Visual Debugging in TEX

how things are done

Hans Hagen

September 25 1996

Although an integral part of CONTEXT, this module is one of the support modules. Its stand alone character permits use in PLAIN TEX or TEX based macropackages. If in some examples the verbatim listings don't show up nice, this is due to processing by a system that does not support buffering. In CONTEXT we show the commands in the margin, use bit more advanced way of numbering, and typeset the source in TEXnicolored verbatim. Sorry for this inconvenience.

This module is still in development. Depending on my personal need and those of whoever uses it, the macros will be improved in terms of visualization, efficiency and compatibility.

```
1  1  \ifx \undefined \writestatus \input supp-mis.tex \fi
```

One of the strong points of TEX is abstraction of textual input. When macros are defined well and do what we want them to do, we will seldom need the tools present in What You See Is What You Get systems. For instance, when entering text we don't need rulers, because no manual shifting and/or alignment of text is needed. On the other hand, when we are designing macros or specifying layout elements, some insight in TEX's advanced spacing, kerning, filling, boxing and punishment abilities will be handy. That's why we've implemented a mechanism that shows some of the inner secrets of TEX.

```
2  2  \writestatus{loading}{Context Support Macros / Visualization}
```

In this module we are going to redefine some TEX primitives and PLAIN macro's. Their original meaning is saved in macros with corresponding names, preceded by `normal`. These original macros are (1) used to temporary restore the old values when needed and (2) used to prevent recursive calls in the macros that replace them.

```
3  3  \unprotect
```

There are three types of boxes, one horizontal and two vertical in nature. As we will see later on, all three types are to be handled according to their orientation and baseline behavior. Especially `\vtop`'s need our special attention.

```
4  4  \let\normalhbox     = \hbox
   5  \let\normalvbox     = \vbox
   6  \let\normalvtop     = \vtop
```

Next come the flexible skips, which come in two flavors too. Like boxes these are handled with TEX primitives.

```
5  7  \let\normalhskip    = \hskip
   8  \let\normalvskip    = \vskip
```

Both penalties and kerns are taken care of by mode sensitive primitives. This means that when making them visible, we have to take the current mode into account.

```
6  9  \let\normalpenalty  = \penalty
  10  \let\normalkern     = \kern
```

Glues on the other hand are macro's defined in PLAIN TEX. As we will see, their definitions make the implementation of their visible counterparts a bit more TEXnical.

```
7 11  \let\normalhglue    = \hglue
  12  \let\normalvglue    = \vglue
```

Math mode has its own spacing primitives, preceded by `m`. Due to the relation with the current font and the way math is typeset, their unit `mu` is not compatible with other dimensions. As a result, the visual appearance of these primitives is kept primitive too.

```
8 13  \let\normalmkern    = \mkern
  14  \let\normalmskip    = \mskip
```

Fills can be made visible quite easy. We only need some additional negation macros. Because PLAIN TEX only offers `\hfilneg` and `\vfilneg`, we define our own alternative double `ll`'ed ones.

```
\def\hfillneg%
  {\normalhskip\!!zeropoint \!!plus-1fill\relax}

\def\vfillneg%
  {\normalvskip\!!zeropoint \!!plus-1fill\relax}
```

The positive stretch primitives are used independant and in combination with `\leaders`.

```
\let\normalhss      = \hss
\let\normalhfil     = \hfil
\let\normalhfill    = \hfill
\let\normalvss      = \vss
\let\normalvfil     = \vfil
\let\normalvfill    = \vfill
```

Keep in mind that both `\hfillneg` and `\vfillneg` are not part of PLAIN TEX and therefore not documented in standard TEX documentation. They can nevertheless be used at will.

```
\let\normalhfilneg  = \hfilneg
\let\normalhfillneg = \hfillneg
\let\normalvfilneg  = \vfilneg
\let\normalvfillneg = \vfillneg
```

Visualization is not always wanted. Instead of turning this option off in those (unpredictable) situations, we just redefine a few PLAIN macros.

```
\def\rlap#1{\normalhbox to \!!zeropoint{#1\normalhss}}
\def\llap#1{\normalhbox to \!!zeropoint{\normalhss#1}}

\def~{\normalpenalty\!!tenthousand\ }
```

Ruled boxes can be typeset is many ways. Here we present just one alternative. This implementation may be a little complicated, but it supports all three kind of boxes. The next command expects a ⟨*box*⟩ specification, like:

```
\makeruledbox0
```

We can make the baseline of a box visible, both dashed and as a rule. Normally the line is drawn on top of the baseline, but a smashed alternative is offered too. If we want them all, we just say:

```
\baselineruletrue
\baselinefilltrue
\baselinesmashtrue
```

At the cost of some overhead these alternatives are implemented using `\if`'s:

```
\newif\ifbaselinerule  \baselineruletrue
\newif\ifbaselinefill  \baselinefillfalse
\newif\ifbaselinesmash \baselinesmashfalse
```

Rules can be turned on and off, but by default we have:

```
\topruletrue
\bottomruletrue
\leftruletrue
\rightruletrue
```

As we see below:

```
\newif\iftoprule       \topruletrue
\newif\ifbottomrule    \bottomruletrue
\newif\ifleftrule      \leftruletrue
\newif\ifrightrule     \rightruletrue
```

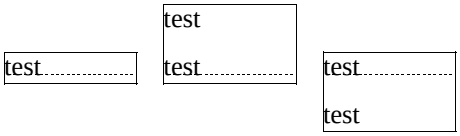
The width in the surrounding rules can be specified by assigning an apropriate value to the dimension used. This module defaults the width to:

```
\boxrulewidth=.2pt
```

Although we are already low on ⟨*dimensions*⟩ it's best to spend one here, mainly because it enables easy manipulation, like multiplication by a given factor.

```
\newdimen\boxrulewidth \boxrulewidth=.2pt
```

The core macro `\makeruledbox` looks a bit hefty. The manipulation at the end is needed because we want to preserve both the mode and the baseline. This means that `\vtop`'s and `\vbox`'es behave the way we expect them to do.



