This Way

ConTEXt magazine #7 MKIV
August 2004

Faking Text and More

Hans Hagen
PRAGMA ADE

Remark

When again a user asked me for the macros that I use to generate fake text, I took a while to document them. Most macros use the built in random number generator. In manuals you may want to control the randomization a bit. You can do that by setting the seed:

```
\setupystem[random=12345]
```

In MkIV there is a lot of visualization available like showing all boxes, glue, characters etc. (try `\showmakeup`) . Many mechanism have dedicated trackers that visualize matters with color. Here we just mention a few possibilities of a module with helpers. This module is loaded with:

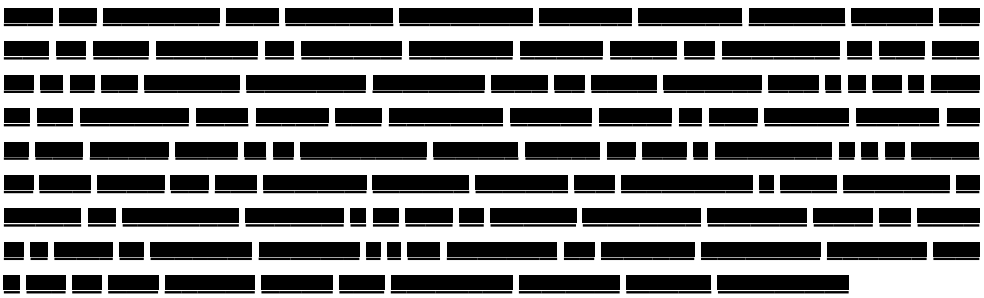
```
\usemodule[visual]
```

Faking words

We don't need much words to demonstrate the macros. Here we fake a single work with `\fakeword`: [redacted]. You can fake a whole bunch with:

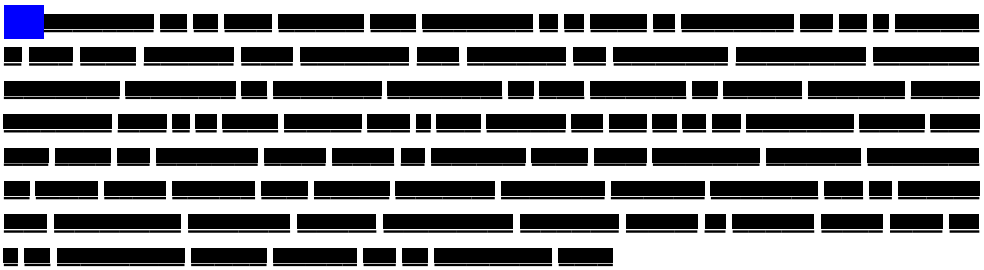
```
\fakewords{100}{200} \par
\fakewords {30} {80} \par
\fakewords{200}{200}
```

[redacted]



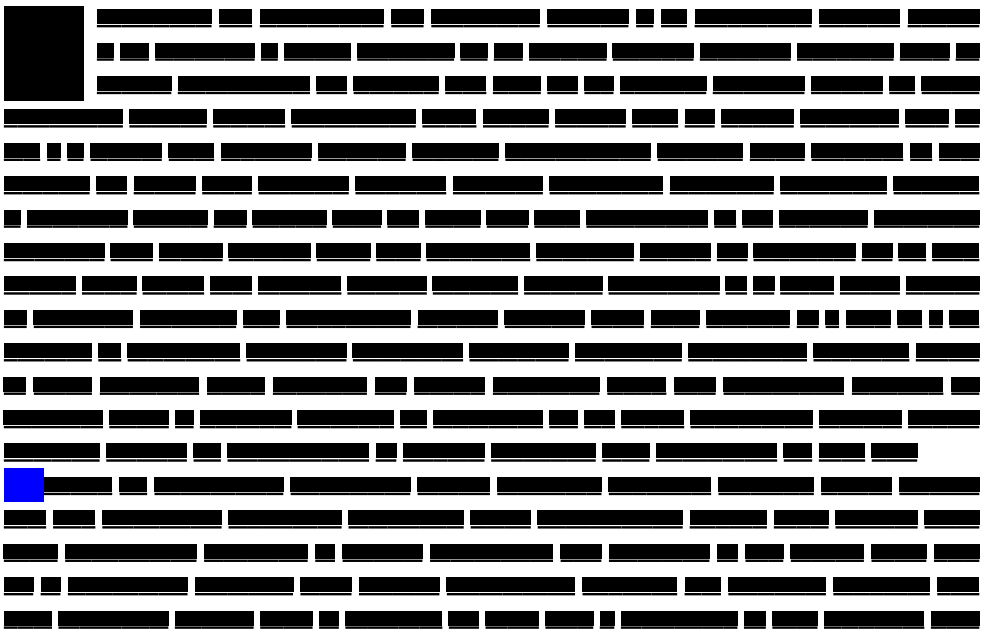
In addition to `\fakewords` we have `\fakenwords`. This time we don't specify a range, but a number and a random seed.

```
\fakenwords{100}{2} % words seed
```



