IT'S IN THE DETAILS

HANS HAGEN
PRAGMA ADE
HASSELT NL

Introduction

On the ConTEXt mailing list, occasionally a user asks if we can post a complete document with the associated style. One reason for not honouring this request is that we want users to cook up their own styles. Besides that, there are a couple of styles in the regular ConTEXt distribution.

When browsing through this document, a ConTEXt user may wonder what style was used to achieve its look and feel. We hope that while reading the text and playing with the examples, the reader will accomplish the skills to define more than just simple layouts.

This document is not easy reading. Occasionally we spend some time explaining features not described in the manuals. The design of this document is to a large extent determined by its purpose, and as a result not always functional. Also the order of presenting features, tips and tricks is kind of random and unstructured. The idea is that the visual effects will draw you to the right trick. Also, if you realy want to benefit from these features, there is no way but to read the whole story.

In spite of all its shortcomings, I hope that you enjoy reading this (yet unfinished) manual. Keep in mind that this manual is far from finished.

Hans Hagen
Hasselt NL
Spring 2002

Table of contents

Although TEX is pretty good in applying glue to the paragraphs and pages, style designers often prefer typesetting on a grid. One reason for this is that when you print on not too thick paper, line will not shine through in annoying ways.

As soon as some typeset component has a height and/or depth larger than those of a single line, we need to compensate for the difference. For instance section headers are typeset in a font larger than the one used for typesetting the body text. This is where the grid snapper comes into action. Consider the following definition:

```
\definehead[DemoSection][section]
\setuphead [DemoSection][before=\blank,after=\blank]
```

To save some space, we don't put too much white space before and after a head. Because we are in grid snapping mode, the blank skips will equal the line height.

```
\setuphead[DemoSection][style=\bfb]
```

We set the style to a font where the sum of the height and depth exceeds the normal lineheight.

1.1 Snapping around (default).

Here we don't tune the snapper, we just apply the default handling. This means that the bottom line of the title will be aligned to the grid. The height is automatically set to a value that matches the grid.

By nature, TEX is not a strong grid thinker. There are several reason for this:

- Display math seldom fits to grid lines.
- Graphics are not designed with grids in mind.
- TEX thinks in paragraphs and pages and not in lines.
- Not standard text, like math, not always fits on grids.

Typesetting on a grid makes most sense when you have a relatively simple document. As soon as parts of the text have unexpected dimensions, for example due to backgrounds with appropriate spacing at the top and bottom, following a grid scheme quickly becomes messy.

So, before you decide to typeset on a grid, first ask yourself the question if it's worth the trouble. In a single column text with lots of whitespace and different typographical elements, grid based typesetting can turn out rather ugly. Many designers get grids forced up by their desk top applications, but in ConTEXt we do have a choice and can get consistent spacing without grids too.

In the next example sections we will explore some ways to influence placement on the grid. We will focus on section headers.

1.2 Snapping around (top).

The dimensions of a header determine the amount of space that is taken. Normally the default placement is ok, but in case you want extra space at the top, you can force that.

```
\setuphead[DemoSection][style=\bfb,grid=top]
```

When top is specified, one extra line is added at the top of the section title, unless we are at the top of a page. The next example also has an extra line, but here the space is distributed between top and bottom.

1.3 Snapping around (both).

Instead of top or both, you can also give bottom. As with top, the section title is placed on the grid line.

1.4 Snapping around (bottom).

So far we have just added extra space to the box to be placed at the grid. Because section headers are typeset larger than the body text, they already get more white space at the top. Therefore we want a more control over exact placement.

There is a second series of switches that gives you more control over the placement, but before we demonstrate the most comfortable one, we will first stepwise introduce the ingredients for moving text on the grid.

1.5 Snapping around (broad,high).

This section head was typeset under the following grid regime:

```
\setuphead[DemoSection][style=\bfb,grid={broad,high}]
```

These two keywords can be preceded by a - (minus sign) to get the opposite effect.

Instead of one keyword, we now provide two. The broad key tells ConTEXt that we're going to manipulate the internals of a grid box. Instead of broad, we can say fit. The difference between both methods is in rounding the number of grid lines that are taken by the grid snapping routines. Here we will only explore the method that takes the real dimensions into account (broad).

The high keyword moves the content up into the reserved space. As you may expect, we can also place the content in the middle.

1.6 Snapping around (broad,middle).

The third placement option is low and we will take this option for further exploration. As you can see, the depth of the content determines the distance between the bottom of the surrounding box and the content box.

1.7 Snapping around (broad,low).

The three options high, middle and low don't yet provide us a way to really manipulate the placement. This is because the depth related to a font is kind of unknown. So, let's get rid of the depth now:

1.8 Snapping around (broad,high,depth).

Instead of two keywords, we've now provided three:

```
\setuphead[DemoSection][style=\bfb,grid={broad,high,depth}]
```

As you can see, the baseline of the section title is now placed on the grid line. Because this is a comfortable starting point, ConTEXt also provides a shortcut

```
\setuphead[DemoSection][style=\bfb,grid=line]
```

1.9 Snapping around (line).

We can now take this reference placement as a starting point for more subtle up and down movements. Next we move up the line by 3pt.

```
\setuphead[DemoSection][style=\bfb,grid={line,-3pt}]
```

Keep in mind that since TEX works top–down, moving up means that we have to supply a negative dimension.

1.10 Snapping around (line,-3pt).

The displacement does not alter the dimensions of the box containing the section header. Therefore, this grid option combines well with the before and after keys.

Occasionally you may want to compensate for the difference in the so called 'topskip' and strutheight. This can be accomplished by the option page, which is meant for this kind of 'top of page' placement.

standard	normal, uncorrected placement on the line
top	add/remove an extra line to the top
both	add/remove half a line to the top and at the bottom
bottom	add/remove an extra line to the top
broad	move the content around
fit	move the content around in a tight box
high	align content to the top
middle	center the content
low	align content to the bottom
depth	ignore the depth of the content
page	apply top skip correction
line	shortcut for `broad,low,depth`
dimension	move content up or down by this amount

The options `high`, `middle`, and `low` can also be used without `broad` and `fit`.

