Spectrum

Wolfgang May
Institut für Informatik,
Universität G"ottingen
Germany
may@informatik.uni-goettingen.de

2003/12/28

Abstract

The Spectrum style allows running colors following the *hsb (hue-saturation-brightness)* color model or fading out and fading in in a text. In addition to change the color with every letter, it is possible to change it with every word, every *group*, every line, or with every line in a tabular. The style requires the dvips package.

1 The Functionality and Commands

Typesetting text with running or fading letters is achieved by following a path through the *hsb (hue-saturation-brightness)* color model: By changing the *hue* value with constant saturation and brightness, the colors run through the optical spectrum (i.e. the rainbow colors). When changing the *brightness* value with constant hue and saturation, a color fades in and out.
The user interface is surprisingly simple: the style provides principally $2 \cdot 6 = 12$ commands with a closely related syntax: Text passages can be either spectrum or faded by the commands

- `\spec`⟨*granularity*⟩`{`⟨*text*⟩`}` where ⟨*granularity*⟩ gives the granularity of coloring steps: letters, words, or expressions (i.e. LaTeX groups) in which the coloring of ⟨*text*⟩ changes,
- `\begin{speclines}` ...`\end{speclines}` where the text in-between is colored a new color with every line,
- `\begin{spectabular}[`⟨*pos*⟩`]{`⟨*column-spec*⟩`}` ...`\end{spectabular}` or
- `\begin{specarray}[`⟨*pos*⟩`]{`⟨*column-spec*⟩`}` ...`\end{specarray}` where a tabular or arry is typeset with changing the color with every line.
- The analogous commands `\fade`⟨*granularity*⟩ provide fading colors.

Additionally, up to three optional arguments can be given to specify the range and stepwidth of colors:

- In the following, c_1, c_1, and s are natural numbers. The spectrum is mapped to the range from 0 (red) to 100 (again, red). For fading, 0 means black (no brightness) and 100 means white (very bright).
- `\specletters[`c_1`][`c_2`]{`$\langle text \rangle$`}`: The coloring starts by c_1 and finally reaches c_2 at the end of the text. The stepwidth is computed by dividing the specified range by the number of steps which is automatically computed by analyzing $\langle text \rangle$. This can only be done for granularity *letters*, *words* and *expressions*.
- `\specletters[`c_1`][`$\pm s$`]{`$\langle text \rangle$`}`: The coloring starts by c_1 and is increased/decreased by s with every step. The hsb-color palette is passed through cyclically.
- `\specletters[`c_1`][`c_2`][`$\pm s$`]{`$\langle text \rangle$`}`: The coloring starts by c_1 and is increased/decreased by s with every step. When c_2 is reached, the direction is inverted. Thus, the coloring alternates inside the given scope.

In addition, the current color is always accessible by pstricks' color setting commands as `linecolor` or `fillcolor`. Although, when a spectrum command is active, `linecolor` is automatically set to the current color. Thus, this feature is mainly needed for fillcolor (see example later).
Moreover, `\spectrumcolorval` outputs the current value of hue or saturation, dependent on which parameter is running.

Letters.
`\specletters[50][90]{specletters is very ...}` specletters is very easy: the color is incremented with every letter. Note that spaces do not add an additional step. On the other hand, one should not forget to reset the color with `\black` after the end of the spectrum's scope (if needed). `\specletters[90][50]{it can also ...}` it can also be run downwards. In addition, since the spectrum's text is shorter, the steps have been bigger.

Words.
`\specwords[10][-5]{Now ...}`Now, every word is set with the same color, the color is changed between the words. If the text is long enough, the color range is passed cyclically, returning to red and entering another round.

Expressions
`\specexpr[50][80][+5]{{now,} {every expression} {is} {typeset} {with} {the} {same} {color.} {It} {also} {allows} {\LaTeX} {commands} {inside} {the} {spectrum} {text.}}` now, every expression is typeset with the same color. It also allows LaTeX commands inside the spectrum text.
Additionally, this paragraph it shows that the color now moves up and down inside the specified range.
`\specexpr[50][80][+5]{if there are no groups, every letter and every space is treated as an expression.}` if there are no groups, every letter and every space is treated as an expression.
The latter is especially useful when typesetting non-english texts: many languages have *umlauts* which produce errors with `\specletters`. See the later examples.

