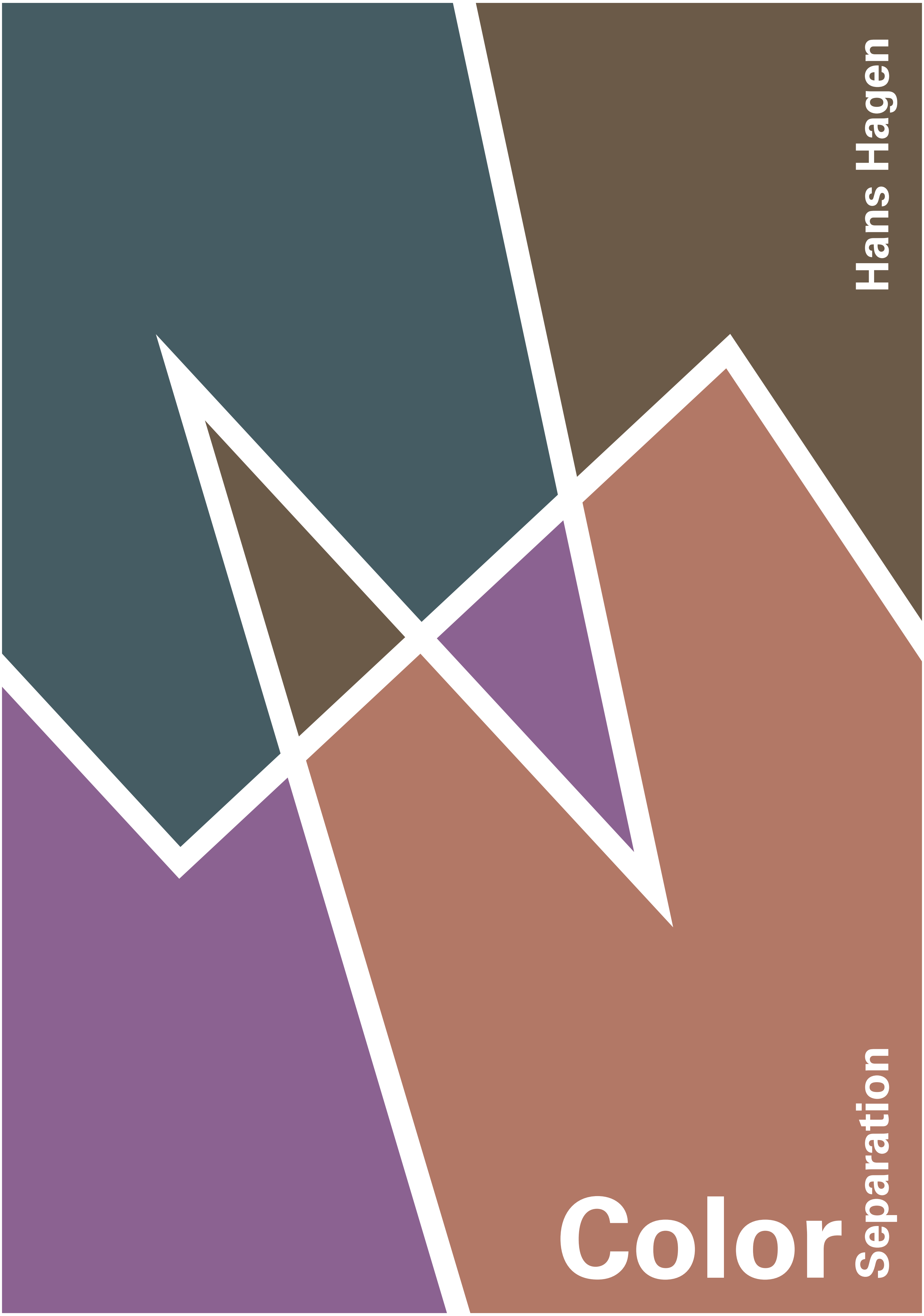


Hans Hagen

Color Separation

1 Color spaces

Currently ConTeXt supports four color spaces: gray, RGB, CMYK, and spot colors. The first three are defined rather straightforward. The subtractive colors in CMYK color space are defined as follows:

