungen – rechnerisch ineinander überführt werden können.

Die unterstützten Farbmodelle sind in Tabelle 2 aufgeführt. Es sei betont, dass diese Farbmodelle unabhängig vom ausgewählten Treiber zur Verfügung gestellt werden. Jedoch geben manche Treiber (beispielsweise `dvipdfm`) nur vor, das hsb-Modell zu unterstützen, sodass eine Strategie implementiert wurde, die dieses Verhalten korrigiert. Die tatsächlich von `xcolor` hinzugefügten Farbmodelle kann man nach einem LaTeX-Lauf der `log`-Datei entnehmen.

Für die „ganzzahligen“ Modelle RGB, HSB und Gray können die in Tabelle 2 angegebenen Konstanten L, M, N in einfacher Weise geändert werden. Dazu dienen die Befehle `\def\rangeRGB{`$\langle L\rangle$`}`, `\def\rangeHSB{`$\langle M\rangle$`}` und `\def\rangeGray{`$\langle N\rangle$`}`, die allerdings *vor* dem Laden von `xcolor` auszuführen sind.

Tabelle 2: Unterstützte Farbmodelle

Name	*Basisfarben/Begriffe*	*Wertebereich*	*Default*
rgb	*red, green, blue*	$[0,1]^3$	
cmy	*cyan, magenta, yellow*	$[0,1]^3$	
cmyk	*cyan, magenta, yellow, black*	$[0,1]^4$	
hsb	*hue, saturation, brightness*	$[0,1]^3$	
gray	*gray*	$[0,1]$	
RGB	*Red, Green, Blue*	$\{0,1,\ldots,L\}^3$	$L = 255$
HSB	*Hue, Saturation, Brightness*	$\{0,1,\ldots,M\}^3$	$M = 240$
Gray	*Gray*	$\{0,1,\ldots,N\}$	$N = 15$

L, M, N sind natürliche Zahlen

Definition von Farben

`\xdefinecolor{`⟨*name*⟩`}{`⟨*model*⟩`}{`⟨*color specification*⟩`}` ist der neue und zentrale Befehl zur Definition von Farben, für welche die erweiterten Möglichkeiten von `xcolor` genutzt werden sollen. Er wird verwendet wie bisher `\definecolor` und ersetzt dieses Kommando, das gleichwohl weiterhin mit seiner ursprünglichen Bedeutung verfügbar ist. Jedoch kann man es per `\let\definecolor=\xdefinecolor` oder mit der Paketoption `override` „überbügeln“, sofern nicht das Farbmodell *named* genutzt werden soll, das von `\xdefinecolor` nicht unterstützt wird.

In `xcolor` werden die folgenden Farben mittels `\xdefinecolor` standardmäßig (re)definiert: *red*, *green*, *blue*, *cyan*, *magenta*, *yellow*, *black*, *white*, *darkgray*, *gray*, *lightgray*.

`\colorlet{`⟨*name*⟩`}[`⟨*model*⟩`]{`⟨*color expression*⟩`}` kopiert die zum Ausführungszeitpunkt aktuelle Bedeutung von ⟨*color expression*⟩ nach ⟨*name*⟩ und definiert damit eine neue Farbe bzw. überschreibt eine bereits vorhandene dieses Namens. Bei nicht leerem ⟨*model*⟩-Parameter wird ⟨*color expression*⟩ zuvor noch in das spezifizierte Farbmodell transformiert. Die neue Farbe ⟨*name*⟩ kann sodann selbstverständlich wieder in anderen Farbausdrücken benutzt werden. Beispielsweise wurde am Anfang dieses Artikels durch das Kommando `\colorlet{tableheadcolor}{gray!25}` eine Farbe definiert, die in den Tabellen per `\rowcolor{tableheadcolor}` zur Färbung der ersten Tabellenzeile Verwendung findet.

Expressionen

Farbausdrücke

Aus Kompatibilitätsgründen gestattet das `xcolor`-Paket die Verwendung der in `color` bereitgestellten Methoden zur Definition benannter Farben. Um daraus resultierende Begriffsverwirrungen zu vermeiden, wird im Folgenden immer unterschieden zwischen

- *Standardfarben*, die direkt oder indirekt über den `\definecolor`-Befehl definiert werden (dabei wäre eine indirekte Definition von „bar“ etwa `\colorlet{bar}{foo}` nach `\definecolor{foo}...`), und
- *erweiterten Farben*, die direkt oder indirekt über die neuen Befehle `\xdefinecolor` oder `\definecolorseries` definiert werden.