Lines
\begin{speclines}[40][60][+4] In this paragraph, every line is typeset in another color. Note that a fixed stepwidth is given and the colors move up and down the specified range. This is done automatically by a modified output routine.
\end{speclines}[40][60][+4]

Tabular The following table is typeset by ...

```
\begin{spectabular}[38][-2][t]{llrrrrrr}
1. BV 09 Borussia Dortmund & 34 & 19 & 11 &  4 & 76-38 & +38 & 68 \\
⋮
18. Krefelder FC Uerdingen & 34 &  5 & 11 & 18 & 33-56 & -23 & 26 \\
\end{spectabular}.
```
[1]

1.	BV 09 Borussia Dortmund	34	19	11	4	76-38	+38	68
2.	FC Bayern München	34	19	5	10	66-46	+20	62
3.	FC Schalke 04	34	14	14	6	45-36	+ 9	56
4.	Borussia Mönchengladbach	34	15	8	11	52-51	+ 1	53
5.	Hamburger SV	34	12	14	8	52-47	+ 5	50
6.	FC Hansa Rostock	34	13	10	11	47-43	+ 4	49
7.	Karlsruher SC	34	12	12	10	53-47	+ 6	48
8.	TSV München 1860	34	11	12	11	52-46	+ 6	45
9.	SV Werder Bremen	34	10	14	10	39-42	- 3	44
10.	VfB Stuttgart	34	10	13	11	59-62	- 3	43
11.	SC Freiburg	34	11	9	14	30-41	-11	42
12.	1. FC Köln	34	9	13	12	33-35	- 2	40
13.	TSV Fortuna Düsseldorf	34	8	16	10	40-47	- 7	40
14.	TSV Bayer 04 Leverkusen	34	8	14	12	37-38	- 1	38
15.	FC St. Pauli Hamburg	34	9	11	14	43-51	- 8	38
16.	1. FC Kaiserslautern	34	6	18	10	31-37	- 6	36
17.	Eintracht Frankfurt	34	7	11	16	43-68	-25	32
18.	Krefelder FC Uerdingen	34	5	11	18	33-56	-23	26

Optional Arguments. If some of the optional arguments are missing, the values of the previous call to \spectrum or \fade are used. Here, especially the case that *no* argument is given – i.e. there is even no starting value – is of interest: Then, the current value is used, and the previous sequence is continued, either cyclic or alternating.
\specletters[20][+1]{This sequence rund from light green into dark blue.} This sequence starts runs from light green into dark blue. The next sequence also uses umlauts, continuing the color sequence from the first one: \specexpr{{C'}est une suite de bleu fonc{\'e} {\`a} rouge}. C'est une suite de bleu foncé à rouge. It can be continued \specexpr{bis schlie{\ss}lich wieder gr{\"u}n erreicht ist.} bis schließlich wieder grün erreicht ist.
\specexpr{Since the primary statement yielded to a cyclic coloring, this is continued through the other sequences. Since the primary statement yielded to a cyclic coloring, this is continued through the other sequences.
The same can be done with alternating sequences: \specexpr[60][100][+4]{Det {\"a}r en f{\"o}ljd som alltid alternerar mellan bl{\aa} och r{\"o}{d.}}

[1] InGermany,thelastthreeteamsarerelegated.Kaiserslauternmanagedtobere-promotedthesubsequentyearandbecomechampiontwoyearslater,i.e. in1998.

Det är en följd som alltid alternerar mellan blå och röd. `\specletters{and it can be continued later, still alternating with the same scope.}` and it can be continued later, still alternating with the same scope.
If only one argument is given, it is taken as a new initial value, retaining the scope of the previous command: `\specletters[80]{e.g., starting again with pink and alternating between blue and red.}` e.g., starting again with pink and alternating between blue and red.
If the initial sequence is given by its start and end value, i.e. `\specletters[40][70]{the stepwidth is computed automatically}`, the stepwidth is computed automatically, `\specletters{the continuation alternates between these values with the computed stepwidth.}` the continuation alternates between these values with the computed stepwidth.
`\specletters[30][+3]{A cyclic sequence can also be continued}` A cyclic sequence can also be continued. `\specletters[60]{The continuation starts with the new value (60=blue), using the same stepwidth.}` The continuation starts with the new value (60=blue), using the same stepwidth.