The `\cleaders` part of the macro is responsible for the visual baseline. The `\normalhfill` belongs to this primitive too. By storing and restoring the height and depth of box `#1`, we preserve the mode.

```
\def\makeruledbox#1%
  {\edef\ruledheight {\the\ht#1}%
   \edef\ruleddepth  {\the\dp#1}%
   \edef\ruledwidth  {\the\wd#1}%
   \setbox\scratchbox=\normalvbox
     {\dontcomplain
      \offinterlineskip
      \hrule
        \!!height\boxrulewidth
        \iftoprule\else\!!width\!!zeropoint\fi
      \normalvskip-\boxrulewidth
      \normalhbox to \ruledwidth
         {\vrule
            \!!height\ruledheight
            \!!depth\ruleddepth
            \!!width\ifleftrule\else0\fi\boxrulewidth
          \ifdim\ruledheight>\!!zeropoint \else \baselinerulefalse \fi
          \ifdim\ruleddepth>\!!zeropoint \else \baselinerulefalse \fi
          \ifbaselinerule
            \ifdim\ruledwidth<20\boxrulewidth
              \baselinefilltrue
            \fi
            \cleaders
              \ifbaselinefill
                \hrule
                  \ifbaselinesmash
                    \!!height\boxrulewidth
                  \else
                    \!!height.5\boxrulewidth
                    \!!depth.5\boxrulewidth
                  \fi
              \else
                \normalhbox
                  {\normalhskip2.5\boxrulewidth
                   \vrule
                   \ifbaselinesmash
                     \!!height\boxrulewidth
                   \else
                     \!!height.5\boxrulewidth
                     \!!depth.5\boxrulewidth
                   \fi
                   \!!width5\boxrulewidth
                   \normalhskip2.5\boxrulewidth}%
              \fi
          \fi
          \normalhfill
          \vrule
            \!!width\ifrightrule\else0\fi\boxrulewidth}%
        \normalvskip-\boxrulewidth
        \hrule
          \!!height\boxrulewidth
          \ifbottomrule\else\!!width\!!zeropoint\fi}%
   \wd#1=\!!zeropoint
   \setbox#1=\ifhbox#1\normalhbox\else\normalvbox\fi
     {\normalhbox{\box#1\lower\ruleddepth\box\scratchbox}}%
   \ht#1=\ruledheight
   \wd#1=\ruledwidth
   \dp#1=\ruleddepth}
```

Just in case one didn't notice: the rules are in fact layed over the box. This way the contents of a box cannot visually interfere with the rules around (upon) it. A more advanced version of ruled boxes can be found in one of the core modules of CONTEXT. There we take offsets, color, rounded corners, backgrounds and alignment into account too.

These macro's can be used instead of `\hbox`, `\vbox` and `\vtop`. They just do what their names state. Using an auxiliary macro would save us a few words of memory, but it would make their appearance even more obscure.

one two three four five

```
\hbox {\strut one two \hbox {three} four five}
```

```
19 98  \def\ruledhbox%
   99    {\normalhbox\bgroup
  100     \dowithnextbox{\makeruledbox\nextbox\box\nextbox\egroup}%
  101     \normalhbox}
```

first line
second line
third line
fourth line
fifth line

```
\vbox {\strut first line \par second line \par
third line \par fourth line \par fifth line \strut
}
```

```
20 102 \def\ruledvbox%
  103    {\normalvbox\bgroup
  104     \dowithnextbox{\makeruledbox\nextbox\box\nextbox\egroup}%
  105     \normalvbox}
```

first line
second line
third line
fourth line
fifth line

```
\vtop {\strut first line \par second line \par
third line \par fourth line \par fifth line \strut
}
```

```
21 106 \def\ruledvtop%
  107    {\normalvtop\bgroup
  108     \dowithnextbox{\makeruledbox\nextbox\box\nextbox\egroup}%
  109     \normalvtop}
```

Of the next two macros the first can be used to precede a box of ones own choice. One can for instance prefix boxes with `\ruledbox` and afterwards — when the macro satisfy the needs — let it to `\relax`.

```
\ruledbox\hbox{What rules do you mean?}
```

The macro `\setruledbox` can be used to directly rule a box.

```
\setruledbox12=\hbox{Who's talking about rules here?}
```

At the cost of some extra macros we can implement a variant that does not need the `=`, but we stick to:

```
22 110 \def\ruledbox%
  111    {\dowithnextbox{\makeruledbox\nextbox\box\nextbox}}

23 112 \def\setruledbox#1=%
  113    {\dowithnextbox{\makeruledbox\nextbox\setbox#1=\nextbox}}
```

Before we meet the visualizing macro's, we first implement ourselves some handy utility ones. Just for the sake of efficiency and readability, we introduce some status variables, that tell us a bit more about the registers we use:

```
\ifflexible
\ifzero
\ifnegative
\ifpositive
```

These status variables are set when we call for one of the investigation macros, e.g.

```
\investigateskip\scratchskip
```