1.11 Snapping around (high).

The effect of this manipulation is less controlled than its previously discussed usage.

1.12 Snapping around (middle).

A complication in moving content around (as with this `middle`) directive, is that we loose track of what we do. Also, subtle influences of line height and depth play a role.

1.13 Snapping around (low).

In order to give you an impression on what it is doing, you can turn on a couple of tracing options:

`\showstruts`	shows the imaginary characters that are added to force height and depth of lines.
`\showgrid`	puts the grid of lines in the background.
`\showgridboxes`	visualizes the content and encapsulating boxes.

When the grid is shown, the snapped components are accompanied by some marginal information. Because we use this for tracing purposes, the format may change over time. The dimensions shown reflect the corrections, pre– and post–snap skips, placement in the encapsulating box (`+` and `-` represent fillers), additional spacing (`00`, `01`, etc.) and a number identifying the snap point.

In desk top publishing applications the grid is pretty dominant in defininig layouts. On the other hand, TEX is pretty good defining layouts in terms of relative dimensions. This means that mapping a desk top publishing layout into its TEX (or ConTEXt) counterpart takes some effort.

We not only have to deal with vertical grids, but also with horizontal ones. Here we focus on the second category. When implementing designs, it is best first to look into the normal page layout areas. For most documents these are sufficient, but occasionally we need a more detailed approach.

When playing with grids, you need to make sure that grid snapping is turned on. It helps if you turn on the grid so that you can see where things end up. When a horizontal grid is defined, gray vertical rules show their boundaries.

```
\setuplayout[grid=yes] \showgrid
```

The \setuplayout command has a few settings that have to do with so called pseudo columns. These are in no sense related to multi column typesetting and only play a role in placing text on specific locations.

```
\setuplayout
  [columndistance=12pt,
   columns=3]
```

You can use \layoutcolumnoffset for positioning relative to the left boundary of the running text:

```
\hskip\layoutcolumnoffset{2}{\red Text positioned in column 2!}
```

Text positioned in column 2!

This mechanism is actually meant to ease the definition of complicated (title) pages where many text and graphic elements need to be anchored at well defined places. The layer mechanism is the most natural candidate for this.

```
\definelayer [text] \setupbackgrounds [text] [background=text]
```

When anchoring elements on a layer, you can specify absolute positions using the x and y keys but grid based positioning is possible with the column and line keys. We need to pass grid as location specifier.

```
\setlayer[text][column=1,line=47,location=grid]{these are not}
\setlayer[text][column=2,line=46,location=grid]{real columns}
\setlayer[text][column=3,line=47,location=grid]{but fake ones}
```

real columns

these are not

but fake ones

```
\setlayer [text] [column=1,line=32,location=grid]
  {\ruledvtop {\hsize\layoutcolumnwidth
     \definedfont[Regular sa 3]nitty\par gritty}}

\setlayer [text] [column=2,line=37,location=grid]
  {\ruledvbox {\hsize\layoutcolumnwidth
     \definedfont[Regular sa 3]nitty\par gritty}}

\setlayer [text] [column=3,line=42,location=grid]
  {\ruledvcenter {\hsize\layoutcolumnwidth
     \definedfont[Regular sa 3]nitty\par gritty}}
```

The data that goes into the layer is collected and flushed as soon as TEX builds the page. The buffer associated to the layer is then ready for new data (for the next page).

In this example, you can see that the baselines of the boxes (here visualized by dashed rules) are put at the specified lines. You can use the TEX box commands `\vbox`, `\vtop` and `\vcenter` to specify where the main baseline of the box content is positioned (at the top or bottom line, or centered).

```
\setlayer [text]
  [column=2,line=48,x=\layoutcolumnwidth,location=left]
  {\definedfont
     [Regular sa 2]%
   \framed
     [background=color,backgroundcolor=red,
      foregroundcolor=white,frame=off]
     {Why ain't I framed?}}
```

nitty
gritty

nitty
gritty

nitty
gritty

Why ain't I framed?

On the previous page we demonstrated a more complicated call to \setlayer and more features will be introduced in later chapters. We position the framed text in column 2 and at line 48. In addition we shift the text over the pseudo column width, i.e. we position the text at the right of the column. The location specifier aligns the text left from the point of positioning.

When we have set up the pseudo columns, we have access to a couple of variables:

\layoutcolumns	counter	number of columns
\layoutlines	counter	number of gridlines
\layoutcolumnwidth	dimension	width of one column
\layoutcolumnoffsetn	macro	position of column n

A rather common way to draw attention to a passage, is to add a background. In this chapter we will therefore discuss how to enhance your document with colorful areas that either or not follow the shape of your paragraph. Be warned: this chapter has so many backgrounds that you should consider wearing sunglasses.

In the previous paragraph we demonstrated two important features of the background handler: you can nest backgrounds and backgrounds can be tight or wide. Features like this will often be used in combination with others, like special section headers. The raw coding of the previous paragraph is therefore not representative.

```
\starttextbackground[intro]
A rather common way to draw attention to a passage, is to add
a background. In this chapter we will therefore discuss how
to enhance your document with \starttextbackground [subintro]
colorful areas that either or not follow the shape of your
paragraph. \stoptextbackground\ Be warned: this chapter has
so many backgrounds that you should consider wearing sunglasses.
\stoptextbackground
```

The outer background commands were defined as follows:

```
\definetextbackground[intro]
  [backgroundcolor=infogray,
   backgroundoffset=.25cm,
   offset=.5cm,
   frame=off,
   location=paragraph,
   color=white]
```

Here, the paragraph option ensures that the background covers the width of the body text. The inner background is defined in a similar way, but this time we choose text placement.

