

Travels in TeX Land: Benefits of thinking a little bit like a programmer

David Walden

Abstract In this column in each issue I muse on my wanderings around the TeX world. Section 1 of this column provides another illustration of the benefit of defining a few simple macros for a particular TeX project. Section 2 gives another example of using an external processor in combination with TeX. Section 3 gives another example.

1 A subtle (to me) issue with L^AT_EX macros

I will be republishing an oral history of my mother-in-law that was originally typed on a typewriter, printed at Kinko's, and bound by a thesis binding company. I want it to continue to have the "look and feel" of a typed book, while still using some of TeX's capabilities. For instance, I am using a proportional typewriter font rather than a non-proportional font such as the original book had.

The book has section titles for its many short sections and no chapter titles. Thus, I want each section title in the Table of Contents, and I want running heads on the pages for the section titles. The only running heads will be these section titles at the left side of even numbered pages and at the right side of odd numbered pages. However, when the section title is the first line on a page, I don't want the section title to also be in the running head; in this case I want the running head to be blank.

I first used the following commands to try to get what I wanted.

```
\RequirePackage{fancyhdr}
\renewcommand{\headrule}{} %don't want a line below the running head
\pagestyle{fancy} \fancyhead{} %copied from the LaTeX Companion 2
\newcommand{\mysectionname}{} %the section name for the running head
% is initially blank
```

```

\fancyhead[RO,LE]{\mysectionname} %use the current section name in
                                % in the right odd and left even positions

%the \mysection{title} command is used to define my section commands
\newcommand{\mysection}[1]{
\noindent\uline{#1}             %insert the section heading in-line
\addcontentsline{toc}{section}{#1} %add it to the TOC
\renewcommand{\mysectionname}{#1} %redefine what is used for the running head
}

```

The above worked except for the obvious problem that it would also place the same text in the running head immediately above a section title on the first line of text of a page.

Rather than trying make `\mysection` a lot more complicated so it automatically handled the above mentioned condition, I defined a new command (`mysectionN`) that I would manually insert when I found a situation where `\mysection` didn't do what I wanted it to do:

```

\newcommand{\NewSectName}{} %need a place to save the section name

\RequirePackage{afterpage}
\newcommand{\mysectionN}[1]{ %delay running head until the Next page
\noindent\uline{#1}
\addcontentsline{toc}{section}{#1}
\renewcommand{\mysectionname}{} %don't want a section head on this page
\renewcommand{\NewSectName}{#1} %save section name for later
%the following makes it take effect after the page change
\afterpage{\renewcommand{\mysectionname}{\NewSectName}}
}

```

Unfortunately, that didn't work. I debugged a while and eventually it dawned on me that maybe the renewed definition the `NewSectName` parameter wasn't available by the time the next page came.¹ So I tried using `\gdef` to force a change to a global version of `NewSectName`, as follows:

1. Actually, I tried looking at this casually for a minute or two several times over several months before deciding finally one day that I had to get serious and really think this through once and for all.

```

\newcommand{\mysectionN}[1]{ %delay running head until Next page
\noindent\uline{#1}
\addcontentsline{toc}{section}{#1}
\renewcommand{\mysectionname}{}
\gdef\NewSectName{#1} %change the *global* version of NewSectName
\afterpage{\gdef\mysectionname{\NewSectName}}
}

```

This sort of worked, except the correct running head did not always appear on the page immediately following the page with the new section title. If there was more than one following page before a new section title was specified, things worked OK. However, in the case where there was one page after the page with the section title that should have had a running head and then it was appropriate to put in another `mysectionN` command, there was no running head on the in-between page and the running head on the page with the second section title was not delayed until the following page. The latter suggested to me that I needed to do a `\clearpage` to get the page without a running head output before a new delayed running head was output. So I added a `\clearpage` as follows

```

\newcommand{\mysectionN}[1]{ %delay running head until Next page
\clearpage % new command added
\noindent\uline{#1}
\addcontentsline{toc}{section}{#1}
\renewcommand{\mysectionname}{}
\gdef\NewSectName{#1}
\afterpage{\gdef\mysectionname{\NewSectName}}
}

```

and the right thing also happened then with the missing running head on the page after the first section title.

With that working, I worried about it being poor form to use `\gdef` in \LaTeX and searched around for an equivalent in \LaTeX . When I couldn't find an equivalent command in my books, I asked the `comp.text.tex` list (on September 9, 2007) whether an equivalent command exists in \LaTeX . Will Robertson responded *immediately* (12 minutes later — love that `comp.text.tex` list) saying that there was no equivalent and, while he noted that I could just use `\gdef`, he also suggested another option. (There were also a couple of additional responses on `comp.text.tex`

discussing various related complication.) I decided to just stick with `gdef` since there was not much chance of confusing myself by using `\gdef` in this limited situation.