We use some dirty trick to check stretchability of ⟨*skips*⟩. Users of these macros are invited to study their exact behavior first. The positive and negative states both include zero and are in fact non-negative (≥ 0) and non-positive (≤ 0) .

```
24 114 \newif\ifflexible
  115 \newif\ifzero
  116 \newif\ifnegative
  117 \newif\ifpositive

25 118 \def\investigateskip#1%
  119    {\relax
  120     \scratchdimen=#1\relax
  121     \edef\!!stringa{\the\scratchdimen}%
  122     \edef\!!stringb{\the#1}%
```

```
    \ifx\!!stringa\!!stringb \flexiblefalse \else \flexibletrue \fi
    \ifdim#1=\!!zeropoint\relax
      \zerotrue      \else
      \zerofalse     \fi
    \ifdim#1<\!!zeropoint\relax
      \positivefalse \else
      \positivetrue  \fi
    \ifdim#1>\!!zeropoint\relax
      \negativefalse \else
      \negativetrue  \fi}
\def\investigatecount#1%
  {\relax
   \flexiblefalse
   \ifnum#1=0
     \zerotrue      \else
     \zerofalse     \fi
   \ifnum#1<0
     \positivefalse \else
     \positivetrue  \fi
   \ifnum#1>0
     \negativefalse \else
     \negativetrue  \fi}
\def\investigatemuskip#1%
  {\relax
   \edef\!!stringa{\the\scratchmuskip}%
   \edef\!!stringb{0mu}%
   \def\!!stringc##1##2\\{##1}%
   \expandafter\edef\expandafter\!!stringc\expandafter
     {\expandafter\!!stringc\!!stringa\\}%
   \edef\!!stringd{-}%
   \flexiblefalse
   \ifx\!!stringa\!!stringb
     \zerotrue
     \negativefalse
     \positivefalse
   \else
     \zerofalse
     \ifx\!!stringc\!!stringd
       \positivefalse
       \negativetrue
     \else
       \positivetrue
       \negativefalse
     \fi
   \fi}
```

Indentation, left and/or right skips, redefinition of `\par` and assignments to `\everypar` can lead to unwanted results. We can therefore turn all those things off with `\dontinterfere`.

```
\def\dontinterfere%
  {\everypar  = {}%
   \let\par   = \endgraf
   \parindent = \!!zeropoint
   \parskip   = \!!zeropoint
   \leftskip  = \!!zeropoint
   \rightskip = \!!zeropoint
   \relax}
```

In this module we do a lot of box manipulations. Because we don't want to be confronted with to many over- and underfull messages we introduce `\dontcomplain`.

```
\def\dontcomplain%
  {\hbadness  = \!!tenthousand
   \hfuzz     = \maxdimen
   \vbadness  = \!!tenthousand
   \vfuzz     = \maxdimen}
```

Now the neccessary utility macros are defined, we can make a start with the visualizing ones. The implementation of these macros is a compromise between readability, efficiency of coding and processing speed. Sometimes we do in steps what could have been done in combination, sometimes we use a few boxes more or less then actually needed, and more than once one can find the same piece of rule drawing code twice.

Depending on the context, one can force visual vertical cues being centered along `\hsize` or being put at the current position. Although centering often looks better, we've chosen the second alternative as default. The main reason for doing so is that often when we don't set the `\hsize` ourselves, TEX takes the value of the surrounding box. As a result the visual cues can migrate outside the current context.

This behavior is accomplished by a small but effective auxiliary macro, which behavior can be influenced by the boolean `\centeredvcue`. By saying

```
\centeredvcuetrue
```

one turns centering on. As said, we turn it off.

```
\newif\ifcenteredvcue  \centeredvcuefalse
\def\normalvcue#1%
  {\normalhbox \ifcenteredvcue to \hsize \fi {\normalhss#1\normalhss}}
```

We could have used the more robust version

```
\def\normalvcue%
  {\normalhbox \ifcenteredvcue to \hsize \fi
   \bgroup\bgroup\normalhss
   \aftergroup\normalhss\aftergroup\egroup
   \let\next=}
```

or the probably best one:

```
\def\normalvcue%
  {\hbox \ifcenteredvcue to \hsize
     \bgroup\bgroup\normalhss
     \aftergroup\normalhss\aftergroup\egroup
   \else
     \bgroup
   \fi
   \let\next=}
```

Because we don't have to preserve ⟨*catcodes*⟩ and only use small arguments, we stick to the first alternative.

We build our visual cues out of rules. At the cost of a much bigger DVI file, this is to be prefered over using characters (1) because we cannot be sure of their availability and (2) because their dimensions are fixed.

As with ruled boxes, we use a ⟨*dimension*⟩ to specify the width of the ruled elements. This dimension defaults to:

```
\testrulewidth=\boxrulewidth
```

Because we prefer whole numbers for specifying the dimensions, we often use even multiples of `\testrulewidth`.

A second variable is introduced because of the stretch components of ⟨*skips*⟩. At the cost of some accuracy we can make this stretch visible.

```
\visiblestretchtrue
```

```
\newdimen\testrulewidth  \testrulewidth=\boxrulewidth
\newif\ifvisiblestretch  \visiblestretchfalse
```

We start with the easiest part, the fills. The scheme we follow is *visual filling – going back – normal filling.* Visualizing is implemented using `\cleaders`. Because the ⟨*box*⟩ that follows this command is constructed only once, the `\copy` is not really a prerequisite. We prefer using a `\normalhbox` here instead of a `\hbox`.

```
\def\setvisiblehfilbox#1\to#2#3#4%
  {\setbox#1=\normalhbox
     {\vrule
        \!!width#2\testrulewidth
        \!!height#3\testrulewidth
        \!!depth#4\testrulewidth}%
   \smashbox#1}
\def\doruledhfiller#1#2#3#4%
  {#1#2%
   \bgroup
     \dontinterfere
     \dontcomplain
     \setvisiblehfilbox0\to{4}{#3}{#4}%
     \setvisiblehfilbox2\to422%
     \copy0\copy2
     \bgroup
```

```
202         \setvisiblehfilbox0\to422%
203         \cleaders
204           \normalhbox to 12\testrulewidth
205             {\normalhss\copy0\normalhss}%
206           #1%
207       \egroup
208       \setbox0=\normalhbox
209         {\normalhskip-4\testrulewidth\copy0\copy2}%
210       \smashbox0
211       \box0
212     \egroup}
```

The horizontal fillers differ in their boundary visualization. Watch the small dots. Fillers can be combined within reasonable margins.

\hss test

\hfil test

\hfill test

\hfil\hfil test \hfil

The negative counterparts are visualizes, but seldom become visible, apart from their boundaries.

\hfilneg test

\hfillneg test

Although leaders are used for visualizing, they are visualized themselves correctly as the next example shows.