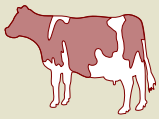
```
\definetextbackground[subintro]
  [backgroundcolor=textgray,
   backgroundoffset=0pt,
   offset=0pt,
   frame=off,
   location=text,
   color=blue]
```

In this document we use protruding characters (hanging punctuation) so we've chosen a rather large offset, one that also matches the rest of the page design.

Those who are familiar with the way TEX works will probably see what problems can occur with backgrounds like this. What happens for instance when we cross page boundaries, and how will more complicated paragraph shapes be

handled?

The current implementation tries to handle page breaks and paragraph shapes as good as possible. This works well in normal one–column mode and columnsets.



In this example, the paragraph shape is determined by the graphic placed left of the text. This feature is implemented using the \hangindent and \hangafter primitives, which means that we need to keep track of their state. In addition, we need to handle the indentation directives \leftskip, \rightskip and \parindent. Because backgrounds end up in a different background overlay, nesting them is no problem, and it is even possible to move them to the front and back, as we will demonstrate in a few lines.

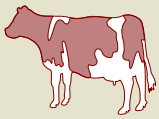
```
\placefigure[left]{none}{\externalfigure[detcow][width=2cm]}

\starttextbackground [A] In this example, the paragraph shape
is determined by the graphic placed left of the text.
\starttextbackground [B] This feature is implemented using the
\type {\hangindent} and \type {\hangafter} primitives, which
means that we need to keep track of their state. In addition,
we need to handle the indentation directives \type {\leftskip},
\type {\rightskip} and \type {\parindent}.  \stoptextbackground\
Because backgrounds end up in a different background overlay,
nesting them is no problem, and it is even possible to move
them to the front and back, as we will demonstrate in a few
lines. \stoptextbackground
```

The backgrounds were defined as:

```
\definetextbackground [A] [backgroundcolor=infogray]
\definetextbackground [B] [backgroundcolor=textgray]

\setuptextbackground
  [backgroundoffset=0pt,
   offset=0pt,
   frame=off,
   location=text]
```



In this example, the paragraph shape is determined by the graphic placed left of the text. Because backgrounds end up in a different background overlay, nesting them is no problem, and it is even possible to move them to the front and back, as we will demonstrate in a few lines.

This time we moved the inner background a few levels up. By default they reside at level=-1. This way, by using a non transparent color, we can hide information.

```
\setuptextbackground [B] [backgroundcolor=green,level=+2]
```

Unless you mess around too much with boxes, backgrounds work as expected in most situations. According to the Merriam–Webster on the authors laptop:

background	the part of a painting representing what lies behind objects in the foreground	one
foreground	the part of a scene or representation that is nearest to and in front of the spectator	two
spectator	one who looks on or watches	three

This is coded similar to normal running text. A table like this is in a way still part of the text flow. As floating body (see table 3.1) it can virtually end up everywhere.

background	the part of a painting representing what lies behind objects in the foreground	one
foreground	the part of a scene or representation that is nearest to and in front of the spectator	two
spectator	one who looks on or watches	three

Table 3.1

background	the part of a painting representing what lies behind objects in the foreground	one
foreground	the part of a scene or representation that is nearest to and in front of the spectator	two
spectator	one who looks on or watches	three

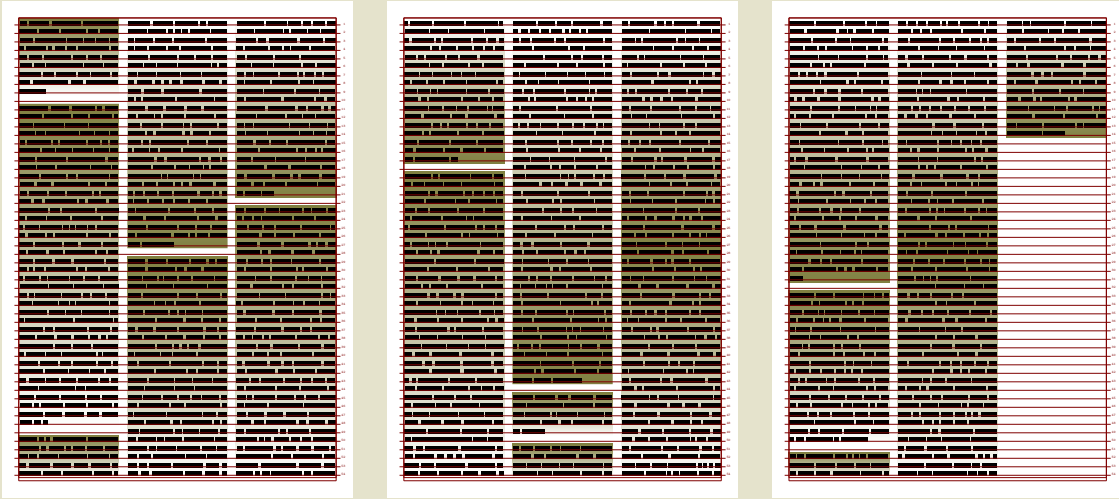
Mytable 3.1

Keeping track of the state of a paragraph in a table in combination with background is not entirely trivial. The current implementation evolved from less clever ones and, unless you start doing complicated box manipulations with the float content, works quite well. One reason why we made backgrounds work in tables (and especially floating tables) is that is was needed for typesetting books for primary and secundary education. In there, we want to be able to hide the answers that students are supposed to fill in.