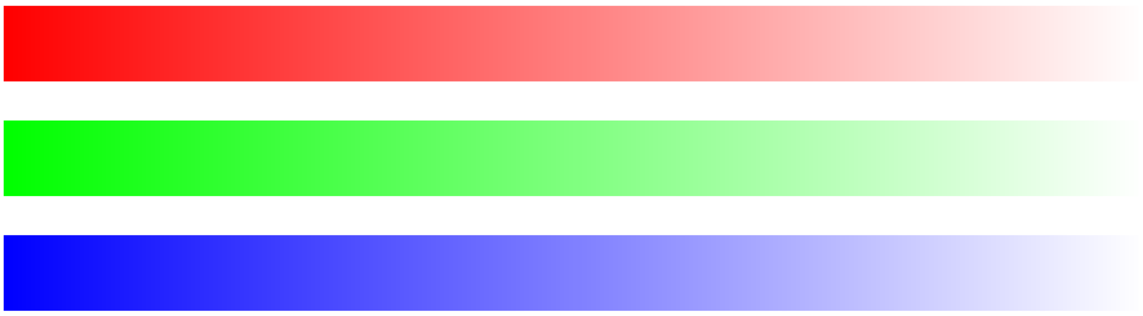
```
\definecolor[my-cyan]    [c=1]
\definecolor[my-magenta] [m=1]
\definecolor[my-yellow]  [y=1]
\definecolor[my-black]   [k=1]
```

These colors (also known as process colors) show up as follows:

In the RGB color space, we use additive colors:

```
\definecolor[my-red]   [r=1]
\definecolor[my-green] [g=1]
\definecolor[my-blue]  [b=1]
```

This time we get:



Gray colors are defined with:

```
\definecolor[my-black] [s=0]
```

Watch how this time we start at zero!

When defining a color in CMYK color space, you can mix c, m, y and k components. On the cover you see four colors defines as:

```
\definecolor [cyan]    [c=1,m=.25,y=.25,k=.25]
\definecolor [magenta] [c=.25,m=1,y=.25,k=.25]
\definecolor [yellow]  [c=.25,m=.5,y=1,k=.25]
\definecolor [black]   [c=.25,m=.5,y=.25,k=1]
```

This leaves us spot colors. These are defined in two steps. First you define the spot color itself:

```
\definecolor[my-spot][c=.7,m=.5,y=.3,k=.1]
```

As any other primary color, this color covers a range:

You can now define colors in this range:

```
\definecolor[bright][my-spot][p=1]
\definecolor[dark]  [my-spot][p=.5]
```

The chained definition is needed because devices, like screen devices, that cannot handle spot colors and need to have a fall back.

2 Special effects

Those familiar with MetaFun will know that it supports shading. Actually the previous shades were produced with code like:

```
\startMPcode
linear_shade (unitsquare xyscaled (TextWidth, 1cm), 5,
  \MPcolor {my-spot}, white) ;
\stopMPcode
```

As you see here, we use `\MPcolor` to tell METAPOST what color to use. Because CONTEXT will handle the color definitions when including the graphic in the document, the proper transformations will take place and the required (PDF) resources will be embedded.

Both CONTEXT and `MetaFun` support transparent colors. As you may expect, transparency works for all color spaces. However, the place where you define them differs:

```
\definecolor[redish]  [r=.5,t=.5,a=1]
\definecolor[cyanish][c=.5,t=.5,a=1]
\definecolor[grayish][s=.5,t=.5,a=1]
```

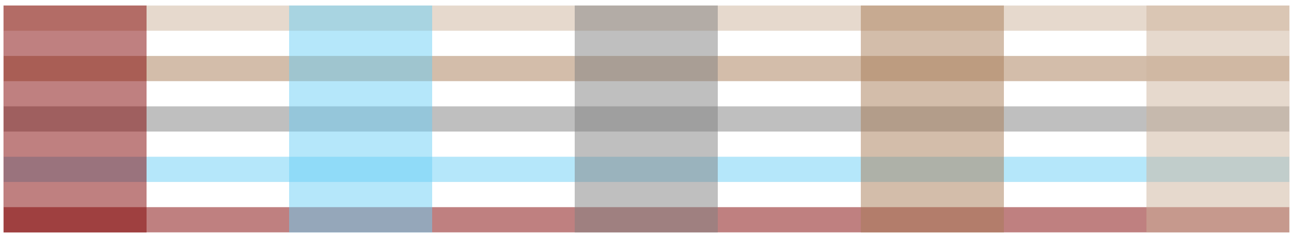
but:

```
\definecolor[spottish][c=.3,m=.5,y=.7,k=.1]
```