`\specexpr` is of special interest for coloring elements of figures for presentations. The following is taken from a PhD thesis on active database languages at our chair.

The Spectre of Rule-Based Database Languages

Deductive Rules *Production Rules* *Active Rules*

declarative **procedural**

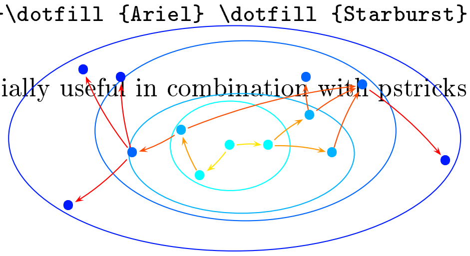
S-Datalog WF-Datalog . RDL I-/P-Datalog . A-RDL . Ariel . Starburst . Postgres

produced by the following source code:

```
\psset{arrowsize=2pt 3}
\psset{nodesep=3pt}
\psset{linewidth=2pt}
\newcommand{\block}{\rule[0.12em]{0.45em}{0.4em}}
\newcommand{\lefta}{\Rnode{l}{}~~~\Rnode{r}{}\ncline{<-}{l}{r}}
\newcommand{\righta}{\Rnode{xl}{}~~~\Rnode{xr}{}\ncline{->}{xl}{xr}}
\paragraph{The Spectre of Rule-Based Database Languages}\ \\\bigskip
\specexpr[25][95]
{{\emph{Deductive Rules}}\hfill
 {\emph{Production Rules}}\hfill
 {\emph{Active Rules}}}\\\bigskip
\specexpr[25][95]{{\textbf{declarative}}\hfill{\textbf{procedural}}} \\\bigskip
{\psset{arrowsize=4pt 6}\psset{fillstyle=hlines}
\specexpr[25][95]{{\lefta}\block\block\block\block\block\block\block
\block\block\block\block\block\block\block\block\block\block\block
\block\block\block\block\block\block\block\block\block\block\block
\block\block\block\block\block\block\block\block\block\block\block
\block\block\block\block\block\block\block\block\block\block\block
\block\block\block\block\block\block\block\block\block\block\block
\block\block\block\block\block\block\block\block\block\block\block
```

```
\block{\righta}}}\black\\\bigskip
\specexpr[25][95]
{{\begin{tabular}{@{}c@{}}
    S-Datalog\\
    WF-Datalog
  \end{tabular}}
  \dotfill
  {\begin{tabular}{@{}c@{}}
    RDL\\
    I-/P-Datalog
  \end{tabular}}
  \dotfill {A-RDL}\dotfill {Ariel} \dotfill {Starburst} \dotfill {Postgres}
  }
```

`\specexpr` is especially useful in combination with pstricks commands:



I I

1.1 Shortcomings and Incompatibilities

If a spectrum sequence distributes over two pages and a footnote appears on the first of those pages, the footnote is colored instead of the first letter of appearing on the next page.
Since for `\speclines`, the output routine is modified, there are many incompatibilities.

2 Code Documentation

3 Documentation of the Macros

```
\typeout{Style `\basename', Version \fileversion\space <\filedate>}
\ProvidesPackage{spectrum}[\filedate \space\fileversion]
```

The color is set by generating raw postscript output, here, commands of the form h s b sethsbcolor where h (hue), s (saturation), and b (brightness) are values between 0 and 1. The internal granularity of color/saturation/brightness is 10000.

```
\let\eop=\par
\newcount\spec@currentcolor\global\spec@currentcolor -1
\newcount\spec@currentsatur\global\spec@currentsatur 9999
\newcount\spec@currentbright\global\spec@currentbright 9999
```

Setting default values: color=-1 means black/white. Saturation 9999 means at least 1, thus full color saturation; analogous for brightness.

```
\newcount\spec@colorval
\newcount\spec@startvalue\global\spec@startvalue 0
\newcount\spec@endvalue\global\spec@endvalue 10000
\newcount\spec@leap
\newcount\exprlength
\newcount\@tempcnt
\newtoks\spec@tabpos\spec@tabpos{c}
\newif\ifspec@cyclic\global\spec@cyclictrue
\def\spec@thespace{ }
```