In figure 3.1 you can see an advanced example of backgrounds running over columns. If you look carefully, you will notice that the background depends on the kind of background at hand:

1. the text starts and flows on
2. the text flows on (or stands alone)
3. the text flows on and ends

This information is available when you want to draw your own backgrounds. Here the graphic was defined as follows:



Page 1 Page 2 Page 3

Figure 3.1

```
\startuseMPgraphic{mpos:par:columnset:shade}
  path p, q, r ; numeric h ;
  for i=1 upto nofmultipars :
    p := multipars[i] ;
    q := multipars[i] topenlarged    -.5h ;
    r := multipars[i] bottomenlarged -.5h ;
    h := bbheight(p) ;
    if     multilocs[i] = 1 :
      linear_shade(p,8,boxfillcolor,white) ;
    elseif multilocs[i] = 2 :
      linear_shade(q,8,boxfillcolor,white) ;
      linear_shade(r,8,white,boxfillcolor) ;
    else :
      linear_shade(p,8,white,boxfillcolor) ;
    fi ;
  endfor ;
\stopuseMPgraphic
```

This graphic is hooked into the background setup by setting the `mp` variable.

```
\definetextbackground
  [shade]
  [location=paragraph,
   backgroundcolor=shadecolor,
   mp=mpos:par:columnset:shade,
   methode=mpos:par:columnset,
   leftoffset=\topskipgap,
   before=\blank,
   after=\blank]
```

todo: parameters

todo: moving up/down in layout stack
todo: default fills/line options

Once a design gets driven by typesetting on a grid, a bit more complex math becomes a problem. A naive designer may test a design using some simple psuedo math, using a few predefined superior characters —many fonts have ^{a} superior ^{2} available or $\frac{1}{2}$— but even simple school math is more than that.

So, in order to typeset math at all, we have to choose a proper math font collection. On the TEXlive distribution you find the Computer Modern Roman (cmr), Times (tx) and Palatino (px) math fonts. The first problem you need to solve is the relative font scaling. If a font collection is used, this is normally taken care of by the designer. However, if you combine different shape designs, some more work is needed.

```
\definetypeface [myface] [rm] [serif] [times]     [default]
\definetypeface [myface] [ss] [sans]  [helvetica] [default]
\definetypeface [myface] [tt] [mono]  [modern]    [default]
\definetypeface [myface] [mm] [math]  [palatino]  [default]
```

Fonts scaled in a similar way (say to 10 points) seldom look similar in size. You can test this with the macro:

```
\showfontstrip[myface]
```

This produced the table:

\rm	\ss	\tt	\mathematics	
xxxx	xxxx	xxxx	xxxx	
12345	12345	12345	12345	
(Agw)	(Agw)	(Agw)	(Agw)	
5.51399pt	6.306pt	5.16666pt	5.78998pt	(x height)
9.336pt	9.996pt	6.29994pt	9.33588pt	(m width)

A comfortable way out of this problem is relative font scaling, as demonstrated in:

```
\definetypeface
  [myface] [rm] [serif] [times]     [default]
\definetypeface
  [myface] [ss] [sans]  [helvetica] [default] [rscale=0.85]
\definetypeface
  [myface] [tt] [mono]  [modern]    [default] [rscale=1.05]
\definetypeface
  [myface] [mm] [math]  [palatino]  [default]
```

We now get a comparable x–height, which is a reasonable guarantee that the look and feel of a mixed font document will be acceptable.

\rm	\ss	\tt	\mathematics	
xxxx	xxxx	xxxx	xxxx	
12345	12345	12345	12345	
(Agw)	(Agw)	(Agw)	(*Agw*)	
5.51399pt	5.36014pt	5.425pt	5.78998pt	(x height)
9.336pt	8.49666pt	6.61496pt	9.33588pt	(m width)

A next step is to determine the optimal baseline distance. A good starting point is a distance of 120% of the body font size. This works out all right for in line math typeset as $1/2$ or $1/\sqrt{2}$ but can be disastrous when fractions are used: $\frac{1}{\sqrt{2}}$.

Normally TEX will take care of this by increasing the baseline distance locally, but how to explain this to grid thinking people. Here we keyed in:

```
... \tform {1/2} or \tform {1/\sqrt{2}} but can be disastrous
when fractions are used: \gform {\frac {1} {\sqrt{2}}} ...
```

The \tform macro takes care of switching to math mode, and \gform adds grid snapping to this. If you cannot convince the designer of the style (or if you are in control yourself) you may consider to increase the baseline distance with a few points. In that case, a logical question is: "How much should the baseline distance be?".

The answer is, ask ConTEXt:

```
\showminimalbaseline
```

This produces:

	->	14.46664pt	=	10.416pt	+	4.05064pt	=	14.46664pt	(ok)
(/)	->	9.3333pt	=	8.33333pt	+	0.99997pt	<	14.46664pt	(ok)
$\frac{1}{2}x^{2}_{3}$	->	14.36287pt	=	10.22478pt	+	4.13809pt	<	14.46664pt	(ok)

You can set the baseline distance of a typeface by adapting its environment:

```
\setupbodyfontenvironment[myface][line=15.5pt]
```

From now on, let's assume that everything is set up to our needs. In the following examples, the third line is too high to fit into the normal height and depth of a line. As a result, when \gform is used, additional spacing is applied.

```
math \gform{\frac{1}{2}}
more math \gform{x^{\frac{1}{2}} + x^{2} + x^{2}}
nothing but math \gform{x^{\frac{1}{2}}_{\frac{1}{2}} + x^{2}}
```

This gives:

math $\frac{1}{2}$

more math $x^{\frac{1}{2}} + x^2 + x^2$

nothing but math $x^{\frac{1}{2}}_{\frac{1}{2}} + x^2$

Because it is nearly impossible to determine how good or bad things will look, additional space is added in line quantities:

```
In educational math books (for the lower grades) authors like
to use fractions, for example \gform {x^{\frac{1}{2}}_{\frac{1}{2}}
+ x^{2} + 2}, and although it may improve readability, it
definitely increases the amount of whitespace when using grids.
```

In educational math books (for the lower grades) authors like to use fractions, for example $x^{\frac{1}{2}}_{\frac{1}{2}} + x^2 + 2$, and although it may improve readability, it definitely increases the amount of whitespace when using grids.

```
So, if this additional white space is too much for your
taste, you can instruct \CONTEXT\ to use half lines, as in:
\gform [-] {x\supsub{\frac{1}{2}}{\frac{1}{2}} + x\super{2}
+ 2}. You can turn on this feature for a whole document,
but in practice this kind of optimization is handy work.
```