Die *aktuelle Farbe*, repräsentiert durch den reservierten Farbnamen „.“ (ohne Anführungszeichen), wird ebenfalls als *erweiterte Farbe* betrachtet. Falls die Paketoption `override` verwendet wird, sind natürlich sämtliche Farben automatisch erweiterte Farben.

Triviale Farbausdrücke

Ein trivialer Farbausdruck ist

$$\langle \textit{color expression} \rangle = \langle \textit{name} \rangle,$$

wobei $\langle \textit{name} \rangle$ den Namen einer *Standardfarbe* oder einer *erweiterten Farbe* bezeichnet.

Nichttriviale Farbausdrücke

Die allgemeine Form eines nichttrivialen Farbausdrucks ist

$$\langle \textit{color expression} \rangle = \langle \textit{prefix} \rangle \langle \textit{name} \rangle \langle \textit{mix expression} \rangle \langle \textit{postfix} \rangle$$

mit folgenden Bestandteilen:

- $\langle \textit{prefix} \rangle$ ist entweder ein leerer String oder ein Minuszeichen „-“ (ohne Anführungszeichen); das Minuszeichen bewirkt, dass die aus dem restlichen Ausdruck resultierende Farbe in ihre Komplementärfarbe konvertiert wird;
- $\langle \textit{name} \rangle$ ist der Name einer *erweiterten Farbe*;

- ⟨*mix expression*⟩ ist entweder ein *vollständiger* oder ein *unvollständiger* Mix-Ausdruck, wie unten beschrieben;
- ⟨*postfix*⟩ ist entweder ein leerer String oder der String „`!!+`“ (ohne Anführungszeichen); letzterer Fall bedarf noch folgender Voraussetzungen:
  - ⟨*name*⟩ ist der Name einer *Farbenfolge*,
  - ⟨*mix expression*⟩ ist ein *vollständiger* Mix-Ausdruck (siehe unten)

  und bewirkt, dass im Anschluss an die Auswertung und Anzeige des Farbausdrucks für die Farbenfolge ⟨*name*⟩ ein Induktionsschritt durchgeführt wird.

Vollständige Mix-Ausdrücke

Die allgemeine Form eines vollständigen Mix-Ausdrucks ist entweder ein leerer String oder

$$\langle \textit{mix expression} \rangle = !\langle num_1 \rangle !\langle name_1 \rangle !\langle num_2 \rangle !\langle name_2 \rangle ! \ldots !\langle num_n \rangle !\langle name_n \rangle$$

wobei gilt

- $n \geq 1$ ist eine ganze Zahl;
- jeder ⟨num_i⟩-Parameter ist eine reelle Zahl aus dem Intervall $[0, 100]$, also $0 \leq \langle num_i \rangle \leq 100$;
- ⟨$name_i$⟩ ist jeweils der Name einer *erweiterten Farbe*.

Unvollständige Mix-Ausdrücke

Ein unvollständiger Mix-Ausdruck ist einfach eine Abkürzung:

$$\begin{aligned} \langle \textit{mix expression} \rangle &= !\langle num_1 \rangle !\langle name_1 \rangle !\langle num_2 \rangle !\langle name_2 \rangle ! \ldots !\langle num_n \rangle \\ &= !\langle num_1 \rangle !\langle name_1 \rangle !\langle num_2 \rangle !\langle name_2 \rangle ! \ldots !\langle num_n \rangle !\texttt{white} \end{aligned}$$

Bedeutung von Farbausdrücken

Hier folgt nun die Erklärung, wie ein Ausdruck der Form

$$\langle prefix \rangle \langle name \rangle !\langle num_1 \rangle !\langle name_1 \rangle !\langle num_2 \rangle ! \ldots !\langle num_n \rangle !\langle name_n \rangle \langle postfix \rangle$$

interpretiert und verarbeitet wird:

1. Zunächst werden die Modell- und Farbparameter von ⟨*name*⟩ ermittelt; damit wird eine temporäre Farbe ⟨*temp*⟩ definiert.
2. Dann wird eine Farbmischung, bestehend aus ⟨num_1⟩% von Farbe ⟨*temp*⟩ und (100 − ⟨num_1⟩)% von Farbe ⟨$name_1$⟩ ausgerechnet; dies ergibt die neue temporäre Farbe ⟨*temp*⟩.
3. Der vorige Schritt wird wiederholt für alle verbleibenden Parameterpaare (⟨num_2⟩,⟨$name_2$⟩), ..., (⟨num_n⟩,⟨$name_n$⟩).
4. Falls ⟨*prefix*⟩ nicht leer ist, wird ⟨*temp*⟩ in seine Komplementärfarbe transformiert.
5. Falls ⟨*postfix*⟩ nicht leer ist, wird der entsprechende Induktionsschritt für die Farbenfolge ⟨*name*⟩ ausgeführt.
6. Nun wird die Farbe ⟨*temp*⟩ angezeigt oder dient als Input für andere Operationen, je nachdem, welches Kommando den ursprünglichen Farbausdruck aufgerufen hat.