All five substitutions use the same auxiliary macro. Watch the positive first – negative next approach.

```
35 213 \def\ruledhss%
  214   {\doruledhfiller\normalhss\normalhfilneg{0}{0}}

36 215 \def\ruledhfil%
  216   {\doruledhfiller\normalhfil\normalhfilneg{10}{-6}}

37 217 \def\ruledhfill%
  218   {\doruledhfiller\normalhfill\normalhfillneg{18}{-14}}

38 219 \def\ruledhfilneg%
  220   {\doruledhfiller\normalhfilneg\normalhfil{-6}{10}}

39 221 \def\ruledhfillneg%
  222   {\doruledhfiller\normalhfillneg\normalhfill{-14}{18}}
```

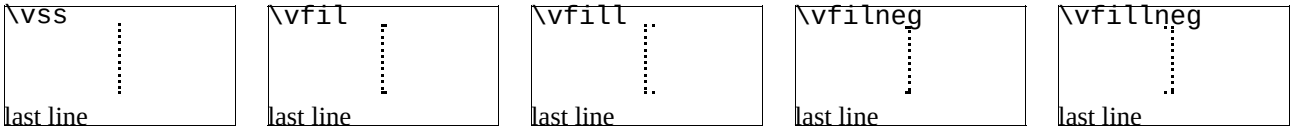
The vertical mode commands adopt the same visualization scheme, but are implemented in a slightly different way.

```
40 223 \def\setvisiblevfilbox#1\to#2#3#4%
  224   {\setbox#1=\normalvcue
  225      {\vrule
  226         \!!width#2\testrulewidth
  227         \!!height#3\testrulewidth
  228         \!!depth#4\testrulewidth}%
  229    \smashbox#1}%

41 230 \def\doruledvfiller#1#2#3%
  231   {#1#2%
  232    \bgroup
  233      \dontinterfere
  234      \dontcomplain
  235      \offinterlineskip
  236      \setvisiblevfilbox0\to422%
  237      \setbox2=\normalhbox
  238        {\normalhskip -#3\testrulewidth\copy0}%
  239      \smashbox2
  240      \copy2
  241      \bgroup
  242        \setbox2=\normalhbox
  243          {\normalhskip -2\testrulewidth\copy0}%
  244        \smashbox2
  245        \copy2
```

```
246         \cleaders
247           \normalvbox to 12\testrulewidth
248             {\normalvss\copy2\normalvss}%
249           #1%
250         \setbox2=\normalvbox
251           {\vskip-2\testrulewidth\copy2}%
252         \smashbox2
253         \box2
254       \egroup
255       \setbox2=\normalvbox
256         {\vskip-2\testrulewidth\copy2}%
257       \smashbox2
258       \box2
259     \egroup}
```

Because they act the same as their horizontal counterparts we only show a few examples.



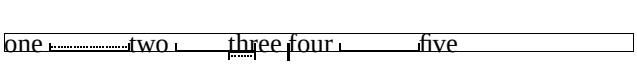
Keep in mind that `\vfillneg` is not part of PLAIN TEX, but are mimmicked by a macro.

```
42 260 \def\ruledvss%
  261   {\doruledvfiller\normalvss\normalvfilneg{2}}
43 262 \def\ruledvfil%
  263   {\doruledvfiller\normalvfil\normalvfilneg{-4}}
44 264 \def\ruledvfill%
  265   {\doruledvfiller\normalvfill\normalvfillneg{-12}}
45 266 \def\ruledvfilneg%
  267   {\doruledvfiller\normalvfilneg\normalvfil{8}}
46 268 \def\ruledvfillneg%
  269   {\doruledvfiller\normalvfillneg\normalvfill{16}}
```

Skips differ from kerns in two important aspects:

- line and pagebreaks are allowed at a skip
- skips can have a positive and/or negative stretchcomponent

Stated a bit different: kerns are fixed skips at which no line or pagebreak can occur. Because skips have a more open character, they are visualized in a open way.




```
one \hskip +30pt plus 5pt two \hskip +30pt \hskip
-10pt plus 5pt three \hskip 0pt four \hskip +30pt
five
```

When skips have a stretch component, this is visualized by means of a dashed line. Positive skips are on top of the baseline, negative ones are below it. This way we can show the combined results. An alternative visualization of stretch could be drawing the mid line over a length of the stretch, in positive or negative direction.

```
47 270 \def\doruledhskip%
  271   {\relax
  272    \dontinterfere
  273    \dontcomplain
  274    \investigateskip\scratchskip
  275    \ifzero
  276      \setbox0=\normalhbox
  277        {\normalhskip-\testrulewidth
  278         \vrule
  279           \!!width4\testrulewidth
  280           \!!height16\testrulewidth
  281           \!!depth16\testrulewidth}%
  282    \else
  283      \setbox0=\normalhbox to \ifnegative-\fi\scratchskip
  284        {\vrule
  285           \!!width2\testrulewidth
  286           \ifnegative\!!depth\else\!!height\fi16\testrulewidth
  287         \cleaders
  288           \hrule
```

```
289             \ifnegative
290               \!!depth2\testrulewidth
291               \!!height\!!zeropoint
292             \else
293               \!!height2\testrulewidth
294               \!!depth\!!zeropoint
295             \fi
296           \normalhfill
297         \ifflexible
298           \normalhskip\ifnegative\else-\fi\scratchskip
299           \normalhskip2\testrulewidth
300           \cleaders
301             \normalhbox
302               {\normalhskip 2\testrulewidth
303                \vrule
304                  \!!width2\testrulewidth
305                  \!!height\ifnegative-7\else9\fi\testrulewidth
306                  \!!depth\ifnegative9\else-7\fi\testrulewidth
307                \normalhskip 2\testrulewidth}%
308             \normalhfill
309         \fi
310         \vrule
311           \!!width2\testrulewidth
312           \ifnegative\!!depth\else\!!height\fi16\testrulewidth}%
313       \setbox0=\normalhbox
314         {\ifnegative\else\normalhskip-\scratchskip\fi
315          \box0}%
316     \fi
317     \smashbox0%
318     \ifvisiblestretch \else
319       \flexiblefalse
320     \fi
321     \ifflexible
322       % breaks ok but small displacements can occur
323       \skip2=\scratchskip
324       \advance\skip2 by -1\scratchskip
325       \divide\skip2 by 2
326       \advance\scratchskip by -\skip2
327       \normalhskip\scratchskip
328       \normalpenalty\!!tenthousand
329       \box0
330       \normalhskip\skip2
331     \else
332       \normalhskip\scratchskip
333       \box0
334     \fi
335     \egroup}
48 336 \def\ruledhskip%
337   {\bgroup
338    \afterassignment\doruledhskip
339    \scratchskip=}
```

The visual skip is located at a feasible point. Normally this does not interfere with the normaltypesetting process. The next examples show (1) the default behavior, (2) the (not entirely correct) distributed stretch and (3) the way the text is typeset without cues.