So, if this additional white space is too much for your taste, you can instruct ConTEXt to use half lines, as in: $x^{\frac{1}{2}}_{\frac{1}{2}} + x^2 + 2$. You can turn on this feature for a whole document, but in practice this kind of optimization is handy work.

```
\setuptextformulas[step=halfline]
```

This grid snap mechanism is only active when document grid snapping is turned on. You can disable math snapping with:

```
\setuptextformulas[grid=no]
```

An inline formula typeset with \gform will not break across lines. However, you can compose a more complex formula from multiple \gform. The following code was used when developing this feature:

```
Crazy math \gform {1+x} or \gform {\dorecurse {10} {1+} 1 =
11} and even more crazy \gform {2^{2^2}_{1_1}}
again\dorecurse {20} { and again} \gform {\sqrt {\frac
{x^{5^5}} {\frac {1} {2}}}} even more\dorecurse {10} { and
more} \tform {\dorecurse {12} {\gform {\sqrt {\frac
{x^{5^5}} {3}}}+\gform {\sqrt {\frac {x^{5^5}} {\frac {1}
```

```
{2}}}}+}x=10}\dorecurse {10} { super crazy math}: \tform
{\dorecurse {15} {\gform {\sqrt {\frac {x^{5^5}} {3}}}+
\gform {\sqrt {\frac {x^{5^5}} {\frac {1} {2}}}}+ }x = 10},
and we're\dorecurse {20} { done}!
```

Crazy math $1+x$ or $1+1+1+1+1+1+1+1+1+1+1 = 11$ and even more crazy $2^{2^2}_{1_1}$ again and again $\sqrt{\frac{x^{5^5}}{\frac{1}{2}}}$ even more and more and more and more and more and more and more and more and more and more and more and more $\sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + x = 10$ super crazy math super crazy math super crazy math super crazy math super crazy math super crazy math super crazy math super crazy math super crazy math super crazy math: $\sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + x = 10$, and we're done done done done done done done done done done done done done done done done done done done done!

The space saving alternative, typeset with `step` set to `halfline`, looks as follows:

Crazy math $1+x$ or $1+1+1+1+1+1+1+1+1+1+1 = 11$ and even more crazy $2^{2^2}_{1_1}$ again and again $\sqrt{\frac{x^{5^5}}{\frac{1}{2}}}$ even more and more and more and more and more and more and more and more and more and more and more and more $\sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} +$

$\sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + x = 10$ super crazy math super crazy math super crazy math super crazy math super crazy math super crazy math super crazy math super crazy math super crazy math super crazy math: $\sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + \sqrt{\frac{x^{5^5}}{3}} + \sqrt{\frac{x^{5^5}}{\frac{1}{2}}} + x = 10$, and we’re done done done done done done done done done done done done done done done done done done done done!

So far we dealt with in–line math. For display math, we can fall back on the predefined math environments, or define our own. We distinguish four different kinds of display math. These are defined with `\defineformula` and can be setup with `\setupformula`.

	single par	single disp	multipar	multi disp
tag	sp	sd	mp	md
`spacebefore`	`none`	`none`		
`spaceafter`	`none`	`none`		
`indentnext`	`no`	`yes`	`no`	`yes`
`alternative`	`single`	`single`	`multi`	`multi`

These environments are used like:

```
\startsdformula
  x = 2 + 4x
\stopsdformula
```

So, the environments are charcterized by a short two character tag. The next examples demonstrate how they differ in spacing and indentation.

Single line display math:

$$x = 2 + 4x$$

looks like this.

Single line paragraph math:

$$x = 2 + 4x$$

looks like this.

Multi line display math:

$$x = \frac{2 + 4x}{\sqrt{135}}$$

[redacted] looks like this.

Multi line paragraph math: [redacted]

$$x = \frac{2 + 4x}{\sqrt{135}}$$

[redacted] looks like this.

It is possible to influence grid placement with the `\moveformula` command.

You can for instance move the formula one line up by using the `-top` directive. The previous multi line paragraph example was codes as:

```
\startmpformula
x = {{2 + 4x} \over {\sqrt{135}}}
\stopmpformula
```

We now change the definition to:

```
\moveformula[-top]
\startmpformula
x = {{2 + 4x} \over {\sqrt{135}}}
\stopmpformula
```

This time we get:

Multi line paragraph math: [redacted]

$$x = \frac{2 + 4x}{\sqrt{135}}$$

[redacted] looks like this.

Graphics, tables and alike are often treated as floating bodies. This means that when such a body does not fit on the current page, it will be moved to the next one. In the examples we will use figures, but everything we demonstrate here applies to all floats.

A side float is a float which placement one way or another depends on the text that follows it. In its simplest form, the text flows around it, for instance in:

```
\placefigure[left,none]{caption}{\framed[height=1cm]{graphic}}
```

The first keyword of such a call is treated as a placement directive, so this figure will be placed left. The none directive nils the caption.

graphic

When the figure does not fit on the page, a page break is issued. A figure can span multiple paragraphs. When a next graphic is placed the previous figure will be padded if needed. First an example of multiple paragraphs.

graphic

Multiple floats in a row will lead to padding. The amount of padding is a combination of empty lines and the normal white space following the float. The visual quality of the result depends on the graphic itself.

graphic

graphic

Here we show the baseline of the first paragraph after the float as well as the filler. The whitespace around a graphic also depends on the interparagraph whitespace. As with many automated mechanisms, compromises are made. A

one point smaller figure may result in an extra empty line.

