Experiments in TEXnicolour — A SLITEX Sub-style for Colour Printers*

David Love

Abstract

SLITEX assumes that colour transparencies will be produced from layers, each printed in a single colour. This sub-style allows a suitable `dvi` driver to produce multi-coloured slides in one go (i.e., using a single layer). It is customisable for PostScript printers or simpler colour devices like the the HP PaintJet.

1 Introduction

SLITEX—a version of LATEX for making slides—supports colours in its output by producing 'colour layers', one per colour [1]. It is assumed that these will be copied to transparencies of the appropriate colour and overlapped to make a composite multi-colour slide. Colour is valuable on slides, but this means of producing it is inconvenient and inappropriate if you have a colour printer and a suitable `dvi` driver for it.

This sub-style for SLITEX allows all the colours to be produced in one layer. It is designed for use with colour PostScript printers or others like the HP PaintJet with suitable `dvi` drivers, following experiments with these printers at Daresbury. It is intended to provoke discussion of the use of colour with TEX, a subject about which little has been said so far in public, rather than be at all definitive.

2 Approach

Use of colour with TEX is non-standard and has to be done by passing information to the `dvi` driver. This could be done in one of two ways:

- using fonts whose names idenitify them to the driver to be printed in a particular colour but are otherwise identical to the black version (possibly using virtual fonts [2]);
- using the `\special` mechanism to tell the driver to change colours.

The second option is easier to implement with an existing driver and allows colour for objects other than characters i.e., rules,[1] and has been adopted here. Unfortunately, existing drivers differ in their `\special` mechanisms and few understand colour explicitly. However, drivers which allow you to include arbitrary PostScript with a `\special` have this ability implicitly since colour operators are defined in PostScript [3]. (Colour gets rendered in shades of grey on a normal laser printer, which can help with proofing.)

There is a proposed standard for `\specials` [4] which includes colour information, but the definition of the `textcolor` `\special` is inconveniently related to the `dvi` stack. Working with existing PostScript drivers, one has to make some assumptions about them, as detailed below.

In the case of the PaintJet printer there does not appear to be an existing driver, so one was produced with a suitable `\special` defined arbitrarily.

3 PaintJet Driver

The Hewlett-Packard PaintJet can print in colour at 180 dpi, which is quite adequate for making transparencies. A driver for it was produced using the HP Laserjet version from Beebe's family [5] as a basis since this has rather similar control sequences. It was extended to maintain four bitmaps independently, allowing three colours other than black to be printed. Allowing more colours would make the code more complicated

*File version 1.2, dated 18/4/90.

[1] `\colorslides` doesn't work properly with rules—they come out in each layer.

and slower as well as using more memory for the bitmaps. It supports one \special to allow colour changes between black, red, green and blue. A defect is that the result of overlapping items of different colours is independent of their order in the dvi file, but this simplifies the driver considerably.

4 Assumptions

The necessary assumptions about the driver are largely isolated in the macros \special@color and \colormix.[2] Suitable changes could quite easily be made to conform with an eventual standard for \specials or use a different driver.

4.1 PostScript

We assume that a PostScript driver understands \specials of the form

\special{ps::␣⟨*code*⟩},

where ⟨*code*⟩ is arbitrary PostScript from which the surrounding code output by the driver is not protected. This is appropriate for Rokicki's dvips and ArborText's DVILASER/PS. We use the RGB colour model [3, §4.8].

4.2 PaintJet or other printer

A non-PostScript printer should understand a special of the form {color␣⟨*colour*⟩}, where ⟨*colour*⟩ can take the values black, red, green or blue, determining the colour of all subsequent printing until the next change. The default is assumed to be black.

5 User interface

As well as redefining some internal SLiTEX macros, we make some new ones as described below.

\psprinterfalse The default is to assume the use of a PostScript printer. Saying \psprinterfalse changes this to produce output suitable for the PaintJet (or similar printer). This should be done only in the preamble.

\colors
slide Any colours you use *must* be declared using \colors (even red, green, blue and black). They do not need including in the argument of \begin{slide} for use with \truecolors, but there needs to be some argument, even if it's null.

\truecolors Like \blackandwhite, \truecolors is invoked from the driver file to process a batch of slides described in the file given as its argument. It works like \blackandwhite in that it makes only one layer, but will include the information for the printer to produce the colours in the dvi file. Notes are not printed by \truecolors.