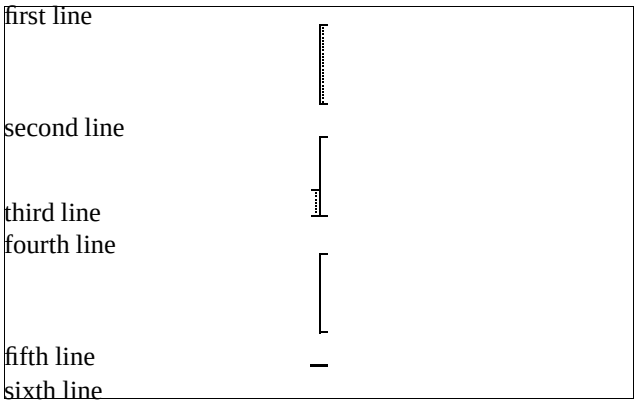
test test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test

test test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test

test test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test

We are less fortunate when implementing the vertical skips. This is a direct result of interference between the boxes that visualize the skip and skip removal at a pagebreak. Normally skips disappear at the top of a page, but not of course when visualized in a `\vbox`. A quite perfect simulation could have been built if we would have had available two more

primitives: \hnop and \vnop. These new primitives could stand for boxes that are visible but are not taken into account in any way. They are there for us, but not for TEX.




```
first line \vskip +30pt plus 5pt second line
\vskip +30pt \vskip -10pt plus 5pt third line
\par fourth line \vskip +30pt fifth line \vskip
0pt sixth line
```

We have to postpone \prevdepth. Although this precaution probably is not completely waterproof, it works quite well.

```
49 340 \def\dodoruledvskip%
   341   {\nextdepth=\prevdepth
   342    \dontinterfere
   343    \dontcomplain
   344    \offinterlineskip
   345    \investigateskip\scratchskip
   346    \ifzero
   347      \setbox0=\normalvcue
   348        {\vrule
   349           \!!width32\testrulewidth
   350           \!!height2\testrulewidth
   351           \!!depth2\testrulewidth}%
   352    \else
   353      \setbox0=\normalvbox to \ifnegative-\fi\scratchskip
   354        {\hrule
   355           \!!width16\testrulewidth
   356           \!!height2\testrulewidth
   357         \ifflexible
   358           \cleaders
   359             \normalhbox to 16\testrulewidth
   360               {\normalhss
   361                \normalvbox
   362                  {\normalvskip 2\testrulewidth
   363                   \hrule
   364                   \!!width2\testrulewidth
   365                   \!!height2\testrulewidth
   366                   \normalvskip 2\testrulewidth}%
   367                \normalhss}%
   368             \normalvfill
   369         \else
   370           \normalvfill
   371         \fi
   372         \hrule
   373           \!!width16\testrulewidth
   374           \!!height2\testrulewidth}%
   375      \setbox2=\normalvbox to \ht0
   376        {\hrule
   377           \!!width2\testrulewidth
   378           \!!height\ht0}%
   379      \ifnegative
   380        \ht0=\!!zeropoint
   381        \setbox0=\normalhbox
   382          {\normalhskip2\testrulewidth % will be improved
   383           \normalhskip-\wd0\box0}%
   384      \fi
   385      \smashbox0%
   386      \smashbox2%
   387      \setbox0=\normalvcue
   388        {\box2\box0}%
   389      \setbox0=\normalvbox
   390        {\ifnegative\normalvskip\scratchskip\fi\box0}%
```

```
391       \smashbox0%
392     \fi
393     \ifvisiblestretch
394       \ifflexible
395         \skip2=\scratchskip
396         \advance\skip2 by -1\scratchskip
397         \divide\skip2 by 2
398         \advance\scratchskip by -\skip2
399         \normalvskip\skip2
400       \fi
401     \fi
402     \normalpenalty\!!tenthousand
403     \box0
404     \prevdepth=\nextdepth % not \dp0=\nextdepth
405     \normalvskip\scratchskip}
```

We try to avoid interfering at the top of a page. Of course we only do so when we are in the main vertical list.

```
50 406 \def\doruledvskip%
407   {\par
408    \ifdim\pagegoal=\maxdimen
409      \ifinner
410        \dodoruledvskip
411      \fi
412    \else
413      \dodoruledvskip
414    \fi
415    \egroup}
51 416 \def\ruledvskip%
417   {\bgroup
418    \afterassignment\doruledvskip
419    \scratchskip=}
```

The macros that implement the kerns are a bit more complicated than needed, because they also serve the visualization of glue, our PLAIN defined kerns with stretch or shrink. We've implemented both horizontal and vertical kerns as ruled boxes.

one two three four five

```
one \kern +30pt two \kern +30pt \kern -10pt three
\kern 0pt four \kern +30pt five
```