Later we will demonstrate a lot of tuning options, but first we give a few more examples. Most of the tuning options can be driven by keywords as well as (global) settings.

```
\placefigure
  [left,nonumber]
  {caption} {\framed[height=1cm]{graphic}}
```

The nonumber keyword suppresses the label and figure number. You can do this for all figures with

```
\setupcaption[figure][number=no]
```

The previous placement command results in the following side float.

graphic

caption

Another handy keyword is none.

```
\placefigure
  [left,none]{quoting knuth}
  {\framed[height=1cm]{graphic}}
```

graphic

Control over spacing is exercised by means of the keywords high, low and fit.

```
\placefigure
  [left,none,high]{}
  {\framed[height=1cm]{graphic}}
```

graphic

```
\placefigure
  [left,none,high,low]{}
  {\framed[height=1cm]{graphic}}
```

graphic

```
\placefigure
  [left,none,fit]{}
  {\framed[height=1cm]{graphic}}
```

graphic

In the examples so far, we saw additional spacing around the graphic. We will now (for a while) disable the surrounding whitespace.

```
\setupfloat
  [figure]
  [sidespacebefore=none,
   sidespaceafter=none]
```

With these settings a simple left placement looks as follows. The top of the side float aligns with the maximum height of a line.

```
\placefigure
  [left,none]
  {} {\framed[height=1cm]{graphic}}
```

graphic

You can change the alignment by setting the sidealign variable, for instance:

```
\setupfloat
  [figure]
  [sidealign=line]
```

The three keywords height, line and depth can also be passed directly:

```
\placefigure
  [left,none,height]{}
  {\framed[height=1cm]{graphic}}
```

The three alignments disable the spacing before the float and show up as follows.

graphic

graphic

graphic

So far the floats took up space in the main text body area. In addition to the left (or right) directive we can use inner or outer to force left or right placement depending in the spread.

Instead of spoiling paper in the text areas, we can use the margin and edges: leftmargin and leftedge, rightmargin and rightedge, but also innermargin and outermargin, inneredge and outeredge.

The next couple of pages we will highlight the margins and edges so that we can see what happens.

```
\placefigure
  [leftmargin,none]
  {} {\framed{!}}
```

!

```
\placefigure
  [leftmargin,none]
  {} {\framed[width=1cm]{!}}
```

!

```
\placefigure
  [leftmargin,none]
  {} {\framed[width=1.5cm]{!}}
```

!

The placement directives can be combined with setting distance and width parameters, thereby not only opening a world of possibilities, but also creating confusion. Therefore, we will illustrate these features by cloning floats.

```
\definefloat
  [marginfigure]
  [figure]

\setupfloat
  [marginfigure]
  [leftmargindistance=-\leftmargintotal,
   default={left,none,low}]
```

The definition command clones figure into a new class of figures. There are two ways to use such a float :

```
\placefloat
  [marginfigure]
  {} {\framed[width=1.5cm]{!}}
```

or directly:

```
\placemarginfigure
  {} {\framed[width=1.5cm]{!}}
```

Both placement calls will result in a figure sticking into the margin.

!

By manipulating the margin distance, you can align graphics to vertical grid

lines, like the edge:

```
\definefloat
  [edgefigure]
  [figure]

\setupfloat
  [edgefigure]
  [leftmargindistance=-\innercombitotal,
   default={left,none,low,high}]
```

The \innercombitotal is one of the many available dimensions. This measure is the combined width of the margin and edge.

```
\placeedgefigure
  {} {\framed[width=1.5cm]{!}}
```

!

```
\placeedgefigure
  {} {\framed[width=\innercombitotal]{!}}
```

!

You need to be aware of the fact that the margins and edges are not related to the backspace and cutspace settings. When you set up a layout, you need to think of the right page as starting point. In a doublesided layout, the margins are swapped in the page composition stage. Unless you explicitly go to a left or right page, you don't know if your leftmargin will be swapped or not.

For this reason ConTEXt provides the inner and outer margin/edge dimensions. These are automatically synchronized when the float is constructed. So, if you want to automatically adapt the float placement and width to the current left margin in a doublesided document, you can use the inner dimensions.

dimension	left page	right page
\outermarginwidth	\leftmarginwidth	\rightmarginwidth
\innermarginwidth	\rightmarginwidth	\leftmarginwidth
\outermargindistance	\leftmargindistance	\rightmargindistance
\innermargindistance	\rightmargindistance	\leftmargindistance

Similar dimensions are available for the edges. You can save yourself some calculations by using the following dimensions:

\leftmargintotal	left margin width	+	left margin distance
\rightmargintotal	right margin width	+	right margin distance
\innermargintotal	inner margin width	+	inner margin distance
\outermargintotal	outer margin width	+	outer margin distance

As you may expect, the edge totals are available as well, which leave a few more totals, namely the combinations of margin and edge.

\leftsidetotal	left margin width	+	left edge total
\rightsidetotal	right margin width	+	right edge total
\innersidetotal	inner margin width	+	inner edge total
\outersidetotal	outer margin width	+	outer edge total
\leftcombitotal	left margin total	+	left edge total
\rightcombitotal	right margin total	+	right edge total
\innercombitotal	inner margin total	+	inner edge total
\outercombitotal	outer margin total	+	outer edge total

Adaptive back- and cutspace dimensions are also available:

\innerspacewidth	adaptive backspace
\outerspacewidth	adaptive cutspace

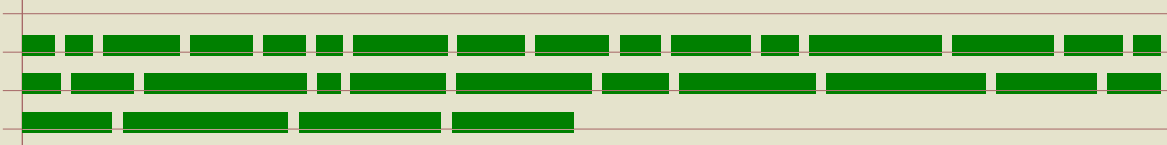
There is one drawback in using the inner and outer dimensions: if you also change the height of the float dynamically, you may end up in a kind of loop because a page break may occur at a non–expected place.

While negative values move float into the margin, positive values will move the float into the text. It will be of no surprise that you can also set the right margin distance. Keep in mind that this distance is not related to the text margin, but to the float margin.

```
\setupfloat
  [edgefigure]
  [leftmargindistance=-\outercombitotal,
   rightmargindistance=-\outercombitotal,
   default={outer,none,low,high}]
```

The locations inner and outer change with the left or right page.

```
\placeedgefigure
  {} {\framed[width=\outercombitotal]{!}}
```



!

```
\placeedgefigure
  {} {\framed[width=8cm]{!}}
```

!