3.1 Parsing the LaTeX Source

```
\def\spectrum@lines@string{lines}
\def\spectrum@tabular@string{tabular}

\def\spectrum#1{\begingroup\obeyspaces
      \gdef\spec@what{color}%
      \global\spec@currentsatur 9999%
      \global\spec@currentbright 9999%
      \gdef\spec@granularity{#1}%
      \@ifnextchar[%]
        {\spec@read@firstarg}%
        {\spec@default@firstarg}}
```

Spaces are obeyed in the argument. Calling \spectrum<gran> makes color the running parameter and sets saturation and brightness to maximal. Granularity is set to <gran>. If no optional argument is given, as a default, the current value of the running parameter is taken.

```
\def\fade#1{\begingroup\obeyspaces
      \gdef\spec@what{bright}%
      \ifnum\spec@currentcolor=-1\global\spec@currentsatur 0%
       \else\global\spec@currentsatur 9999\fi
      \gdef\spec@granularity{#1}%
      \@ifnextchar[%]
        {\spec@read@firstarg}%
        {\spec@default@firstarg}}
```

Analogous to \spectrum<gran>: Fading makes brightness the running parameter. If color is black/white (-1), saturation is set to 0, else to maximal (9999).

```
\def\spec@read@firstarg[#1]{%
  \ifcat#1\noexpand\the
     \global\csname spec@current\spec@what\endcsname=#1%
     \global\multiply\csname spec@current\spec@what\endcsname by100%
  \else
```

```
37      \ifcat#19\global\csname spec@current\spec@what\endcsname=#1%
38         \global\multiply\csname spec@current\spec@what\endcsname by100%
39       \else\spec@check@tabular#1\fi\fi
40    \spec@default@firstarg}
```

If the optional argument is `c`, `t`, or`b`, then it is due to an array specification. In this case, `\spec@check@tabular` is called (see line 268). Else, the optional argument, multiplied by 100, gives the `\spec@startvalue`. The current value of the running parameter (given by `\spec@what`) is set to `\spec@startvalue`.

```
41 \def\spec@default@firstarg{%
42    \@ifnextchar[%]
43      {\spec@read@secondarg}%
44      {\spec@exec}}%
```

If no other optional argument (providing either the upper bound of the spectrum/brightness range) follows, the old value of `\spec@leap` is taken as default value.

```
45 \def\spec@read@secondarg[%]
46   {\global\spec@cyclictrue
47    \@ifnextchar+{\spec@read@stepsize}%
48    {\@ifnextchar-{\spec@read@stepsize}%
49     {\@ifnextchar c{\spec@read@tabpos}%
50      {\@ifnextchar t{\spec@read@tabpos}%
51       {\@ifnextchar b{\spec@read@tabpos}{\spec@read@scope}}}}}}%
```

Again, we have to take care about optional tabular positioning. Otherwise, it is checked if a positive or negative step amount is given (branching to `\spec@read@stepsize`). If no, the optional parameter gives an upper bound of the range of the running parameter (branching to `\spec@read@scope`).

```
52 \def\spec@read@tabpos#1]{\spec@check@tabular{#1}\spec@exec}
```

In case of a tabular positioning argument, it is read and `\spec@check@tabular` is called (see line 268).

```
53 \def\spec@read@scope#1]{%
54    \global\spec@startvalue\csname spec@current\spec@what\endcsname
55    \global\spec@endvalue=#1%
56    \global\multiply\spec@endvalue by100%
57    \global\spec@cyclicfalse
58    \@ifnextchar[%]
59       {\spec@read@leap}%
60       {\spec@calc@leap}}
```

Otherwise, the optional argument, multiplied by 100, gives the upper bound of the range of the running parameter; in this case, the spectrum is not cyclic but has to change its running direction when reaching one of the bounds.
If another optional argument follows, this gives the stepwidth or is a tabular positioning argument, else, the stepwidth is computed using the length of the spectrum text.

```
61 \def\spec@read@leap[%]
62   {\@ifnextchar+{\spec@read@stepsize}%
63    {\@ifnextchar-{\spec@read@stepsize}%
64     {\@ifnextchar c{\spec@read@tabpos}%
65      {\@ifnextchar t{\spec@read@tabpos}%
66       {\@ifnextchar b{\spec@read@tabpos}%
67       {\PackageError{\basename}{+/- required in third opt arg}\@ehc}}}}}}
```