with:

```
\definecolor[brightish][spottish][p=1,t=.5,a=1]
\definecolor[darkish]   [spottish][p=.5,t=.5,a=1]
```

The following graphic demonstrates that these colors indeed are transparent.



3 Options

You can mix color spaces in one document. However, what color space you use, depends on the medium on which the document will be read.

For screen documents, RGB is often best, if only because screen devices use these three color channels themselves. All channels zero gives you black, and all channels one produces white.

Printing houses often prefer CMYK and/or spot colors. They will split the document into the colors used for printing, so in case of CMYK only, there will be four variants, one for each process color. If a document has only one or two colors, spot colors will give good results, if only because no screen has to be used. Also, the less channels, the cheaper the print.

One danger luring with CMYK is that some applications produce output depending on the medium that will be used: plain paper, glossy paper, colored paper etc. It's best to let this kind of fine–tuning to the raster image processor that will produce the plates, since that device knows best what it is (in)capable off and what compensations are needed.

You can control CONTEXT's color mechanism in several ways using the `\setupcolors` command. First of all, you need to enable colors:

```
\setupcolors[state=start]
```

By default all four methods are supported, but you can change this behaviour:

```
\setupcolors[rgb=no,spot=no]
```

This will force CONTEXT to use CMYK and gray scales only (gray scale takes less code and will remap on the black process color).

As long as you use `\MPcolor` in your METAPOST graphics, you can be sure that those graphics will obey the color settings. In the background, CONTEXT will convert colors defined in the disabled color spaces into those enabled.

Although it can be tempting to use names for colors like *red* or *blue*, it makes sense to use more meaningful names, like *important* or *dangerous*.

```
\definecolor[dangerous][r=1]
```

Even better is to define colors in an indirect way:

```
\definecolor[red][r=1]
\definecolor[dangerous][red]
```

This means that when we map more colors on red, we only have one definition of red to maintain. At the time of definition of *dangerous*, only the name is mapped, which means that we can change the definition of red afterwards. If you want to freeze colors at definition time, say:

```
\setupcolors[expansion=yes]
```

You can specify colors in your preferred color space, but your preferences may not match those of your printer or printing house. For that reason, CONTEXT provides color conversion. You can for instance force CMYK output by saying:

```
\setupcolors[rgb=no,spot=no,cmyk=yes]
```

If you also turn off CMYK support, you will get weighted gray scales. You can also demand reduced CMYK color, where the black component is avoided, by setting `reduction` to `yes`. Table 1 shows what happens when such conversions are asked for. We use the previously defined colors.

color	mixed	rgb	cmy (reduced)	cmyk	gray
red	1 0 0	1 0 0	0 1 1 0	0 1 1 0	0.300
green	0 1 0	0 1 0	1 0 1 0	1 0 1 0	0.590
blue	0 0 1	0 0 1	1 1 0 0	1 1 0 0	0.110
cyan	1 0 0 0	0 1 1	1 0 0 0	1 0 0 0	0.700
magenta	0 1 0 0	1 0 1	0 1 0 0	0 1 0 0	0.410
yellow	0 0 1 0	1 1 0	0 0 1 0	0 0 1 0	0.890
spot	.7 .5 .3 .1	0.200 0.400 0.600	0.800 0.600 0.400 0	.7 .5 .3 .1	0.362
bright	my-spot 1	0.200 0.400 0.600	0.800 0.600 0.400 0	0.700 0.500 0.300 0.100	0.362
dark	my-spot .5	0.100 0.200 0.300	0.400 0.300 0.200 0	0.350 0.250 0.150 0.050	0.181

Table 1 This table show what values end up in the output files when conversion takes place.