Drop caps can be faked as follows:

```
\fakedroppedcaps{3}
\fakewords{100}{200} \par
\fakewords{100}{200}
```



You can visualize the indentation by adding another faker:

```
\fakeparindent \fakewords{100}{200}
```

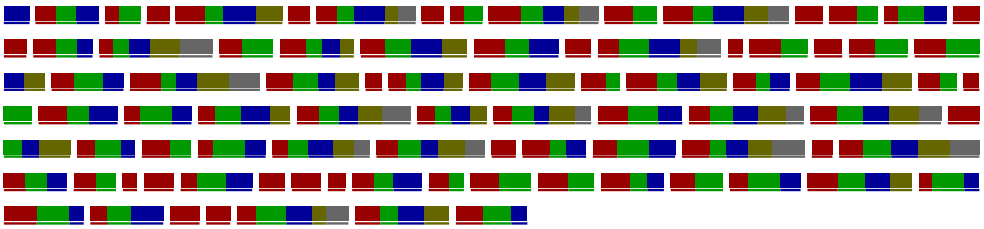
You can suppress indentation with:

```
\onlyfakewords{100}{200}
```

You can influence the color by redefining one or more of the folowing fake colors:

```
\definecolor[fakerulecolor]      [black]
\definecolor[fakebaselinecolor]  [green]
\definecolor[fakeparindentcolor][blue]
```

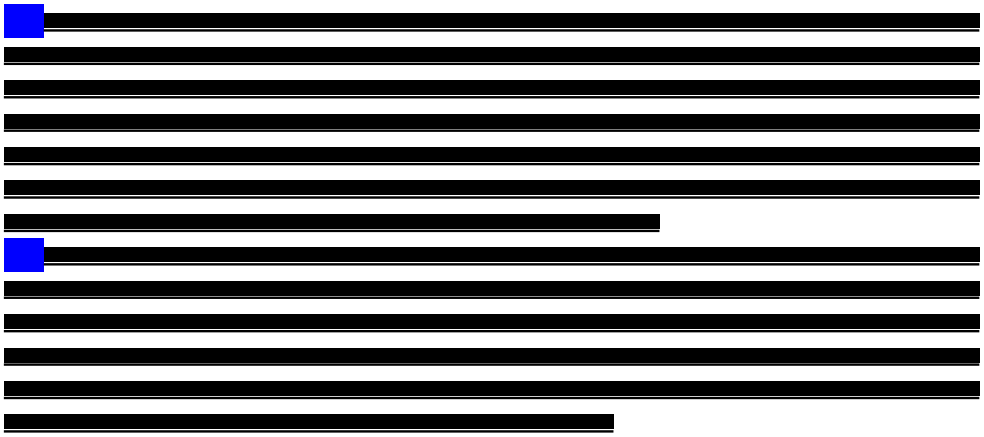
In case you wonder if fake words hyphenate, they kind of do, as is shown here:



Faking lines

Lines can be faked with:

```
\fakelines{3}{5}
\fakelines{4}{8}
```



This is (of course) more efficient than faking words.