In einem typischen Ausdruck in Schritt 2, also ⟨*temp*⟩!⟨num_i⟩!⟨$name_i$⟩, wird in den Spezialfällen mit ⟨num_i⟩=100 bzw. ⟨num_i⟩=0 die Farbe ⟨*temp*⟩ bzw. ⟨$name_i$⟩ ohne weitere Transformationen verwendet. Im echten Mischfall $0 <$ ⟨num_i⟩ < 100 können natürlich die beiden beteiligten Farben in unterschiedlichen Modellen definiert sein, etwa `\xdefinecolor{foo}{rgb}{...}` und `\xdefinecolor{bar}{cmyk}{...}`. Dann wird in der Regel die zweite Farbe ⟨$name_i$⟩ in das Modell der ersten Farbe ⟨*temp*⟩ transformiert; danach wird die Mischung in diesem Modell berechnet.[1] Infolgedessen werden die Ausdrücke ⟨*temp*⟩!⟨num_i⟩!⟨$name_i$⟩ und ⟨$name_i$⟩!⟨100−num_i⟩!⟨*temp*⟩, obwohl theoretisch gleichwertig, in der Praxis nicht notwendigerweise zum selben optischen Ergebnis führen.

Verwendung der Erweiterungen

Die im `color`-Paket definierten Farbbefehle (`\color`, `\textcolor`, `\colorbox`, `\fcolorbox`, `\pagecolor`) können wie üblich benutzt werden. Allerdings wird durch `xcolor` die Möglichkeit geschaffen, überall dort, wo *Namen* von Farben als Argumente erwartet werden, *Farbausdrücke* zu verwenden, wie das folgende Beispiel zeigt (mit `\fboxrule=3pt`):

[1] Ausnahme: Um seltsame Ergebnisse zu vermeiden, wird diese Regel umgekehrt, sofern ⟨*temp*⟩ im gray-Modell definiert ist; in diesem Fall erfolgt eine Konvertierung in das zu ⟨$name_i$⟩ gehörende Farbmodell.

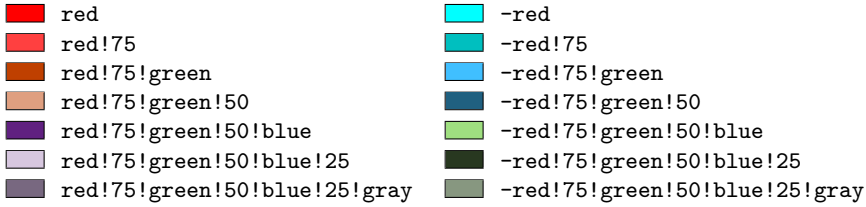


Abbildung 1: Anwendungsbeispiele für Farbausdrücke

`\fcolorbox{red!70!green}{-red!70!green}{Test}`

Es sei hier noch einmal darauf hingewiesen, dass die erweiterten Möglichkeiten nur für *erweiterte Farben* verwendet werden können, die also direkt oder indirekt per `\xdefinecolor` definiert worden sind. Daneben können, wie bei `\color[`⟨*model*⟩`]{`⟨*specification*⟩`}`, Farbspezifikationen wie gewohnt direkt angegeben werden, siehe die Dokumentation [3].

Anwendungsbeispiele für Farbausdrücke sind in den Abbildungen 1 und 2 bis 4 zu sehen. Dabei wurden die folgenden Definitionen verwendet:

```
\xdefinecolor{MyGreen}{cmyk}{0.92,0,0.87,0.09}
\colorlet{MyGreen-hsb}[hsb]{MyGreen}
\colorlet{MyGreen-gray}[gray]{MyGreen}
```

Zugriff auf die aktuelle Farbe

Innerhalb eines Farbausdrucks dient „.“ als Platzhalter für die aktuelle Farbe. Beispielsweise erzeugt

```
{Test \color{.!75}Test \color{.!75}Test \color{.!75}Test
 \color{.!75}Test \color{.!75}Test}.
```

die Ausgabe Test Test Test Test Test Test. Es ist auch möglich, die aktuelle Farbe für die spätere Verwendung zwischenzuspeichern, etwa mit dem Kommando `\colorlet{foo}{.}`.