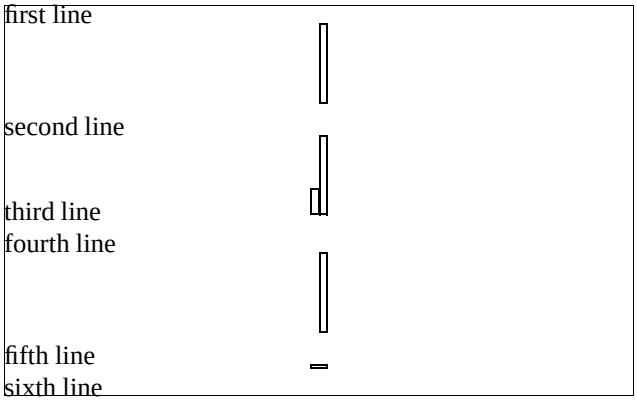
Positive and negative kerns are placed on top or below the baseline, so we are able to track their added result. We didn't mention spacings of 0 pt yet. Zero values are visualized a bit different, because we want to see them anyhow.

```
52 420 \def\doruledhkern%
421   {\dontinterfere
422    \dontcomplain
423    \baselinerulefalse
424    \investigateskip\scratchskip
425    \boxrulewidth=2\testrulewidth
426    \ifzero
427      \setbox0=\ruledhbox to 8\testrulewidth
428        {\vrule
429           \!!width\!!zeropoint
430           \!!height16\testrulewidth
431           \!!depth16\testrulewidth}%
432      \setbox0=\normalhbox
433        {\normalhskip-4\testrulewidth\box0}%
434    \else
435      \setbox0=\ruledhbox to \ifnegative-\fi\scratchskip
436        {\vrule
437           \!!width\!!zeropoint
438           \ifnegative\!!depth\else\!!height\fi16\testrulewidth
439         \ifflexible
440           \normalhskip2\testrulewidth
441           \cleaders
442             \normalhbox
443               {\normalhskip 2\testrulewidth
444                \vrule
445                  \!!width2\testrulewidth
446                  \!!height\ifnegative-7\else9\fi\testrulewidth
447                  \!!depth\ifnegative9\else-7\fi\testrulewidth
448                \normalhskip 2\testrulewidth}%
```

```
            \normalhfill
          \else
            \normalhfill
          \fi}%
      \testrulewidth=2\testrulewidth
      \setbox0=\ruledhbox{\box0}%  \make...
    \fi
    \smashbox0%
    \normalpenalty\!!tenthousand
    \normalhbox to \!!zeropoint
      {\ifnegative\normalhskip1\scratchskip\fi
       \box0}%
    \afterwards\scratchskip
    \egroup}

\def\ruledhkern#1%
  {\bgroup
   \let\afterwards=#1\relax
   \afterassignment\doruledhkern
   \scratchskip=}
```

After having seen the horizontal ones, the vertical kerns will not surprise us. In this example we use `\par` to switch to vertical mode.




```
first line \par \kern +30pt second line \par
\kern +30pt \par \kern -10pt third line \par
fourth line \par \kern +30pt fifth line \par
\kern 0pt sixth line
```

Like before, we have to postpone `\prevdepth`. If we leave out this trick, we got ourselves some wrong spacing.

```
\def\dodoruledvkern%
  {\nextdepth=\prevdepth
   \dontinterfere
   \dontcomplain
   \baselinerulefalse
   \offinterlineskip
   \investigateskip\scratchskip
   \boxrulewidth=2\testrulewidth
   \ifzero
     \setbox0=\ruledhbox to 32\testrulewidth
       {\vrule
          \!!width\!!zeropoint
          \!!height4\testrulewidth
          \!!depth4\testrulewidth}%
   \else
     \setbox0=\ruledvbox to \ifnegative-\fi\scratchskip
       {\hsize16\testrulewidth
        \ifflexible
          \cleaders
            \normalhbox to 16\testrulewidth
              {\normalhss
               \normalvbox
                 {\normalvskip 2\testrulewidth
                  \hrule
                    \!!width2\testrulewidth
                    \!!height2\testrulewidth
                  \normalvskip 2\testrulewidth}%
               \normalhss}%
            \normalvfill
        \else
          \vrule
```

```
            \!!width\!!zeropoint
            \!!height\ifnegative-\fi\scratchskip
          \normalhfill
        \fi}
  \fi
  \testrulewidth=2\testrulewidth
  \setbox0=\ruledvbox{\box0}%  \make...
  \smashbox0%
  \setbox0=\normalvbox
    {\ifnegative\normalvskip\scratchskip\fi
     \normalvcue
      {\ifnegative\normalhskip-16\testrulewidth\fi\box0}}%
  \smashbox0%
  \normalpenalty\!!tenthousand
  \box0
  \prevdepth=\nextdepth} % not \dp0=\nextdepth


\def\doruledvkern%
  {\ifdim\pagegoal=\maxdimen
     \ifinner
       \dodoruledvkern
     \fi
  \else
     \dodoruledvkern
  \fi
  \afterwards\scratchskip
  \egroup}


\def\ruledvkern#1%
  {\bgroup
   \let\afterwards=#1\relax
   \afterassignment\doruledvkern
   \scratchskip=}


\def\ruledkern%
  {\ifvmode
     \let\next=\ruledvkern
  \else
     \let\next=\ruledhkern
  \fi
  \next\normalkern}
```

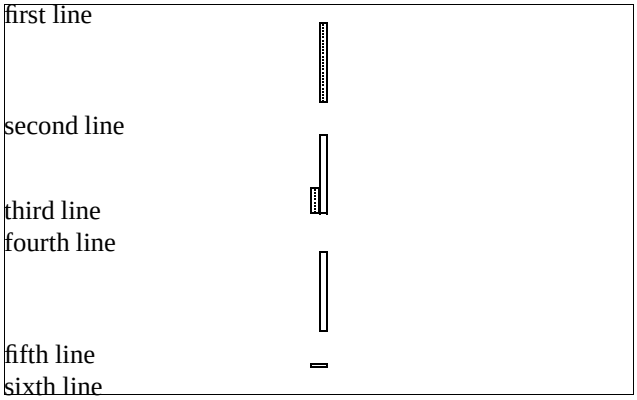
The non-primitive glue commands are treated as kerns with stretch. This stretch is presented as a dashed line. I have to admit that until now, I've never used these glue commands.