Faking figures

Faking figures does not make that much sense.

```
\fakefigure
  [left][]
  {10em}{12em}
  {3\lineheight}{5\lineheight}

\fakewords{100}{200}
```

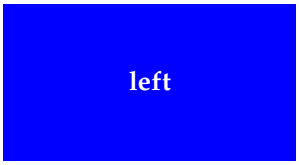


Figure 1

In this case the width will vary between `10em` and `12em`, while the height end up somewhere between 3 and 5 times the lineheight.

If you want nice placeholders you can better use the MetaPost `dum` library. This one hooks into the external figure placement macros and will produce a random graphic (with more or less random colors).

```
\useMPlibrary[dum]
\placefigure
  [left][]
  {\fakewords{3}{6}}
  {\externalfigure[ForTheMomentFaked][width=3cm,height=2cm]}

\fakewords{100}{200}
```

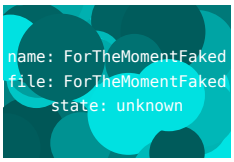


Figure 2

Faking formulas

Another probably seldom used placeholder is `\fakeformula`:

```
\startformula \fakeformula \stopformula
```

$$\blacksquare + \blacksquare + \blacksquare + \blacksquare + \blacksquare = \blacksquare$$

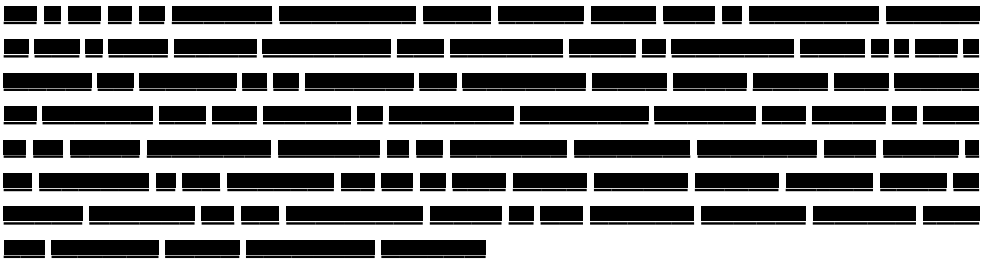
An alternative, showing baselines, is:

```
\startformula \fakespacingformula \stopformula
```

$$\blacksquare + \blacksquare + \blacksquare = \blacksquare$$

You can trigger drawing of baseline yourself too:

```
\showbaselines
\fakewords{100}{200} \par
\fakewords {30} {80} \par
\fakewords{200}{200}
```



In this case you will notice that this document is not typeset on a grid, and therefore, since the blank space is set to big the baseline visualization shows this distance when applicable.

source code of this document

```
% language=uk

% author    : Hans Hagen
% copyright : PRAGMA ADE & ConTeXt Development Team
% license   : Creative Commons Attribution ShareAlike 4.0 International
% reference : pragma-ade.nl | contextgarden.net | texlive (related) distributions
% origin    : the ConTeXt distribution
%
% comment   : Because this manual is distributed with TeX distributions it comes with a rather
%             liberal license. We try to adapt these documents to upgrades in the (sub)systems
%             that they describe. Using parts of the content otherwise can therefore conflict
%             with existing functionality and we cannot be held responsible for that. Many of
%             the manuals contain characteristic graphics and personal notes or examples that
%             make no sense when used out-of-context.

\usemodule[mag-01,abr-02,visual]

\startbuffer[abstract]
    The \type {m-visual} module is used in some manuals that come with \CONTEXT\
    to generate random text. This is sometimes less confusing that nice quotes
    because the reader can then distinguish the explanation from the example.
    This module is not extensive (but may grow) and is just an addition to
    already built in visualization tools.
\stopbuffer

\startdocument
  [title={Faking Text and More},
   author=Hans Hagen,
   affiliation=PRAGMA ADE,
   date=August 2004,
   number=7 \MKIV]

\setupindenting[medium] \indenting[always] \setupwhitespace[none]

\subject{Remark}

When again a user asked me for the macros that I use to generate fake text, I
took a while to document them. Most macros use the built in random number
generator. In manuals you may want to control the randomization a bit. You can do
that by setting the seed:

\starttyping
\setupystem[random=12345]
\stoptyping

% Some more visualization tricks are discussed in the visual debugger modules \type
% {supp-vis.tex} and \type {core-vis.tex}. If you have special wishes, let me know.
% If they make sense (or more important: if they can be implemented in a decent
% way) they may be honored in the future.

In \MKIV\ there is a lot of visualization available like showing all boxes, glue,
characters etc.\ (try \type {\showmakeup}) . Many mechanism have dedicated
trackers that visualize matters with color. Here we just mention a few
possibilities of a module with helpers. This module is loaded with:

\starttyping
\usemodule[visual]
\stoptyping

\subject{Faking words}
```