In manchen Fällen ist die aktuelle Farbe von eher begrenztem Nutzen: So impliziert etwa die interne Konstruktion einer `\fcolorbox`, dass zum Zeitpunkt der Auswertung des Ausdrucks für die Hintergrundfarbe inzwischen die Rahmenfarbe lokal zur aktuellen Farbe geworden ist – und diese muss rein gar nichts mit der aktuellen Farbe *außerhalb* der Box zu tun haben.

`color`-Definition: `0.92 0 0.87 0.09 k 0.92 0 0.87 0.09 K`
`xcolor`-Definition: `{cmyk}{0.92,0,0.87,0.09}`

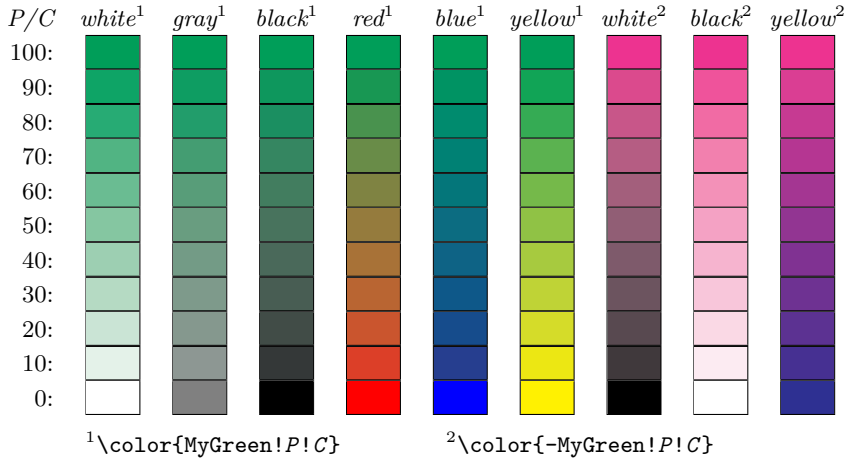


Abbildung 2: Farbtöne – „MyGreen“

`color`-Definition: `0 0.91 0.03995 rg 0 0.91 0.03995 RG`
`xcolor`-Definition: `{hsb}{0.34065,1,0.91}`

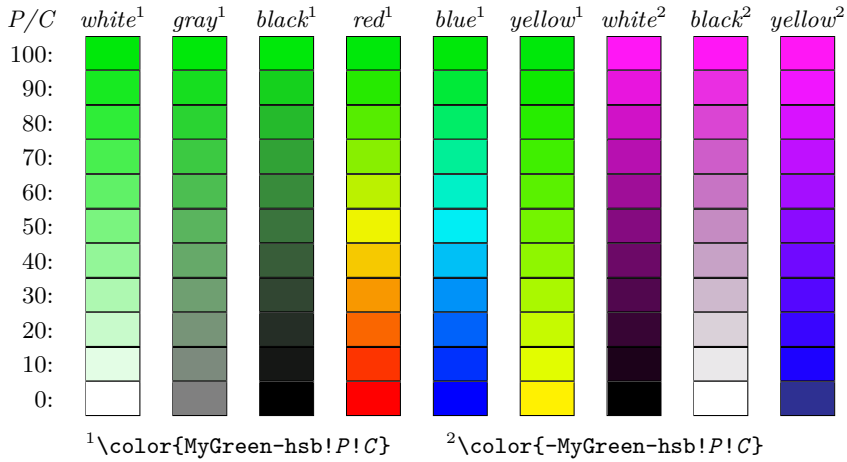


Abbildung 3: Farbtöne – „MyGreen-hsb“

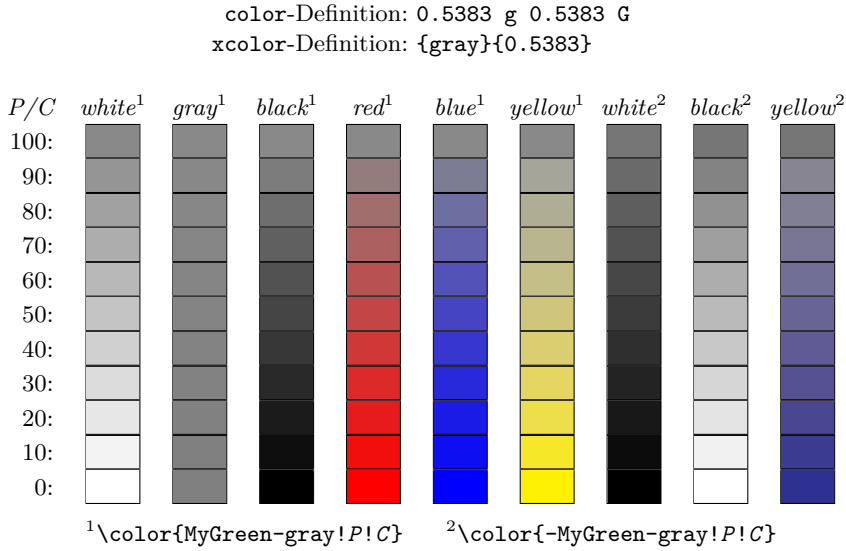


Abbildung 4: Farbtöne – „MyGreen-gray“

Informationsgesellschaft

Beschaffung von Informationen ...