Due to the nature of TEX and the control needed for output devices, you need to be aware of interference of color switches and typesetting. The low level color control commands (for instance literal PDF code) ends up in the page stream. We will demonstrate this with an example (here we use the same cyan variant as on the cover):

We start with no color, `1 .25 .25 .25 k 1 .25 .25 .25 K`but now we have switched to cyan, `/my-spot CS 1 SC /my-spot cs 1 sc`and now we're typesetting in bright, `1 .25 .25 .25 k 1 .25 .25 .25 K` and some cyan, and `.5 0 0 rg .5 0 0 RG/Tr2 gs`after a bit of redish`/Tr0 gs1 .25 .25 .25 k 1 .25 .25 .25 K`, we're back at cyan again.`0 g 0 G`

The output (DVI converted to POSTSCRIPT, or PDF) is made up of so called independent pages. This means that, apart from resources like fonts, printing a page should not demand processing of previous pages. So, when we use special features, like color, we have no knowledge about the current color at a page break. Therefore, we need to synchronize colors after a page break.

TEX invokes the output routine (page builder) after a paragraph is processed. If the paragraph is typeset in a color, the color is already reset to its orginal. However, when TEX decides that it should break in the middle of that particular paragraph, we have to make sure that we add codes like shown in the previous example at the top of the next page. CONTEXT keeps track of colors and tries to synchronize the current state and the output state as good as possible.

Starting and ending a color obeys TEX's grouping mechanism. This means that when you want to typeset a document as a whole in a color, you need to group the whole document. This can interfere with the final page break, and as a result, low level data streams like the ones shown before may end up on a page of their own, which results in an empty page. In order to prevent such problems, you can set the document color with:

```
\setupcolors[textcolor=somecolor]
```

Similar complications can occur when you use colors in the `\framed` command, which is also used extensively in core macros. The `\color` command expects an argument, and therefore exactly knows where to end coloring. Commands like `\red` on the other hand, hook themselves into the grouping mechanism. As a result, their behaviour may interfere with other mechanism, like automatic strut placement. Take the following text (here we use the same magenta variant as on the cover):

```
The small bars shown here are called struts. Normally they
are invisible, but they play an important role in ensuring
consistent vertical spacing. You can {\magenta visualize}
them with \type {\showstruts}.
```

If we frame this text with:

```
\framed[align=middle]{\cyan...}
```

we get:

The small bars shown here are called struts. Normally they are in-
visible, but they play an important role in ensuring consistent
vertical spacing. You can visualize them with `\showstruts`.

At first sight the result looks ok, but let's visualize the struts:

The small bars shown here are called struts. Normally they are in-
visible, but they play an important role in ensuring consistent
vertical spacing. You can visualize them with `\showstruts`.

As you can see, there is some white space before the strut. Normally such a strut tries to get rid of it, so how come that it fails here? Let's also visualize the low level streams:

1 .25 .25 .25 k 1 .25 .25 .25 K The small bars shown here are called struts. Normally they are
invisible, but they play an important role in ensuring consistent vertical spacing.
You can .25 1 .25 .25 k .25 1 .25 .25 K visualize 1 .25 .25 .25 k 1 .25 .25 .25 K them with `\showstruts`. 0 g 0 G

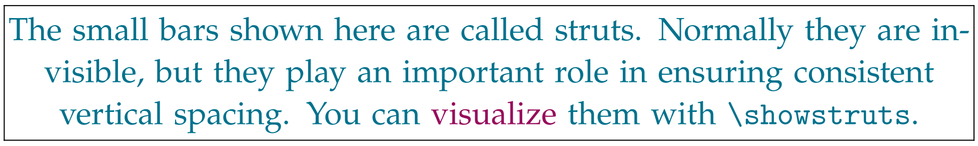
It now becomes clear that the color commands end up before the strut, and as a result there is no way to go back. This is why in such situations it's better to apply the colors in a different way:

```
\framed[align=middle,foregroundcolor=cyan]{...}
```

This results in:

1 .25 .25 .25 k 1 .25 .25 .25 K The small bars shown here are called struts. Normally they are invisible,
but they play an important role in ensuring consistent vertical spacing. You
can .25 1 .25 .25 k .25 1 .25 .25 K visualize 1 .25 .25 .25 k 1 .25 .25 .25 K them with `\showstruts`. 0 g 0 G

Now, when we turn off the visualization, we get the properly aligned text:



The small bars shown here are called struts. Normally they are invisible, but they play an important role in ensuring consistent vertical spacing. You can visualize them with `\showstruts`.