source code of this document

```
We don't need much words to demonstrate the macros. Here we fake a single work
with \type {\fakeword}: \fakeword. You can fake a whole bunch with:

\startbuffer
\fakewords{100}{200} \par
\fakewords {30} {80} \par
\fakewords{200}{200}
\stopbuffer

\typebuffer \getbuffer

In addition to \type {\fakewords} we have \type {\fakenwords}. This time we don't
specify a range, but a number and a random seed.

\startbuffer
\fakenwords{100}{2} % words seed
\stopbuffer

\typebuffer \getbuffer

Drop caps can be faked as follows:

\startbuffer
\fakedroppedcaps{3}
\fakewords{100}{200} \par
\fakewords{100}{200}
\stopbuffer

\typebuffer \getbuffer

You can visualize the indentation by adding another faker:

\startbuffer
\fakeparindent \fakewords{100}{200}
\stopbuffer

\typebuffer \getbuffer

You can suppress indentation with:

\startbuffer
\onlyfakewords{100}{200}
\stopbuffer

\typebuffer \getbuffer

You can influence the color by redefining one or more of the folowing fake
colors:

\startbuffer
\definecolor[fakerulecolor]     [black]
\definecolor[fakebaselinecolor] [green]
\definecolor[fakeparindentcolor][blue]
\stopbuffer

\typebuffer \getbuffer

In case you wonder if fake words hyphenate, they kind of do, as is shown here:
\bgroup \showfakewords \onlyfakewords{100}{200} \egroup

\subject{Faking lines}

Lines can be faked with:
```

source code of this document

```
\startbuffer
\fakelines{3}{5}
\fakelines{4}{8}
\stopbuffer

\typebuffer \getbuffer

This is (of course) more efficient than faking words.

\subject{Faking figures}

Faking figures does not make that much sense.

\startbuffer
\fakefigure
  [left][]
  {10em}{12em}
  {3\lineheight}{5\lineheight}

\fakewords{100}{200}
\stopbuffer

\typebuffer \getbuffer

In this case the width will vary between \type {10em} and \type {12em}, while the
height end up somewhere between 3 and~5 times the lineheight.

If you want nice placeholders you can better use the \METAPOST\ \type {dum}
library. This one hooks into the external figure placement macros and will
produce a random graphic (with more or less random colors).

\startbuffer
\useMPlibrary[dum]
\placefigure
  [left][]
  {\fakewords{3}{6}}
  {\externalfigure[ForTheMomentFaked][width=3cm,height=2cm]}

\fakewords{100}{200}
\stopbuffer

\typebuffer \getbuffer

\subject{Faking formulas}

Another probably seldom used placeholder is \type {\fakeformula}:

\startbuffer
\startformula \fakeformula \stopformula
\stopbuffer

\typebuffer \getbuffer

An alternative, showing baselines, is:

\startbuffer
\startformula \fakespacingformula \stopformula
\stopbuffer

\typebuffer \getbuffer

You can trigger drawing of baseline yourself too:
```

source code of this document

```
\startbuffer
\showbaselines
\fakewords{100}{200} \par
\fakewords {30} {80} \par
\fakewords{200}{200}
\stopbuffer

\typebuffer \bgroup \getbuffer \egroup

In this case you will notice that this document is not typeset on a grid, and
therefore, since the blank space is set to big the baseline visualization shows
this distance when applicable.

\stopdocument
```

source code of this document