Das `color`-Paket verlässt den Pfad der Treiberunabhängigkeit unmittelbar bei der Definition einer Farbe: `\definecolor{foo}{rgb}{.8,.6,.4}` erzeugt intern ein Makro `\color@foo`, das im Falle des Treibers `dvips` den Inhalt „`rgb .8 .6 .4`“, bei `pdftex` jedoch „`.8 .6 .4 rg .8 .6 .4 RG`“, hat. Es bedarf externer Hilfe in Gestalt des Pakets `colorinfo` [9], um hieraus wieder allgemein verwendbare Informationen zu erzeugen.

Dahingegen erzeugt `\xdefinecolor{foo}{rgb}{.8,.6,.4}` intern ein Makro `\xcolor@foo` mit dem Inhalt „`\@xcolor@{rgb}{0.8,0.6,0.4}`“, es sei denn, per Paketoption wurde ein anderes Zielmodell festgelegt.

`\extractcolorspec{`⟨*color expression*⟩`}{`⟨*cmd*⟩`}` dient dazu, die genauen Parameter eines Farbausdrucks in ein Hilfsmakro zu schreiben, siehe dazu das Beispiel im nächsten Abschnitt.

`\tracingcolors=`⟨*num*⟩ steuert, wieviel Information über die verwendeten Farben in die Protokolldatei geschrieben wird.

. . . und deren Verarbeitung

`\convertcolorspec`
`{`⟨*source model*⟩`}{`⟨*color specification*⟩`}{`⟨*target model*⟩`}{`⟨*cmd*⟩`}`
konvertiert eine Farbe, die durch die numerische ⟨*color specification*⟩ im Modell ⟨*source model*⟩ gegeben ist, in das Modell ⟨*target model*⟩ und speichert die neue Farbspezifikation im Makro ⟨*cmd*⟩. Dabei können ⟨*source model*⟩ und ⟨*target model*⟩ beliebige Modelle aus Tabelle 2 sein. Diesen Befehl kann man auch verwenden, wenn man eine Konvertierung vornehmen will, ohne das Ausgangsmodell explizit zu kennen:

```
\extractcolorspec{red!75!green!50}\tmpa
\expandafter\convertcolorspec\tmpa{hsb}\tmpb
\expandafter\convertcolorspec\tmpa{RGB}\tmpc
```

liefert die Parameter des Ausdrucks „`red!75!green!50`“:

```
\tmpa: {rgb}{0.875,0.625,0.5}
\tmpb: 0.05557,0.42857,0.875
\tmpc: 223,159,128
```

Folgen von Farben

Automatische Farbgenerierung kann in grafischen Anwendungen nützlich sein, bei denen eine möglicherweise große und unspezifizierte Anzahl an Farben benötigt wird und der Anwender nicht jede einzelne Farbe festlegen will oder kann. Dafür wird hier der Begriff der *Farbenfolge* eingeführt, bestehend aus einer Basisfarbe und einem Schema, wie aus der aktiven die nächste Farbe konstruiert wird. Es liegt also ein induktives Verfahren vor.

Die praktische Anwendung besteht aus drei Teilen: Definition einer Farbenfolge (üblicherweise einmal pro Dokument), Initialisierung der Folge (potentiell mehrfach) und Anwendung – mit oder ohne Voranschreiten – der aktiven Farbe (potentiell vielfach).