The main message here is that, when you apply color, you need to be aware of the fact that this introduces (normally) invisible content. Therefore, in case of doubt, use the `\color` or `\startcolor` command.

4 Groups and palets

The reference manual describes in detail how you can group colors and how you can define color palets. Since we're dealing with color based workflows here, we only want to stress that using color palets may ease your task when one style has to serve multiple documents with different color settings. You define a palet as follows:

```
\definepalet[first] [maincolor=cyan,subcolor=yellow,extracolor=magenta]
\definepalet[second][maincolor=magenta,subcolor=cyan,extracolor=yellow]
```

You can inherit them as well:

```
\definepalet [default] [first]
```

This means that you can isolate color definitions pretty well. Such a palet is enabled by:

```
\setuppalet[first]
```

Because in the end, we still deal with colors, all the color manipulations mentioned in this document apply to palets as well. You can inspect a palet by `\showpalet`, as demonstrated in figure 1. The gray bars are the weighted gray alternatives.

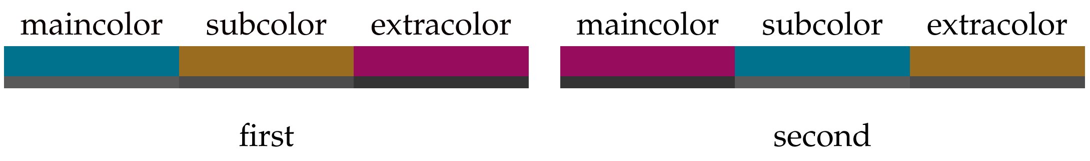


Figure 1 Two color palets visualized.

5 Separation

We now come to the main topic of this manual: color separation. An RGB color has three color components, while an CMYK color has four. Although modern printing devices should be able to deal with both, in print the CMYK color space is preferred.

When printing in full color, the four colors are printed separately, which comes down to printing four instances of the same document in succession. The separation needed for this can best be dealt with in the engine itself, but there may be occasions where you need to provide the separations yourself.

You get the cyan channel version of a document when you say:

```
\setupcolors[split=c]
```

In a similar way, you can ask for a magenta (`m`), yellow `y`, red (`r`), green (`g`), blue (`b`), or black (`s`) version. If you want a spot color, you set `split` to the desired color:

```
\setupcolors[split=yourspot]
```

A more convenient way to do this, is to use TEXEXEC:

```
texexec --separation=c
```

There is a preliminary script `texsplit` that automatically splits the document, using information stored in the auxiliary file `jobname.tuo`.

Graphics are a complication in this process. For black–and–white graphics, you can specify in which channel they have to end up:

```
\externalfigure[graphic][split=c]
```

Color graphics need to be separated independent of the document and we assume that this has been done. We discuss three ways to deal with including them.

A convenient way to assign the (separated) graphics to a channel is the following:

```
\externalfigure[graphic\colorsplitsuffix]
```

In this case, the original graphic is called `graphic`, while the cyan one is called `graphic-c`, and the rest accordingly.

An alternative is to collect the graphics in a figure database and just switch databases. This saves you the effort of specifying the suffix each time.

```
\usefigurebase[graphics\colorsplitsuffix]
```

Instead of a suffix, you can apply a prefix (which is sometimes handier when you want to list files):

```
\usefigurebase[\colorsplitprefix graphics]
```

Alternatively, you can use a different path for the split graphics:

```
\setupexternalfigures[directory={./figures\colorsplitsuffix}]
```

A third alternative is to use modes. You can of course introduce special modes, like:

```
\startmode[cyan]
  \setupcolors[split=c]
  \usefigurebase[cyan]
\stopmode
```

Here you need to enable a mode or run TEXEXEC with `--mode=cyan`. However, CONTEXT provides channel specific system modes. A system mode is prefixed by a `*`.

```
\startmode[*color-c]
  all kind of channel specific settings
\stopmode
```

All this may sound complicated, but color separation is a complex matter anyway, so you'd better know what you're doing. This is especially true when you set `criterium`, which by default is set to `all`. This means that when a 100% spot color is used, it is assumed to completely knock out its background. On the other hand, for instance a 50% spot color is painted on a white background.

```
\setupcolors[criterium=none]
```

When `criterium` is set this way, the background is always painted in white. This assumes a rather precise printing workflow.