5.1 PostScript specifics

\colormix Before you use a new PostScript colour, you have to define how it is mixed from red, green and blue. \colormix does this for you. It takes four arguments. The first is the name of the colour and the others are numeric values for the intensities of red, green and blue respectively in the mixture. (See the description of RGB in [3].) These values must lie in the range 0–1 such that 1 is most intense and 0 is no intensity. Thus you might define yellow by

\colormix{yellow}{0}{1}{1}

You don't need to define red, green, blue or black—they are done already.

You could put blocks of text on a coloured background either by setting them on top of a suitably-sized rule or by using explicit \specials.

[2] I spell the English way except in code names, where I reluctantly use 'color' for consistency with existing code.

5.2 Other printers

Only red, green, blue and black are available. Use of `\colormix` will produce a warning. Using a non-white background won't work very well with the PaintJet.

5.3 Problem with list environments

A problem occurs if you want the first text in a list environment item to be a different colour to the label since `\specials` before the text appear in the `dvi` file before the label. A way round this is to insert `\leavevmode` after `\item`. This sort of problem may occur in other situtaions.

6 Code

6.1 Driver-independent code

```
\typeout{SliTeX style option 'truecols' (v. \fileversion\space of \filedate)}
```

`\if@truecol` We introduce a new switch to tell us whether we can assume we have true colour printing capability and unset it initially.

```
\newif \if@truecol \@truecolfalse
```

`\the@color` We keep track of the current colour using `\the@color` (to get colours right with grouping). Initially it's black.

```
\def\the@color{\black}
```

`\ifpsprinter` This tells us if we're dealing with a PostScript printer (the default) or not.

```
\newif \ifpsprinter \psprintertrue
```

`\truecolors` A new `\truecolors` command is defined, which works like `\blackandwhite`, printing everything in the file given as its argument in one layer. Thus it must set the `\@bw` switch. This slightly unfortunate change to `\@bw`'s meaning makes the alterations easier.

```
\newcommand{\truecolors}[1]{\@truecoltrue \blackandwhite{#1}}
```

`\@color` We modify `\@color` to take note of the `@truecol` setting (if `@bw` is set). If appropriate, it puts out a colour-changing `\special` using `\special@color`. We can omit the test for a colour change in math mode if we're not depending on invisible fonts.

```
\def\@color#1{%
  \if@truecol \else \@mmodetest \fi %  change
  \if@bw \@visibletrue
    \if@truecol \special@color{#1}\fi %  change
  \else \@visiblefalse
    \@for \@x@a :=#1\do{\ifx\@x@a\@currcolor\@visibletrue\fi}\fi
  \@currsize \@currfont \ignorespaces}
```

`\endslide` We ensure that the driver colour is reset to black at the end of each slide by setting it in `\endslide`.

```
\def\endslide{\@color{black}\par\break}
```

`\note` We don't want to waste truecolour output on producing notes, which we will if we don't re-define `\note` (since `@bw` is set by `\truecolors`). We just add a test on `@truecol`.

```
\def\note{%
  \stepcounter{note}%
  \if@bw
    \if@truecol \gdef\@slidesw{F}%  added
    \else
      \gdef\@slidesw{T}%
      \if@onlynotesw\@whilenum \c@slide > \@donotehigh\relax
        \do{\@setlimits\@donotelist\@donotelow\@donotehigh}\ifnum
```

```
      \c@slide < \@donotelow\relax \gdef\@slidesw{F}\fi\fi\fi
  \else \gdef\@slidesw{F}\fi
  \if\@slidesw T\newpage\thispagestyle{note}\else
  \end{note}\@gobbletoend{note}\fi }
```

`\colors` We re-define `\colors` to check that we know about those specified. It could, perhaps, be dispensed with at the cost of incompatibility with standard SLiTEX. If we're not using PostScript only the four pre-defined colours are available.

```
\def\colors#1{%
  \@for\@colortemp:=#1\do{%
    \@ifundefined{\@colortemp @mix}{%  change
      \ifpsprinter
        \errhelp{You could use I to define it now.}%
        \errmessage{You need to use \noexpand\colormix to define
                    \@colortemp}%
      \else
        \errhelp{It's safe to carry on.}%
        \errmessage{You can only use black, red, green and blue with
                    this printer, not \@colortemp}%
        \expandafter\xdef\csname\@colortemp\endcsname{\relax}%
      \fi
    }{}%
    \expandafter\xdef\csname\@colortemp\endcsname
      {\noexpand\@color{\@colortemp}}%
  }% (do)
  \ifx\@colorlist\@empty \gdef\@colorlist{#1}%
  \else \xdef\@colorlist{\@colorlist,#1}\fi}
```