Definition einer Farbenfolge

`\definecolorseries`
`{`⟨*name*⟩`}{`⟨*model*⟩`}{`⟨*method*⟩`}[`⟨*b-model*⟩`]{`⟨*b-spec*⟩`}[`⟨*s-model*⟩`]{`⟨*s-spec*⟩`}`
definiert eine Farbenfolge ⟨*name*⟩, deren Schrittberechnungen innerhalb des Farbmodells ⟨*model*⟩ (rgb, cmy, cmyk, hsb oder gray) durchgeführt werden, wobei ⟨*method*⟩ den gewünschten Algorithmus auswählt (`step`, `grad` oder `last`, siehe unten). Weitere Details werden durch die restlichen Argumente festgelegt:

- `[`⟨*b-model*⟩`]{`⟨*b-spec*⟩`}` spezifiziert die Basisfarbe *base* im Algorithmus entweder direkt, etwa `[rgb]{1,0.5,0.5}`, oder als Farbausdruck, etwa `{-yellow!50}`, wenn das optionale Argument fehlt.
- `[`⟨*s-model*⟩`]{`⟨*s-spec*⟩`}` spezifiziert, wie der Schrittvektor *step* im ausgewählten Algorithmus `method` berechnet wird:
  - `step`, `grad`: Das optionale Argument ist bedeutungslos und ⟨*s-spec*⟩ ist ein Parametervektor, dessen Dimension durch ⟨*model*⟩ determiniert wird, beispielsweise `{0.1,-0.2,0.3}` im Falle von rgb, cmy oder hsb.
  - `last`: Die letzte Farbe der Folge wird entweder direkt spezifiziert, also `[rgb]{1,0.5,0.5}`, oder als Farbausdruck, also `{-yellow!50}`, falls das optionale Argument fehlt.

Das allgemeine Schema lautet

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

für $n = 1, 2, \ldots$, wobei U beliebige reelle m-Vektoren in den m-Einheitswürfel abbildet:

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

Somit liefert jeder Schritt des Algorithmus eine zulässige Farbe mit Parametern aus dem Intervall $[0, 1]$.

Die unterschiedlichen Methoden benutzen unterschiedliche Varianten zur Berechnung des Schrittvektors *step*:

- `step`, `grad`: Das letzte Argument `{`⟨*s-spec*⟩`}` definiert den Richtungsvektor *grad*.
- `last`: `{`⟨*s-spec*⟩`}` bzw. `[`⟨*s-model*⟩`]{`⟨*s-spec*⟩`}` definiert den Parametervektor der Farbe *last*.

Durch den Befehl `\resetcolorseries` wird schließlich der tatsächliche Schrittvektor *step* ausgerechnet:

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

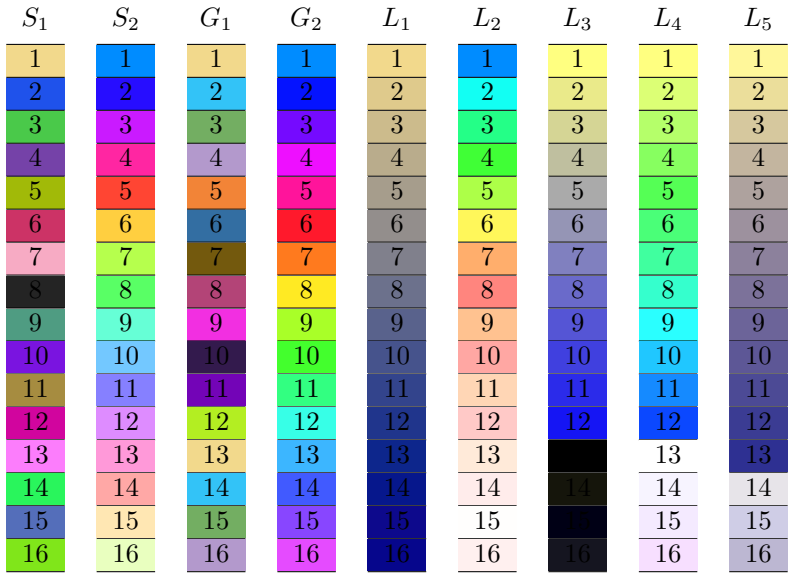
Bei der Definition einer Farbenfolge darf auch der Platzhalter „.“, der für die aktuelle Dokumentenfarbe steht, verwendet werden. Beispielsweise wird durch `\definecolorseries{foo}{rgb}{last}{.}{-.}` eine Folge definiert, die mit der aktuellen Farbe beginnt und mit deren Komplementärfarbe endet. Allerdings wird hier die konkrete Farbdefinition zur Ausführungszeit von `\definecolorseries` verwendet (entsprechend dem `\let`-Befehl in TeX); es besteht kein Zusammenhang mit irgendeiner Farbe, die an einer anderen Stelle des Dokuments durch „.“ repräsentiert wird.