Checking if the argument gives a tabular positioning or a stepwidth.

```
\def\spec@read@stepsize#1]{%
   \global\spec@leap=#1%
   \global\multiply\spec@leap by100%
   \spec@exec}
```

Stepwidth is multiplied internally by 100, and the spectrum typesetting is started by calling `\spec@anyth@exec`.

```
\def\spec@calc@leap#1{%
   \ifx\spec@granularity\spectrum@lines@string
     {\PackageError{\basename}{Scope with lines requires stepsize}\@ehc}\fi
   \ifx\spec@granularity\spectrum@tabular@string
     {\PackageError{\basename}{Scope with tabular requires stepsize}\@ehc}\fi
   \spec@scope@calc@leap{#1}%
   \spec@exec{#1}}
```

This is executed if a range but no stepwidth is give, In this cases, the stepwidth is computed using the length of the spectrum text. This is not possible for granularity “lines” or “tabular”. Otherwise, the stepwidth is computed and the spectrum typesetting is started.

```
\def\spec@exec{%
    \spec@sethsb
    \csname spec@\spec@granularity GoOn\endcsname}
```

At first, the current hsb-color is set, then the typesetting is started with the appropriate granularity.

```
\def\spec@End#1{\endgroup}%
```

at the end, the `groupp` must be closed, and one character must be thrown away.

```
\def\spec@scope@calc@leap#1{%
    \exprlength=0%
    \csname cardinality@\spec@granularity\endcsname#1\eop
    \spec@leap\spec@endvalue
    \advance\spec@leap by -\spec@startvalue
    \global\divide\spec@leap by\exprlength}

\def\endcardinality#1{}
```

Again, at the end, one character must be thrown away.

3.2 Generating postscript output

```
\@ifundefined{Raw@PS}{\def\Raw@PS{\Gin@PS@raw}}{\relax}
```

supplies for compatibility with old versions of dvips.

```
\def\spec@setvalue#1{.\ifnum\csname#1\endcsname<1000 0\fi
          \ifnum\csname#1\endcsname<100 0\fi
          \ifnum\csname#1\endcsname<10 0\fi
          \the\csname#1\endcsname 0}
```

Given a value n between 0 and 9999, this produces the output $n/10000$ in decimal, for evaluation by the `sethsbcolor` command of postscript.

`\sethsb`

```
\def\spec@sethsb{%
  \begingroup
  \@tempcnt\spec@currentcolor
    \divide\@tempcnt by 10000%
    \multiply\@tempcnt by -10000%
```

```
100     \advance\@tempcnt by \spec@currentcolor
101     \spec@currentcolor\@tempcnt
102   \ifnum\spec@currentcolor=-1\spec@currentcolor0\fi%
103   \@tempcnt\spec@currentbright
104     \divide\@tempcnt by 10000%
105     \multiply\@tempcnt by -10000%
106     \advance\@tempcnt by \spec@currentbright
107     \spec@currentbright\@tempcnt
108   \Raw@PS{%
109       \spec@setvalue{spec@currentcolor}
110       \spec@setvalue{spec@currentsatur}
111       \spec@setvalue{spec@currentbright}
112       sethsbcolor}%
113   \global\def\color@spectrumcolor{%
114       \spec@setvalue{spec@currentcolor}
115       \spec@setvalue{spec@currentsatur}
116       \spec@setvalue{spec@currentbright}
117       sethsbcolor}%
118   \endgroup
119   \psset{linecolor=spectrumcolor}
120   \global\spec@colorval=\the\csname spec@current\spec@what\endcsname
121   \divide\spec@colorval by 100}
122 \def\spectrumcolorval{\the\spec@colorval}
```

Lines 95–106: Locally calculates values of hue and brightness modulo 10000. `\spec@currentcolor=-1` is the black/white switch – any nonnegative hue value does the work saturation is 0 then.

Lines 107–112: produces output of the form `h s b sethsbcolor` to be interpreted by postscript

Lines 113–117: defines `\color@spectrumcolor` to do exactly this when called. `\color@spectrumcolor` is e.g. called by `linecolor=spectrumcolor` in pstricks, thus making the current color also accessible for pstricks commands.

Line 118: sets the pstricks attribute linecolor to the current color.

Lines 119,120: sets `\spec@colorval` to the current color value with granularity 100.

Line 121: makes the current color value accessible to the user in the `\spectrumcolorval` counter.