one two three four five

```
one \hglue +30pt plus 5pt two \hglue +30pt \hglue
-10pt plus 5pt three \hglue 0pt four \hglue +30pt
five
```

```
\def\doruledhglue%
  {\leavevmode
   \scratchcounter=\spacefactor
   \vrule\!!width\!!zeropoint
   \normalpenalty\!!tenthousand
   \ruledhkern\normalhskip\scratchskip
   \spacefactor=\scratchcounter
   \egroup}


\def\ruledhglue%
  {\bgroup
   \afterassignment\doruledhglue\scratchskip=}
```




```
first line \vglue +30pt plus 5pt second line
\vglue +30pt \vglue -10pt plus 5pt third line
\par fourth line \vglue +30pt fifth line \vglue
0pt sixth line
```

```
\def\doruledvglue%
  {\par
   \nextdepth=\prevdepth
   \hrule\!!height\!!zeropoint
   \normalpenalty\!!tenthousand
   \ruledvkern\normalvskip\scratchskip
   \prevdepth=\nextdepth
   \egroup}
\def\ruledvglue%
  {\bgroup
   \afterassignment\doruledvglue\scratchskip=}
```

Mathematical kerns and skips are specified in mu. This font related unit is incompatible with those of ⟨*dimensions*⟩ and ⟨*skips*⟩. Because in math mode spacing is often a very subtle matter, we've used a very simple, not overloaded way to show them.

```
\def\dodoruledmkern#1%
  {\dontinterfere
   \dontcomplain
   \setbox0=\normalhbox
     {$\normalmkern\ifnegative-\fi\scratchmuskip$}%
   \setbox0=\normalhbox to \wd0
     {\vrule
        \!!height16\testrulewidth
        \!!depth16\testrulewidth
        \!!width\testrulewidth
      \leaders
        \hrule
          \!!height\ifpositive16\else-14\fi\testrulewidth
          \!!depth\ifpositive-14\else16\fi\testrulewidth
        \normalhfill
      \ifflexible
        \normalhskip-\wd0
        \leaders
          \hrule
            \!!height\testrulewidth
            \!!depth\testrulewidth
          \normalhfill
      \fi
      \vrule
        \!!height16\testrulewidth
        \!!depth16\testrulewidth
        \!!width\testrulewidth}%
   \smashbox0%
   \ifnegative
      #1\scratchmuskip
     \box0
   \else
     \box0
      #1\scratchmuskip
   \fi
   \egroup}
```

$a = b + c$

```
$a \mkern 3mu = \mkern 3mu b \quad \mkern -2mu
+ \mkern -2mu \quad c$
```

```
\def\doruledmkern%
```

```
  {\investigatemuskip\scratchmuskip
   \flexiblefalse
   \dodoruledmkern\normalmkern}

\def\ruledmkern%
  {\bgroup
   \afterassignment\doruledmkern\scratchmuskip=}
```

$a \mskip 3mu = \mskip 3mu b \quad \mskip -2mu + \mskip -2mu \quad c$

```
\def\doruledmskip%
  {\investigatemuskip\scratchmuskip
   \flexibletrue
   \dodoruledmkern\normalmskip}

\def\ruledmskip%
  {\bgroup
   \afterassignment\doruledmskip\scratchmuskip=}
```

After presenting fills, skip, kerns and glue we've come to see penalties. In the first implementation — most of the time needed to develop this set of macros went into testing different types of visualization — penalties were mere small blocks with one black half, depending on the sign. This most recent version also gives an indication of the amount of penalty. Penalties can go from less than −10000 to over +10000, and their behavior is somewhat non-lineair, with some values having special meanings. We therefore decided not to use its value for a lineair indicator.

one two three four five

```
one \penalty +100 two \penalty +100 \penalty
-100 three \penalty 0 four \penalty +100 five
```

The small sticks at the side of the penalty indicate it size. The next example shows the positive and negative penalties of 0, 1, 10, 100, 1000 and 10000.

test test test test test test test

test test test test test test test

This way stacked penalties of different severance can be shown in combination.

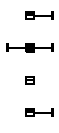
test test test test

```
\def\setruledpenaltybox#1#2#3#4#5#6%
  {\setbox#1=\normalhbox
     {\ifnum#2=0 \else
        \ifnum#2>0
          \def\sign{+}%
        \else
          \def\sign{-}%
        \fi
        \dimen0=\ifnum\sign#2>9999
                  28\else
                \ifnum\sign#2>999
                  22\else
                \ifnum\sign#2>99
                  16\else
                \ifnum\sign#2>9
                  10\else
                  4
                \fi\fi\fi\fi \testrulewidth
        \ifnum#2<0
          \normalhskip-\dimen0
          \normalhskip-2\testrulewidth
          \vrule
            \!!width2\testrulewidth
            \!!height#3\testrulewidth
            \!!depth#4\testrulewidth
        \fi
        \vrule
          \!!width\dimen0
          \!!height#5\testrulewidth
          \!!depth#6\testrulewidth
        \ifnum#2>0
          \vrule
            \!!width2\testrulewidth
            \!!height#3\testrulewidth
```

```
643             \!!depth#4\testrulewidth
644           \fi
645       \fi}%
646     \smashbox#1}
68 647 \def\doruledhpenalty%
648   {\dontinterfere
649    \dontcomplain
650    \investigatecount\scratchcounter
651    \testrulewidth=2\testrulewidth
652    \boxrulewidth=\testrulewidth
653    \setbox0=\ruledhbox to 8\testrulewidth
654      {\ifnegative\else\normalhss\fi
655       \vrule
656         \!!depth8\testrulewidth
657         \!!width\ifzero0\else4\fi\testrulewidth
658       \ifpositive\else\normalhss\fi}%
659    \setruledpenaltybox{2}{\scratchcounter}{0}{8}{-3.5}{4.5}%
660    \normalpenalty\!!tenthousand
661    \setbox0=\normalhbox
662      {\normalhskip-4\testrulewidth
663       \ifnegative
664         \box2\box0
665       \else
666         \box0\box2
667       \fi}%
668    \smashbox0%
669    \box0
670    \normalpenalty\scratchcounter
671    \egroup}
69 672 \def\ruledhpenalty%
673   {\bgroup
674    \afterassignment\doruledhpenalty
675    \scratchcounter=}
```