Initialisierung einer Farbenfolge

`\resetcolorseries[`⟨*cycle*⟩`]{`⟨*name*⟩`}` ist ein Kommando, das mindestens einmal ausgeführt werden muss, um die Farbenfolge ⟨*name*⟩ benutzen zu können. Es setzt die aktive Farbe der Folge auf den Basiswert zurück und berechnet – im Falle der Methoden `grad` und `last` – den tatsächlichen Schrittvektor gemäß der Größe ⟨*cycle*⟩, einer von Null verschiedenen reellen Zahl, und Gleichung (1). Falls das optionale Argument leer ist, wird der im Makro `\colorseriescycle` gespeicherte Wert herangezogen (Default: 16). Das optionale Argument wird im Falle der `step`-Methode ignoriert.

Anwendung einer Farbenfolge

Um die aktive Farbe einer Farbenfolge anzuzeigen, kann der Name der Folge wie ein gewöhnlicher Farbname in Ausdrücken verwendet und damit per `\color`, `\textcolor`, ... aufgerufen werden. Benutzt man darüber hinaus die weiter vorne beschriebene spezielle Syntax mit dem ⟨*postfix*⟩ „`!!+`“, dann wird durch `\color{`⟨*name*⟩`!!+}` etc. nicht nur die Farbe angezeigt, sondern auch ein Induktionsschritt ausgeführt. In diesem Fall wird also die aktive Farbe der Folge geändert. In Abbildung 5 werden verschiedene Ausprägungen von Farbenfolgen demonstriert.



	Individuelle Definitionen
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}`
	Gemeinsame Definitionen

```
\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}
```

Abbildung 5: Beispiele für Farbenfolgen

Um dem Titel dieses Artikels gerecht zu werden, sei hier noch ein weiteres Beispiel (Abbildung 6) mit Quelltext gezeigt. Es ist im Wesentlichen der Dokumentation [6] entnommen.

```
\documentclass{article}
\usepackage[pst]{xcolor}
\pagestyle{empty}
\newcommand*\SheepHead
 {\begin{pspicture}(3,1.5)
    \pscustom[liftpen=2,fillstyle=solid,fillcolor=sheep!!+]{%
      \pscurve(0.5,-0.2)(0.6,0.5)(0.2,1.3)(0,1.5)(0,1.5)
        (0.4,1.3)(0.8,1.5)(2.2,1.9)(3,1.5)(3,1.5)(3.2,1.3)
        (3.6,0.5)(3.4,-0.3)(3,0)(2.2,0.4)(0.5,-0.2)}
    \pscircle*(2.65,1.25){0.12\psunit}% Eye
    \psccurve*(3.5,0.3)(3.35,0.45)(3.5,0.6)(3.6,0.4)% Muzzle
    % Mouth
    \pscurve(3,0.35)(3.3,0.1)(3.6,0.05)
    % Ear
    \pscurve(2.3,1.3)(2.1,1.5)(2.15,1.7)
    \pscurve(2.1,1.7)(2.35,1.6)(2.45,1.4)
  \end{pspicture}}
\newcommand*\Sheep
 {\SheepHead\SheepHead\SheepHead\SheepHead\SheepHead}
\definecolorseries{sheep}{rgb}{step}[rgb
  ]{.95,.85,.55}{.17,.47,.37}
\resetcolorseries{sheep}
\begin{document}
\psset{unit=0.5}
\begin{pspicture}(-8,-1.5)(8.5,7.5)
  \rput(0,6){\Sheep}
  \rput(0,4.5){\Sheep}
  \rput(0,3){\Sheep}
  \rput(0,1.5){\Sheep}
  \rput(0,0){\Sheep}
\end{pspicture}
\end{document}
```

Unterschiede zwischen Farben und Farbenfolgen

Obwohl sie sich bei der Anwendung innerhalb von Farbausdrücken gleichartig verhalten, sind die durch `\xdefinecolor` und `\definecolorseries` definierten Objekte grundsätzlich verschieden hinsichtlich ihres Gültigkeitsbereiches: `\definecolor` und `\xdefinecolor` erzeugen stets *lokale* Farben, wohingegen `\definecolorseries` *globale* Objekte generiert (sonst wäre es ziemlich unbequem, den Induktionsmechanismus innerhalb von Tabellen oder Grafiken

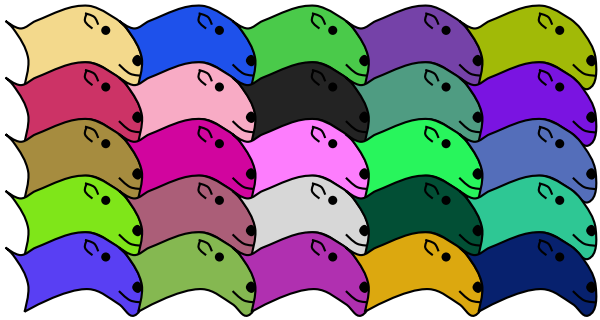


Abbildung 6: Mit Farbenfolgen eingefärbte Schafherde

anzuwenden). Nimmt man beispielsweise an, dass „bar“ eine undefinierte Farbe ist, so kann man nach

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

Befehle wie `\color{foo}` oder `\color{foo!!+}` uneingeschränkt verwenden, während `\color{bar}` unweigerlich zu einer Fehlermeldung führt. Jedoch kann man vermöge `\colorlet{bar}{foo}` oder `\colorlet{bar}{foo!!+}` die aktive Farbe einer Farbenfolge lokal speichern – ohne oder mit Induktionsschritt.