`\step` makes a step in the spectrum.

```
123 \def\spec@step{%
124   \global\advance\csname spec@current\spec@what\endcsname by\spec@leap%
125   \ifspec@cyclic
126   \ifnum\csname spec@current\spec@what\endcsname<0%
127         \global\advance\csname spec@current\spec@what\endcsname by 10000\fi
128   \ifnum\csname spec@current\spec@what\endcsname>9999%
129         \global\advance\csname spec@current\spec@what\endcsname by -10000\fi
130   \else
131     \ifnum\spec@leap<0%
132      \ifnum\csname spec@current\spec@what\endcsname<\spec@endvalue%
133         \global\spec@leap-\spec@leap
134         \global\advance\csname spec@current\spec@what\endcsname by \spec@leap
135         \global\advance\csname spec@current\spec@what\endcsname by \spec@leap
136         \spec@swapstartend
```

```
137         \fi\fi
138     \ifnum\spec@leap>0%
139      \ifnum\csname spec@current\spec@what\endcsname>\spec@endvalue%
140         \global\spec@leap-\spec@leap
141         \global\advance\csname spec@current\spec@what\endcsname by \spec@leap
142         \global\advance\csname spec@current\spec@what\endcsname by \spec@leap
143         \spec@swapstartend
144         \fi\fi
145   \fi
146   \spec@sethsb}
```

Line 124: increments color or brightness by `\spec@leap`.

Lines 125–140: if the values are not intended to be cyclic, it must be checked if the end value is already reached. If so, the sign of `\spec@leap` is inverted (changing the direction) and start and end value are swapped.

Lines 141–146: checks the range to be between 0 and 9999.

```
147 \def\spec@swapstartend{\@tempcnta\spec@endvalue
148         \global\spec@endvalue\spec@startvalue
149         \global\spec@startvalue\@tempcnta}
```

macro to exchange start and end value.

3.3 Granularity: Expressions

```
150 \def\specexpr{\spectrum{expr}}
151 \def\fadeexpr{\fade{expr}}
152 \def\cardinality@expr{\take@expr@next}
153 \def\take@expr@next{\futurelet\nextToken\take@@expr@next}
154 \def\take@@expr@next{%
155   \ifx\nextToken\eop
156     \advance\exprlength by-1\let\nextaction\endcardinality\else
157    \let\nextaction=\take@@@expr@next
158      \advance\exprlength by1\fi
159   \nextaction}
160 \def\take@@@expr@next#1{%
161   \ifx#1\spec@thespace\advance\exprlength by-1\fi
162   \take@expr@next}
```

Count the number of expressions in the spectrum scope by a loop.

```
163 \def\spec@exprGoOn#1{\spec@exprDoIt#1\eop}
164 \def\spec@exprDoIt{\futurelet\nextToken\spec@exprLookAtToken}
165 \def\spec@exprLookAtToken{%
166   \ifx\nextToken\eop\let\nextaction=\spec@End\else
167     \let\nextaction=\spec@exprOutputAndNextExpr\fi
168   \nextaction}%
```

Separate the spectrum expressions. Every expression is typeset by `\spec@exprOutputAndNextExpr`.

```
169 \def\spec@exprOutputAndNextExpr#1{#1%
170   \ifx#1\spec@thespace\else\spec@step\fi
171   \spec@exprDoIt}
```

Typeset the expression, if it is not a space, take the next color and go on. This ignores spaces not always correctly.

3.4 Granularity: Words

```
\def\specwords{\spectrum{words}}
\def\fadewords{\fade{words}}
```

```
\def\cardinality@words{\take@words@next}
\def\take@words@next{\futurelet\nextToken\take@@words@next}
\def\take@@words@next{%
   \ifx\nextToken\eop\let\nextaction\endcardinality\else
   \let\nextaction=\take@@@words@next
   \ifcat a\nextToken\else
     \ifcat -\nextToken\else
       \advance\exprlength by1\fi\fi\fi
   \nextaction}
\def\take@@@words@next#1{\take@words@next}
```

Count the number of words in the spectrum scope by a loop.

```
\def\spec@wordsGoOn#1{\spec@wordsDoIt#1\eop}
\def\spec@wordsDoIt{\futurelet\nextToken\spec@wordsLookAtToken}
\def\spec@wordsLookAtToken{%
  \ifx\nextToken\eop\let\nextaction=\spec@End\else
   \ifcat a\nextToken\let\nextaction=\spec@wordsOutput\else
    \ifcat -\nextToken\let\nextaction=\spec@wordsOutput\else
      \let\nextaction=\spec@wordsNextWord\fi\fi\fi
  \nextaction}%
```