2 Using $\text{T}_{\text{E}}\text{X}$ to help investigate colors

I became interested in specifying color within $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$. I looked up the `color` package in *The $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ Graphics Companion*, second edition, which was just published. (The book is so new it has a 2008 publication date on the back of the title page. It has about 925 pages on graphics capabilities that can be used with $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$, something approaching 400 more pages than the first edition that was published a decade ago. The extensive indexes for the book are on-line at www.latex-project.org/guides/lgc2-ap3.pdf.)

From the book, it appeared that the following commands were relevant to me thinking about color:

```
\usepackage{xcolor}
\setlength{\fboxsep}{10pt}
\colorbox[rgb]{1,0,0}{}
```

For instance, the above creates a red colored square 10 points on a side, e.g.,

```
\colorbox[rgb]{1,0,0}{}
```

I quickly decided that I'd like to see a spectrum of little boxes showing various color combinations—perhaps all the combinations of single decimal digits of specifications of red, green, and blue in their 0–1 intervals. However, that would be about 1,000 little colored squares, and I certainly did not want to type in all those lines of $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$. I also did not want to take the time, at the moment, to find and dig into the $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}/\text{T}_{\text{E}}\text{X}$ graphics packages that might be able to show me color spectrums.

So, I wrote a little Perl program: `gen-color.pl` on the HTML page for this paper. This program generated a $\text{T}_{\text{E}}\text{X}$ file: `colors.tex` on the HTML page. The this file was input into a little $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ file: `color.tex` on the HTML page. (In section 2 of my previous column (starting on page 9 at <http://www.tug.org/pracjourn/2007-3/walden/>), I previously touted the benefit of being able to generate a $\text{T}_{\text{E}}\text{X}$

(or \LaTeX) file with another program before typesetting it with \TeX . This is something one cannot do as easily with text processing or typesetting systems with embedded markup (for example, Word.)

The \LaTeX that, in effect, was compiled was as follows:

```
\documentclass{article}

\usepackage{geometry}
\geometry{paper=letterpaper, lmargin=.5in, rmargin=.5in}

\usepackage{xcolor}

\title{RGB Color Examples}

\author{} \date{}

\begin{document}

\maketitle

\setlength{\fboxsep}{10pt}
\def\Model{rgb}

\def\Red{0}
\def\Green{0}
\def\Blue{0}
\colorbox[\Model]{\Red,\Green,\Blue}{} \quad%
  \hbox to .75in{\Red/\Green/\Blue} \quad
\def\Red{0}
\def\Green{0}
\def\Blue{0.1}
\colorbox[\Model]{\Red,\Green,\Blue}{} \quad%
  \hbox to .75in{\Red/\Green/\Blue} \quad
\def\Red{0}
\def\Green{0}
\def\Blue{0.2}
```

```

\colorbox[\Model]{\Red,\Green,\Blue}{}\quad%
  \hbox to .75in{\Red/\Green/\Blue}\quad
\def\Red{0}
\def\Green{0}
\def\Blue{0.3}
\colorbox[\Model]{\Red,\Green,\Blue}{}\quad%
  \hbox to .75in{\Red/\Green/\Blue}\quad
\def\Red{0}
\def\Green{0}
\def\Blue{0.4}
\colorbox[\Model]{\Red,\Green,\Blue}{}\quad%
  \hbox to .75in{\Red/\Green/\Blue}\quad

%another 95 instances of the above code with varying values
% of red, green and blue go here

```

That in turn resulted in the PDF file `color.pdf` on the HTML page of this paper.

Notice that I defined `\def\Model{rgb}` in my little \LaTeX program, which might seem unnecessary. However, by using `\Model` (for color model) in the `colorbox` commands generated by the Perl program, I can more easily change to using another color model in generating my example, for example `\def\Model{cmy}` for use of the cyan, magenta, and yellow color model.

Obviously, the left-to-right, top-to-bottom order of the color boxes in the PDF file in the above example is quite fragmented in terms of a continuous color spectrum. Therefore, to better understand how the RGB color model works, I dipped into chapter 5 of *Grokking the GIMP* (by Carey Bunks, New Riders Publisher, Indianapolis, IN, 2000); the book is also available on the Internet at <http://gimp-savvy.com/BOOK/>.²

It obvious that values of red, green, and blue, each running from 0 to 1 (or from 0 to 256 or whatever scaling is used) is the set of all points in a cube, as shown in Figure 1. And, of course, I also have long known that other colors can be created from combinations or red, green, and blue, or from combinations of cyan,

2. It's beside the point of this column, but I will mention that Carey Bunks is an old friend from the MIT Juggling Club.