The size of a vertical penalty is also shown on the horizontal axis. This way there is less interference with the often preceding or following skips and kerns.

first line
second line
third line
fourth line
fifth line



```
first line \par \penalty +100 second line \par
\penalty +100 \par \penalty -100 third line \par
\penalty 0 fourth line \par \penalty +100 fifth
line
```

```
70 676 \def\doruledvpenalty%
677   {\ifdim\pagegoal=\maxdimen
678    \else
679      \nextdepth=\prevdepth
680      \dontinterfere
681      \dontcomplain
682      \investigatecount\scratchcounter
683      \testrulewidth=2\testrulewidth
684      \boxrulewidth=\testrulewidth
685      \setbox0=\ruledhbox
686        {\vrule
687           \!!height4\testrulewidth
688           \!!depth4\testrulewidth
689           \!!width\!!zeropoint
690         \vrule
691           \!!height\ifnegative.5\else4\fi\testrulewidth
692           \!!depth\ifpositive.5\else4\fi\testrulewidth
693           \!!width8\testrulewidth}%
694      \setruledpenaltybox{2}{\scratchcounter}{4}{4}{.5}{.5}%
695      \setbox0=\normalhbox
696        {\normalhskip-4\testrulewidth
697         \ifnegative
698           \box2\box0
699         \else
700           \box0\box2
701         \fi
702         \normalhss}%
```

```
      \smashbox0%
      \normalpenalty\!!tenthousand
      \nointerlineskip
      \dp0=\nextdepth  % not \prevdepth=\nextdepth
      \normalvbox
        {\normalvcue{\box0}}%
    \fi
    \normalpenalty\scratchcounter
    \egroup}

\def\ruledvpenalty%
  {\bgroup
   \afterassignment\doruledvpenalty
   \scratchcounter=}

\def\ruledpenalty%
  {\ifvmode
     \let\next=\ruledvpenalty
   \else
     \let\next=\ruledhpenalty
   \fi
   \next}
```

For those who want to manipulate the visual cues in detail, we have grouped them.

```
\def\showfils%
  {\let\hss      = \ruledhss
   \let\hfil     = \ruledhfil
   \let\hfill    = \ruledhfill
   \let\hfilneg  = \ruledhfilneg
   \let\hfillneg = \ruledhfillneg
   \let\vss      = \ruledvss
   \let\vfil     = \ruledvfil
   \let\vfill    = \ruledvfill
   \let\vfilneg  = \ruledvfilneg
   \let\vfillneg = \ruledvfillneg}

\def\dontshowfils%
  {\let\hss      = \normalhss
   \let\hfil     = \normalhfil
   \let\hfill    = \normalhfill
   \let\hfilneg  = \normalhfilneg
   \let\hfillneg = \normalhfillneg
   \let\vss      = \normalvss
   \let\vfil     = \normalvfil
   \let\vfill    = \normalvfill
   \let\vfilneg  = \normalvfilneg
   \let\vfillneg = \normalvfillneg}

\def\showboxes%
  {\baselineruletrue
   \let\hbox     = \ruledhbox
   \let\vbox     = \ruledvbox
   \let\vtop     = \ruledvtop}

\def\dontshowboxes%
  {\let\hbox     = \normalhbox
   \let\vbox     = \normalvbox
   \let\vtop     = \normalvtop}

\def\showskips%
  {\let\hskip    = \ruledhskip
   \let\vskip    = \ruledvskip
   \let\kern     = \ruledkern
   \let\mskip    = \ruledmskip
   \let\mkern    = \ruledmkern
   \let\hglue    = \ruledhglue
   \let\vglue    = \ruledvglue}

\def\dontshowskips%
  {\let\hskip    = \normalhskip
   \let\vskip    = \normalvskip
   \let\kern     = \normalkern
   \let\mskip    = \normalmskip
```

```
    \let\mkern    = \normalmkern
    \let\hglue    = \normalhglue
    \let\vglue    = \normalvglue}

\def\showpenalties%
  {\let\penalty  = \ruledpenalty}

\def\dontshowpenalties%
  {\let\penalty  = \normalpenalty}
```

All these nice options come together in two macros. The first one turns the options on, the second turnes them off. Both macros only do their job when we are actually showing the composition.

```
\showingcompositiontrue
\showcomposition
```

Because the output routine can do tricky things, like multiple column typesetting and manipulation of the pagebody, shifting things around and so on, the macro `\dontshowcomposition` best can be called when we enter this routine. Too much visual cues just don't make sense. In CONTEXT this has been taken care of.

```
\newif\ifshowingcomposition

\def\showcomposition%
  {\ifshowingcomposition
     \showfils
     \showboxes
     \showskips
     \showpenalties
   \fi}

\def\dontshowcomposition%
  {\ifshowingcomposition
     \dontshowfils
     \dontshowboxes
     \dontshowskips
     \dontshowpenalties
   \fi}
```

Just to make things even more easy, we have defined:

```
\showmakeup
```

For the sake of those who don't (yet) use CONTEXT we preset `\defaulttestrulewidth` to the already set value. Otherwise we default to a corps related value.

```
\def\defaulttestrulewidth{.2pt}
```

Beware, it's a macro not a ⟨*dimension*⟩.

```
\ifx\korpsgrootte\undefined
  \edef\defaulttestrulewidth{\the\testrulewidth}
\else
  \def\defaulttestrulewidth{.02\korpsgrootte}  % still dutch
\fi

\def\showmakeup%
  {\testrulewidth=\defaulttestrulewidth
   \showingcompositiontrue
   \showcomposition}

\protect
```

The documented source you have been reading was processed using some surrogate makeup. When this file is processed in CONTEXT, a few more examples show up here, like a local table of contents and a local register.