Separate the spectrum words. Every single letter is typeset by `\spec@wordsOutput`. At the end of a word, `\spec@wordsNextWord` is called.

```
\def\spec@wordsOutput#1{#1\spec@wordsDoIt}
\def\spec@wordsNextWord#1{#1\spec@step\spec@wordsDoIt}
```

`\spec@wordsOutput` typesets a single letter. `\spec@wordsNextWord` is called at the end of a word, typesetting the current symbol (in general, this is a space) and taking the next color.

3.5 Granularity: Letters

```
\def\specletters{\spectrum{letters}}
\def\fadeletters{\fade{letters}}
```

```
\def\cardinality@letters{\take@letters@next}
\def\take@letters@next{\futurelet\nextToken\take@@letters@next}
\def\take@@letters@next{%
  \ifx\nextToken\eop
   \advance\exprlength by-1\let\nextaction\endcardinality\else
   \let\nextaction=\take@@@letters@next
    \ifcat a\nextToken\advance\exprlength by1\else
     \ifcat -\nextToken\advance\exprlength by1\fi\fi\fi
   \nextaction}
\def\take@@@letters@next#1{\take@letters@next}
```

Count the number of letters in the spectrum scope by a loop.

```
\def\spec@lettersGoOn#1{\spec@lettersDoIt#1\eop}
\def\spec@lettersDoIt{\futurelet\nextToken\spec@lettersLookAtToken}
\def\spec@lettersLookAtToken{%
  \ifx\nextToken\eop\let\nextaction=\spec@End\else
  \ifcat a\nextToken\let\nextaction=\spec@lettersOutput\else
    \ifcat -\nextToken\let\nextaction=\spec@lettersOutput\else
      \let\nextaction=\spec@lettersNextWord\fi\fi\fi
  \nextaction}%
```

Handle individual letters. Every single letter is typeset by `\spec@lettersOutput`. At the end of a word, `\spec@lettersNextWord` is called.

```
\def\spec@lettersOutput#1{#1\spec@step\spec@lettersDoIt}
\def\spec@lettersNextWord#1{ \spec@lettersDoIt}
```

`\spec@lettersOutput` takes the next color and typesets a single letter. `\spec@lettersNextWord` is called at the end of a word, typesetting the current symbol (in general, this is a space), not taking the next color.

3.6 Granularity: Lines

The following is adapted from `lineno.sty`:

```
\def\speclines{\spectrum{lines}}
\def\fadelines{\fade{lines}}
```

The colors have to be attached by the output routine. We simply set the `\interlinepenalty` to -100000. The output routine will be called after each line in the paragraph, except the last, where we trigger by `\par`. The `\spec@penalty` is small enough to compensate a bunch of penalties (e.g., with `\samepage`).

```
\newcount\spec@penalty\spec@penalty=-100000
\mathchardef\spec@penaltypar=25000
```

`\output` So let's make a hook to `\output`, the direct way. The LaTeX macro `\@reinserts` puts the footnotes back on the page.

```
\output=\expandafter{%
        \expandafter \spec@LineBrTest
        \expandafter \if@tempswa
                        \the\output
                      \else
                        \@reinserts
                        \MakeLineStep
                      \fi}
```

The float mechanism inserts `\interlinepenalty`s during `\output`. So carefully reset it before going on. Else we get double color step on every float placed in horizontal mode.

Neither a `\linelabel` nor a `\marginpar` should insert a penalty. Nor should any other float. So let us suppress the `\interlinepenalty` altogether with the `\@nobreak` switch.

Since (ltspace.dtx, v1.2p)[1996/07/26], the `\@nobreaktrue` does it's job globally. We need to do it locally here.

`LineBreakTest`

```
\def\my@nobreaktrue{\let\if@nobreak\iftrue}
\def\spec@LineBrTest{%
  \ifnum\interlinepenalty<-\spec@penaltypar
     \advance\interlinepenalty-\spec@penalty
     \my@nobreaktrue
     \fi
  \@tempswatrue
  \ifnum\outputpenalty>-\spec@penaltypar\else
     \ifnum\outputpenalty>-175000\relax
       \@tempswafalse
       \fi
```

```
    \fi
  }
```

put in the natural `\interlinepenalty`, except after the last line.