`\pagestyle` Finally we change the default pagestyle since the alignment marks would only be useful with overlays.

```
\pagestyle{plain}
```

6.2 Driver-dependent code

`\colormix` The PostScript user can mix new colours in the RGB model using `\colormix`. It takes four arguments, the name of the colour (which must then be declared with `\colors`) and three real numbers in the range 0–1 describing respectively the intensity of red, green and blue in the result. The result is to define a macro of the form `\`⟨*colour*⟩`@mix` to be a list of arguments 2–4.

```
\def\colormix#1#2#3#4{%
  \ifpsprinter \else \typeout{Warning: \noexpand\colormix used with
    \noexpand\psprinterfalse.}\fi
  \color@range@check{#2}\color@range@check{#3}\color@range@check{#4}%
  \expandafter\xdef\csname#1@mix\endcsname{#2 #3 #4}}
```

`\color@range@check` We need to check that the numeric arguments of `\colormix` are in the range 0–1. Since they're real numbers we convert them to dimensions to test.

```
\def\color@range@check#1{%
  \def\fred{\errhelp{Carry on, but you'll get a PostScript error if
              you try to print the results.}%
            \errmessage{\noexpand\colormix arguments must be in the
              range 0--1}}%
  \ifdim#1 pt < 0pt \fred \fi \ifdim#1 pt > 1pt \fred \fi }
```

`\special@color` This has to change the current colour on the output device to that given as its argument, but first has to store the current value and make sure it is reinstated at the end of the current group. (Coloured fonts would avoid the need for this.) Note the

{} inserted at the end of the group to avoid spaces being gobbled afterwards as with {\red foo} bar. Probably there's a better way of doing this.

```
\def\special@color#1{%
  \def\the@color{\csname#1\endcsname}\aftergroup\the@color
    \aftergroup{\aftergroup}%
```

To change colour with PostScript we use a \special which will emit code of the form

⟨*red level*⟩ ␣⟨*green level*⟩␣⟨*blue level*⟩␣setrgbcolor

Perhaps this should be parameterised. The three levels are provided by a macro of the form \⟨*colour*⟩@mix which must be defined for each ⟨*colour*⟩ which is used. It might be useful to define a white font (RGB=[1 1 1]) which could be painted over a coloured region.

```
\ifpsprinter \special{ps:: \csname#1@mix\endcsname\space
                           setrgbcolor\space}%
```

For the PaintJet we output

\special{color␣⟨*colour*⟩},

where ⟨*colour*⟩ can be black, red, green or blue and we assume that #1 is one of these.

```
    \else \special{color #1}\fi % (\ifpsprinter)
  } % (\def\special@color)
```

\black@mix
\red@mix
\blue@mix
\green@mix

Here we define the RGB mixes for the PostScript colours we support initially. These macros need defining for the PaintJet too, so that we can check on colours the user tries to invoke.

```
\def\black@mix{0 0 0} \def\red@mix{1 0 0}
\def\blue@mix{0 0 1}  \def\green@mix{0 1 0}
```

References

[1] Leslie Lamport. *LaTeX: A Document Preparation System*; slitex.tex dated 10 November 1986; slides.sty dated 17 January 1986.

[2] Donald Knuth. "Virtual fonts: More fun for grand wizards," *TUGboat*, 11(1), 1990, p. 13.

[3] Adobe Systems Inc. *PostScript Language Reference Manual.* Addison-Wesley, 1985, ISBN 0-201-10174-2.

[4] Don Hosek. Proposed DVI \special command standard. (See *TUGboat*, 10(2), 1989, p. 188.)

[5] Nelson H. F. Beebe. "Public Domain TeX DVI Driver Family," *TUGboat* 8(1), p. 41.

◇ David Love
SERC Daresbury Laboratory,
Warrington WA4 4AD, UK
d.love@daresbury.ac.uk