Zeilenweise

Zur Erzeugung farbiger Zeilen, Spalten oder Zellen in Tabellen hat sich das Paket `colortbl` [4] bewährt. In `xcolor` wird darauf aufbauend die Möglichkeit zur Verfügung gestellt, Tabellenzeilen automatisch alternierend (also ungerade und gerade Zeilen jeweils einheitlich) zu färben.

`\rowcolors[`⟨*commands*⟩`]%`
`{`⟨*num*⟩`}{`⟨*odd-row color expression*⟩`}{`⟨*even-row color expression*⟩`}`
muss ausgeführt werden, *bevor* die Tabelle beginnt. ⟨*num*⟩ gibt an, welches die erste gemäß dem durch ⟨*odd-row color expression*⟩ (ungerade Zeilennummern) und ⟨*even-row color expression*⟩ (gerade Zeilennummern) vorgegebe-

```
\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

Abbildung 7: Alternierende Zeilenfarben in Tabellen: `\rowcolors` und `\rowcolors*`

nen Farbschema zu färbende Tabellenzeile ist. Die Farbargumente können auch leer bleiben (= keine Farbe). Neben `\rowcolors` existiert auch eine Variante `\rowcolors*`, bei der die in ⟨*commands*⟩ angegebenen Befehle in Zeilen mit deaktiviertem Farbschema (siehe unten) ignoriert werden, während in der „normalen“ Version ⟨*commands*⟩ in jeder Tabellenzeile angewandt werden. Solche optionalen Befehle können beispielsweise `\hline` oder `\noalign{`⟨*stuff*⟩`}` sein.

Mit den Befehlen `\showrowcolors` und `\hiderowcolors` kann man temporär das Farbschema aktivieren und deaktivieren. Daneben kann jederzeit mittels `\rowcolor` eine individuell abweichende Färbung einzelner Zeilen erreicht werden. Der Zähler `\rownum` gewährt innerhalb einer Tabelle Zugriff auf die aktuelle Tabellenzeile.

Ein ausführliches Beispiel ist in Abbildung 7 aufgeführt.

Zusammenfassung und Ausblick

In diesem Artikel wurde geschildert, wie das Paket `xcolor` für diverse Arten von Farbschattierungen, Farbmischungen und Farbenfolgen eingesetzt werden kann. Eine eingehende Diskussion der zugrunde liegenden Umrechnungsformeln zwischen den verschiedenen Farbmodellen wird Gegenstand eines späteren Artikels sein.

Literatur

[1] Adobe Systems Incorporated: *PostScript Language Reference Manual*; Addison-Wesley; 3. Aufl.; 1999; `http://www.adobe.com/products/postscript/pdfs/PLRM.pdf`.

[2] David P. Carlisle: *`color` – Standard LaTeX Color*; 1999; CTAN: `macros/latex/required/graphics/color.*`.

[3] David P. Carlisle: *Packages in the 'graphics' bundle*; 1999; CTAN: `macros/latex/required/graphics/grfguide.tex`.

[4] David P. Carlisle: *`colortbl` – Color table columns*; 2001; CTAN: `macros/latex/contrib/carlisle/colortbl.*`.

[5] David P. Carlisle: *`pstcol` – PSTricks color compatibility*; 2001; CTAN: `macros/latex/required/graphics/pstcol.*`.

[6] Denis Girou: *`pst-fill` – A PSTricks package for filling and tiling areas*; 2000; CTAN: `graphics/pstricks/doc/pst-fill.doc`.

[7] Michel Goossens, Sebastian Rahtz und Frank Mittelbach: *The LaTeX Graphics Companion*; Addison-Wesley; 1997.

[8] Uwe Kern: *`xcolor` – LaTeX color extensions*; 2003; CTAN: `macros/latex/contrib/xcolor/`.

[9] Rolf Niepraschk: *`colorinfo` – Info from defined colors*; 2003; CTAN: `macros/latex/contrib/colorinfo/`.