`\MakeLineStep`

```
\def\MakeLineStep{\unvbox\@cclv
   \nointerlineskip
   \ifnum\outputpenalty=-\spec@penaltypar\else
       \@tempcnta\outputpenalty
       \advance\@tempcnta -\spec@penalty
       \penalty\@tempcnta
   \fi
   \spec@step\spec@sethsb}
```

The line numbering is controlled via `\par`. LaTeX saved the TeX-primitive `\par` in `\@@par`. We push it one level further out, and redefine `\@@par` to insert the `\interlinepenalty` needed to trigger the color step And we need to allow page-breaks after a paragraph.

```
\let\@@@par\@@par
\def\spec@linepar{\ifvmode\@@@par\else\ifinner\@@@par\else
     \advance\interlinepenalty\spec@penalty
        \@@@par
        \penalty-\spec@penaltypar
        \kern\z@
     \advance\interlinepenalty -\spec@penalty
     \fi\fi\everypar{\leavevmode\spec@sethsb}}
```

The basic commands to enable and disable color steps: `\@par` and `\par` are only touched, when they are `\let` to `\@@@par`/`\spec@linepar`.

`LinesGoOn`

```
\def\spec@linesGoOn{%
     \let\@@par\spec@linepar
     \ifx\@par\@@@par\let\@par\spec@linepar\fi
     \ifx\par\@@@par\let\par\spec@linepar\fi}
%
\def\endspeclines{\let\@@par\@@@par
  \ifx\@par\spec@linepar\let\@par\@@@par\fi
  \ifx\par\spec@linepar\let\par\@@@par\fi
  \spec@End{}}
\let\endfadelines\endspeclines
```

3.7 Granularity: Arrays and Tabulars

```
\def\spectabular{\spec@tabpos{c}\spectrum{tabular}}
\def\spec@check@tabular#1{%
  \spec@tabpos{#1}%
  \ifx\spec@granularity\spectrum@tabular@string\else
    {\PackageError{\basename}{[c/t/b] only allowed for tabular}\@ehc}\fi}
```

arguments `[c]`, `[t]`, and `[b]` can only occur for tabulars and arrays.

```
\def\spec@tabularGoOn{\let\@halignto\@empty\spec@tabular}
\def\spec@tabular{\leavevmode \hbox \bgroup $\let\@acol\@tabacol
```

```
274   \let\@classz\@tabclassz
275   \let\@classiv\@tabclassiv \let\\\@tabularcr\spec@tabarray}
276 \def\spec@tabarray{\m@th
277   \@ifnextchar[\spec@array{\spec@array[\the\spec@tabpos]}}
```

The original tabular/array environment is copied for the spectrum variant.

```
278 \def\spec@array[#1]#2{%
279   \if #1t\vtop \else \if#1b\vbox \else \vcenter \fi\fi
280   \bgroup
281   \setbox\@arstrutbox\hbox{%
282     \vrule \@height\arraystretch\ht\strutbox
283            \@depth\arraystretch \dp\strutbox
284            \@width\z@}%
285   \@mkpream{#2}%
286   \edef\@preamble{%
287     \ialign \noexpand\@halignto
288       \bgroup
289       \noexpand\spec@step %% inserted for spectrum style
290       \@arstrut \@preamble \tabskip\z@skip \cr}%
291   \let\@startpbox\@@startpbox
292   \let\@endpbox\@@endpbox
293   \let\tabularnewline\\%
294     \let\par\@empty
295     \let\@sharp##%
296     \set@typeset@protect
297     \lineskip\z@skip\baselineskip\z@skip
298     \@preamble}
```

Line 289 is added for the spectrum variant, taking the next color with every line-break.

```
299 \def\endspectabular{\endtabular\spec@End{}}
```

4 Acknowledgements

The modification of the output routine is adapted from Stephan Böttcher's `lineno.sty`. In addition, I would thank those people who answered my questions in the newsgroups comp.text.tex and comp.lang.postscript, especially Denis Girou who maintains this wonderful pstricks stuff. The figure on page 1 is due to my colleague Bertram Ludäscher's PhD thesis.