As a result of manipulating the floats margin settings, the side floats can start in the margin (or edge). You should not confuse this with margin floats, i.e. side floats that are explicitly placed in the margins.

```
\placefigure[leftmargin,none]
  {} {\framed{!}}
```

!

```
\placefigure[leftmargin,none]
  {} {\framed[width=.5cm]{!}}
```

!

```
\placefigure[leftmargin,none]
  {} {\framed[width=1.5cm]{!}}
```

!

```
\placefigure[leftmargin,none]
  {} {\framed[width=5cm]{!}}
```

!

The margin side floats align to the margin and the edge floats to the edge. This way you can create bleeding figures.

```
\placefigure[leftedge,none]
  {} {\framed{!}}
```

There are situations where you don't know the dimensions in advance. In order to prevent unwanted side effects, for instance part of a graphic disappearing outside the page boundary, ConTEXt provides a few options. The most crude one is setting the criterium, as in:

```
\setupfloat
  [figure]
  [criterium=.25\textwidth]
```

This will automatically turn figures that are wider than 25% of the text width into normal floats instead of side floats. But let's not fall back on that feature now.

You can use maxwidth and minwidth variables to control the placement in more detail. The exact result depends on the settings of location. By default we center, but you can set the location to left or right to achieve a different alignment.

```
\definefloat
  [midmarginfigure]
  [figure]

\setupfloat
  [midmarginfigure]
  [minwidth=\leftmarginwidth,
   default={leftmargin,none}]
```

You can use maxwidth and minwidth variables to control the placement in more detail. The exact result depends on the settings of location. By default we center, but you can set the location to left or right to achieve a different alignment.

```
\placemidmarginfigure
  {} {\framed[width=1.5cm]{!}}
```

The meaning of maxwidth depends on the kind of float. First we place a left float with a width smaller than maxwidth.

```
\setupfloat[figure][maxwidth=2cm]

\placefigure[left,none]{}{\framed[width=1cm]{!}}
```

!

When the width exceeds the maxwidth, the float will be centered. This is because we have no reference alignment point.

```
\placefigure[left,none]{}{\framed[width=5cm]{!}}
```

!

In margin floats, the maxwidth settings have a different result. First we place a small graphic.

```
\setupfloat[figure][maxwidth=\leftmarginwidth]

\placefigure[leftmargin,none]{}{\framed[width=1cm]{!}}
```

!

Because the left and right margin of this document are the same —the edges differ— we don't need to use inner and outer dimensions.

```
\setupfloat[figure][maxwidth=\leftmarginwidth]
```

A wider than maxwidth graphic will behave like a mixture of a margin and text side float. Watch how we align the float to the margin.

```
\placefigure[leftmargin,none]{}{\framed[width=5cm]{!}}
```

!

Instead of setting the width you can give hanging a try. The next examples demonstrate this.

```
\placefigure[leftmargin,hanging,none]{}{\framed[width=5cm]{!}}
```

!

```
\placefigure[left,hanging,none]{}{\framed[width=5cm]{!}}
```

!

You can move down/up margin floats with the \movesidefloat macro. Such shifts come in handy when you have multiple side floats near to each other.

```
\movesidefloat [+2*line]
\placemidmarginfigure {} {\framed{!}}
```

!

Given the default placement template, this is equivalent to the following command. Watch out, a simple line has a different effect (alignment).

```
\placemidmarginfigure
  [leftmargin,none,+2*line]
  {} {\framed{!}}
```

Another nice keyword is long:

```
\placefigure
  [leftmargin,none,long]
  {} {\framed[height=2cm,width=2cm]{!}}

Watch how we move down. The effect is that we skip over the
margin figure.

\placefigure
  [leftmargin,none]
  {} {\framed[height=1cm,width=2cm]{!}}
```

Watch how we move down. The effect is that we skip over the margin figure.

!

!

```
\placefigure
  [leftmargin,none]
  {} {\framed[height=2cm,width=2cm]{!}}

Do we clash or not?

\placefigure
  [leftmargin,none]
  {} {\framed[height=2cm,width=2cm]{!}}

Did we clash or not?
```

Do we clash or not?

!

Did we clash or not?

!

There are a few macros that can be of help with solving clashes in side floats:

\flushsidefloats	This macro moves down as much as is needed to separate the side floats of each other.
\forgetsidefloats	this macro kind of forgets that a side float is in progress.

Use these macros with care. If you change the dimensions of the graphic and/or content involved, reconsider the use of these directives.

The next couple of spreads we will demonstrate some example definitions. These placements are taken from one of the styles we made for typesetting a series of school math books which illustrations and tables all over the pages.

First we fine tune the spacing around side floats and verbatim text.

```
\setupfloats
  [sidespacebefore=none,
   sidespaceafter=depth]

\setuptyping
  [margin=]
```

The placements have rather verbose names. In this case the word ‘edge’ is used to identify bleeding floats (with an cut–off margin of 3mm). The ‘text’ floats are side floats positioned in the main text flow.

```
\setupfloats
  [sidespacebefore=none,
   sidespaceafter=depth]
```

```
\setuptyping
  [margin=]
```

Watch how we define fall backs for too wide content (criterium as well as use maxwidth to manipulate the placement of content that falls off the margins.