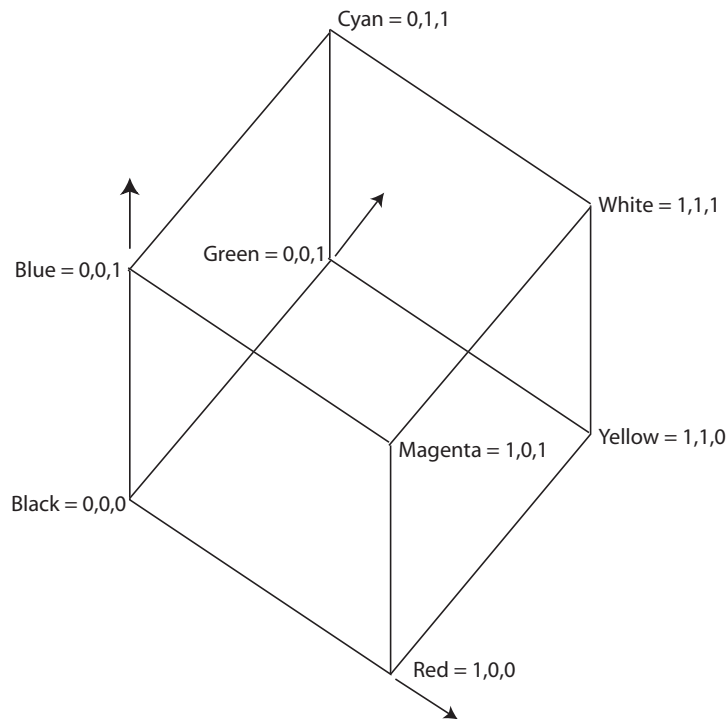


Figure 1: RGB color cube, derived from page 149 of *Grokking the GIMP*

yellow, and magenta if that color model is used. What I never comprehended before was how all of these points are part of the same cube, as shown by the following figure drawn from page 149 of Bunks' book:

As I understand Bunks' explanation, the points on the diagonal of the cube from 0,0,0=Black to 1,1,1=White are the gray scale.

Thus, I modified my Perl program to shown the faces of the RGB cube and the gray diagonal; see `gen-color-r.pl` on the HTML page for this paper³ and see the resulting PDF in `colors-r.pdf`.

Having seen what surfaces of the cube and the Black-White diagonal looked like, I wondered how to think about the rest of the points inside the cube. Carey's book helped by providing a explanation of the translation between the RGB and HSV models. HSV is the hue-saturation-value model (where value is a measure

3. I kept the program simple for you to read by just straight lining the code to generate the various views of the color cube, rather than calling a subroutine with various values of \$r, \$g, and \$b/.

of brightness). V is a position along the Black-White diagonal where $0,0,0$ is completely dark and $1,1,1$, is completely light. One then imagines a plane extending perpendicular from the Black-White diagonal at point V . On this plane one draws a circle of radius S around the Black-White diagonal. This circle is a set of points more or less close to the Black-White diagonal. Close (small value of S) is very unsaturated (not very colorful — close to gray); far (large value of S) is very saturated (very colorful). Finally, one imagines an angle H from the Red axis, with its origin at the Black-White diagonal. The point on the circle at angle H is a point of color in the RGB cube.

The technique described above was pretty easy and didn't require me to learn much new about \LaTeX . However, I probably should also learn to do what I have done in this section using in \LaTeX only `xcolor` or only `PSTRICKS`. Maybe that can be something I try for my next column.

3 Another example of using \LaTeX with an external processor

My wife and I had long maintained a hard copy address book in which we (mostly she) wrote the addresses of people on our Christmas card list and addresses of other people. Several times the book has had to be rewritten when it ran out of pages, there were too many strikeouts, or the pages were simply falling out from long use. The last time the address book was in need of replacement — a couple of years ago — I convinced my wife (or at least she grudgingly agreed) that we should maintain the address list on-line going forward.

So we retyped our address book as a plain text computer file using a format I made up; see the file named `addresses.txt` on the HTML page for this paper for an example page from the on-line address book. The Perl program `label.pl` (see HTML page) converts the address book into a `.tex` file, e.g., `test.tex`, that is input into `labels.tex` which processes it into the format for adhesive backed address labels bought from the Avery company. (I found the `labels.tex` file somewhere on the Internet. Probably for this Christmas I will write my own version of this \LaTeX program so it will be easier for me to tweak it to do what I want it to do for different size labels.)

Biographical note

David Walden is retired after a career as an engineer, engineering manager, and general manager involved with research and development of computer and other high tech systems. He holds an undergraduate math degree and completed a graduate school sequence of courses in computer science. More history is at www.walden-family.com/dave.