For external figures, you can specify a background color with `splitcolor`. By default this color is set to white.

```
\setupexternalfigures[splitcolor=white]
```

6 Bonus example

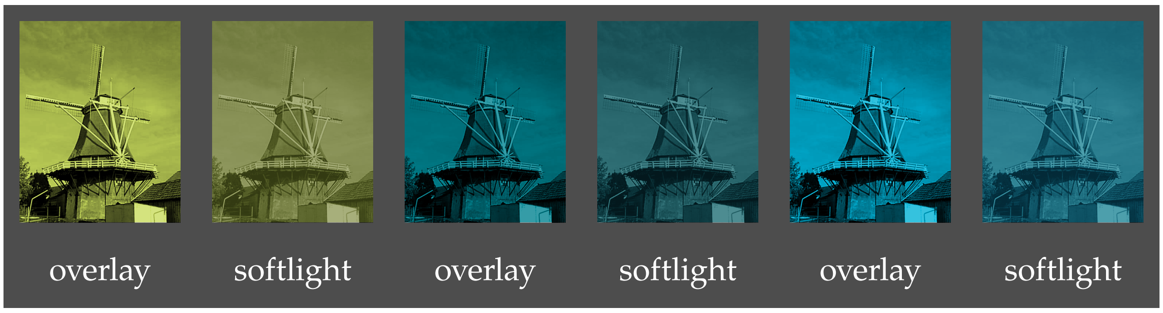
When color separation was added CONTEXT, some experiments were conducted with coloring bitmap graphics. One of the poor–mans colorizations is presented here.

First we define a few transparent colors. Out of the alternatives, we choose `overlay` and `softlight`. (More about transparency can be found in the MetaFun manual.)

```
\definecolor [p-one]   [c=1,y=.3,k=.3]
\definecolor [p-two]   [c=.25,y=.8]
\definecolor [p-three] [c=.9,y=.15]

\definecolor [one-1]   [p-one]   [p=1,t=1,a=overlay]
\definecolor [one-2]   [p-one]   [p=1,t=1,a=softlight]
```

```
\definecolor [two-1]   [p-two]   [p=1,t=1,a=overlay]
\definecolor [two-2]   [p-two]   [p=1,t=1,a=softlight]

\definecolor [three-1] [p-three] [p=1,t=1,a=overlay]
\definecolor [three-2] [p-three] [p=1,t=1,a=softlight]

\definecolor [p-one]   [c=1,y=.3,k=.3]
\definecolor [p-two]   [c=.25,y=.8]
\definecolor [p-three] [c=.9,y=.15]

\definecolor [one-1]   [p-one]   [p=1,t=1,a=overlay]
\definecolor [one-2]   [p-one]   [p=1,t=1,a=softlight]

\definecolor [two-1]   [p-two]   [p=1,t=1,a=overlay]
\definecolor [two-2]   [p-two]   [p=1,t=1,a=softlight]

\definecolor [three-1] [p-three] [p=1,t=1,a=overlay]
\definecolor [three-2] [p-three] [p=1,t=1,a=softlight]
```

This cheap trick leads to:



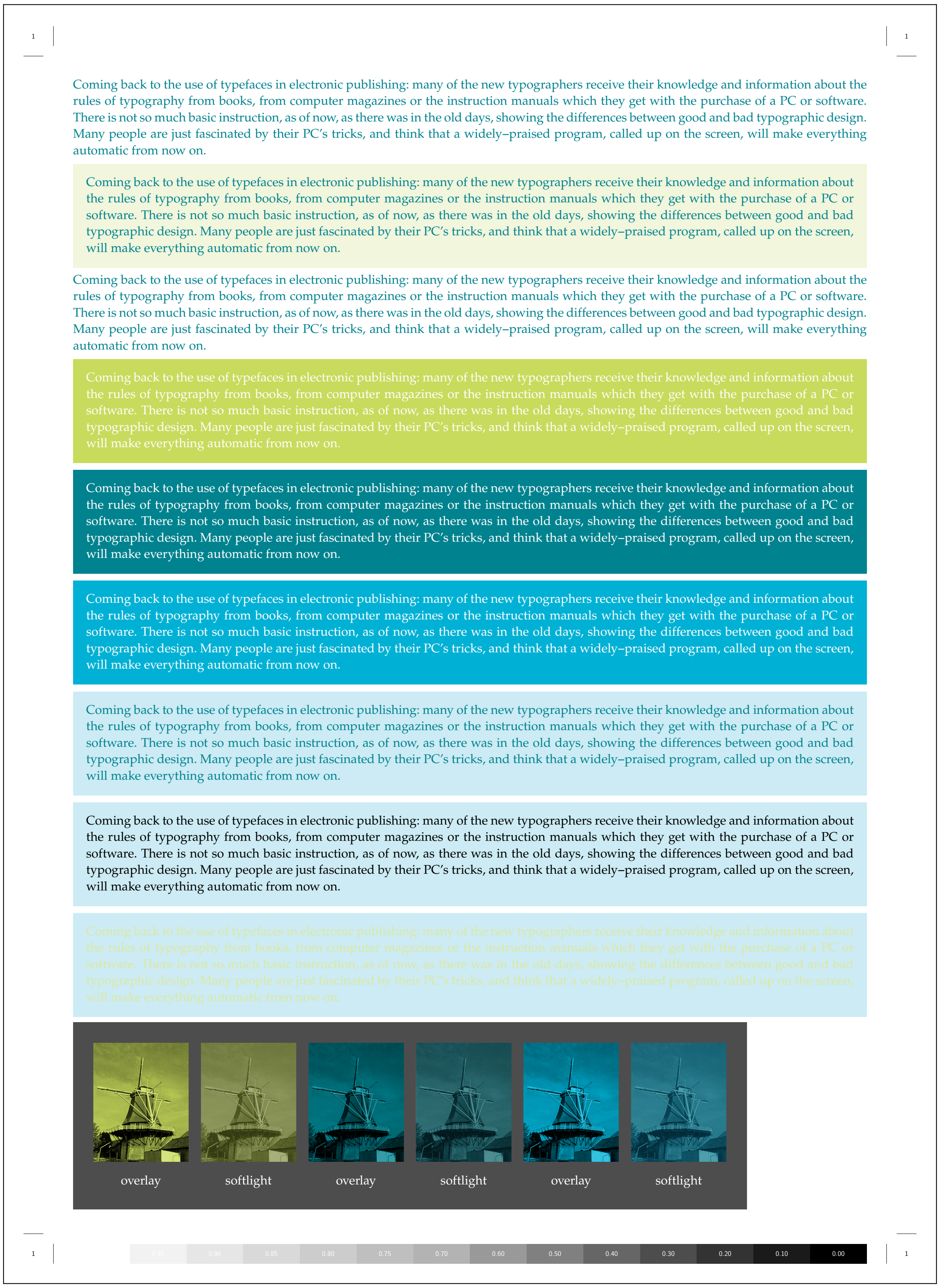
In this example, the gray background was accomplished with:

```
\definecolor[darkgray][s=.3]

\defineframedcontent
  [mill]
  [background=color,
   backgroundcolor=darkgray,
   frame=off,
   offset=1ex]
```

and:

```
\startframedcontent[mill] ... \stopframedcontent
```

This example was part of a test document with three spot colors and black, which demands four splits. The original and its four splits are shown on the next pages.



Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Figure 2 A test document with three spot colors & black.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

0.95 0.90 0.85 0.80 0.75 0.70 0.60 0.50 0.40 0.30 0.20 0.10 0.00

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

overlay softlight overlay softlight overlay softlight