The black rules are set up with:

```
\setupblackrules[color=tred,depth=0pt,height=1.5cm]
```

```
\setupfloat
  [marginfigure]
  [criterium=.5\textwidth,
   maxwidth=\rightmarginwidth,
   default={outermargin,none}]

\placemarginfigure{}{\blackrule[width=.25cm]}


\placemarginfigure{}{\blackrule[width=.5cm]}


\placemarginfigure{}{\blackrule[width=1cm]}


  \placemarginfigure{}{\blackrule[width=2cm]}


    \placemarginfigure{}{\blackrule[width=4cm]}


\placemarginfigure{}{\blackrule[width=8cm]}


\placemarginfigure{}{\blackrule[width=16cm]}
```

```
\setupfloat
  [marginfigure]
  [criterium=.5\textwidth,
   maxwidth=\rightmarginwidth,
   default={outermargin,none}]

\placemarginfigure{}{\blackrule[width=.25cm]}

\placemarginfigure{}{\blackrule[width=.5cm]}

\placemarginfigure{}{\blackrule[width=1cm]}

\placemarginfigure{}{\blackrule[width=2cm]}

\placemarginfigure{}{\blackrule[width=4cm]}

\placemarginfigure{}{\blackrule[width=8cm]}

\placemarginfigure{}{\blackrule[width=16cm]}
```

```
\setupfloat
  [middlemarginfigure]
  [minwidth=\rightmarginwidth,
   criterium=\snijwit,
   location=middle,
   default={outermargin,none}]

\placemiddlemarginfigure{}{\blackrule[width=.25cm]}

\placemiddlemarginfigure{}{\blackrule[width=.5cm]}

\placemiddlemarginfigure{}{\blackrule[width=1cm]}

\placemiddlemarginfigure{}{\blackrule[width=2cm]}

\placemiddlemarginfigure{}{\blackrule[width=4cm]}

\placemiddlemarginfigure{}{\blackrule[width=8cm]}

\placemiddlemarginfigure{}{\blackrule[width=16cm]}
```

```
\setupfloat
  [middlemarginfigure]
  [minwidth=\rightmarginwidth,
   criterium=\snijwit,
   location=middle,
   default={outermargin,none}]

\placemiddlemarginfigure{}{\blackrule[width=.25cm]}

\placemiddlemarginfigure{}{\blackrule[width=.5cm]}

\placemiddlemarginfigure{}{\blackrule[width=1cm]}

\placemiddlemarginfigure{}{\blackrule[width=2cm]}

\placemiddlemarginfigure{}{\blackrule[width=4cm]}

\placemiddlemarginfigure{}{\blackrule[width=8cm]}

\placemiddlemarginfigure{}{\blackrule[width=16cm]}
```

```
\setupfloat
  [middlefigure]
  [default={here,none}]

\placemiddlefigure{}{\blackrule[width=.25cm]}

\placemiddlefigure{}{\blackrule[width=.5cm]}

\placemiddlefigure{}{\blackrule[width=1cm]}

\placemiddlefigure{}{\blackrule[width=2cm]}

\placemiddlefigure{}{\blackrule[width=4cm]}

\placemiddlefigure{}{\blackrule[width=8cm]}
```

```
\setupfloat
  [middlefigure]
  [default={here,none}]

\placemiddlefigure{}{\blackrule[width=.25cm]}

\placemiddlefigure{}{\blackrule[width=.5cm]}

\placemiddlefigure{}{\blackrule[width=1cm]}

\placemiddlefigure{}{\blackrule[width=2cm]}

\placemiddlefigure{}{\blackrule[width=4cm]}

\placemiddlefigure{}{\blackrule[width=8cm]}
```

```
\setupfloat
  [textfigure]
  [criterium=.5\textwidth,
   default={outer,none}]

   \placetextfigure{}{\blackrule[width=.25cm]}

    \placetextfigure{}{\blackrule[width=.5cm]}

      \placetextfigure{}{\blackrule[width=1cm]}

          \placetextfigure{}{\blackrule[width=2cm]}

                \placetextfigure{}{\blackrule[width=4cm]}

                          \placetextfigure{}{\blackrule[width=8cm]}

\placetextfigure{}{\blackrule[width=16cm]}
```

```
\setupfloat
  [textfigure]
  [criterium=.5\textwidth,
   default={outer,none}]

\placetextfigure{}{\blackrule[width=.25cm]}

\placetextfigure{}{\blackrule[width=.5cm]}

\placetextfigure{}{\blackrule[width=1cm]}

\placetextfigure{}{\blackrule[width=2cm]}

\placetextfigure{}{\blackrule[width=4cm]}

\placetextfigure{}{\blackrule[width=8cm]}

\placetextfigure{}{\blackrule[width=16cm]}
```

```
\setupfloat
  [leftfigure]
  [criterium=.5\textwidth,
   default={left,none}]

   \placeleftfigure{}{\blackrule[width=.25cm]}

    \placeleftfigure{}{\blackrule[width=.5cm]}

      \placeleftfigure{}{\blackrule[width=1cm]}

          \placeleftfigure{}{\blackrule[width=2cm]}

                \placeleftfigure{}{\blackrule[width=4cm]}

                          \placeleftfigure{}{\blackrule[width=8cm]}

\placeleftfigure{}{\blackrule[width=16cm]}
```

```
\setupfloat
  [leftfigure]
  [criterium=.5\textwidth,
   default={left,none}]

   \placeleftfigure{}{\blackrule[width=.25cm]}

    \placeleftfigure{}{\blackrule[width=.5cm]}

      \placeleftfigure{}{\blackrule[width=1cm]}

          \placeleftfigure{}{\blackrule[width=2cm]}

                \placeleftfigure{}{\blackrule[width=4cm]}

                          \placeleftfigure{}{\blackrule[width=8cm]}

\placeleftfigure{}{\blackrule[width=16cm]}
```

```
\setupfloat
  [rightfigure]
  [criterium=.5\textwidth,
   default={right,none}]

\placerightfigure{}{\blackrule[width=.25cm]}

\placerightfigure{}{\blackrule[width=.5cm]}

\placerightfigure{}{\blackrule[width=1cm]}

\placerightfigure{}{\blackrule[width=2cm]}

\placerightfigure{}{\blackrule[width=4cm]}

\placerightfigure{}{\blackrule[width=8cm]}

\placerightfigure{}{\blackrule[width=16cm]}
```

```
\setupfloat
  [rightfigure]
  [criterium=.5\textwidth,
   default={right,none}]

\placerightfigure{}{\blackrule[width=.25cm]}

\placerightfigure{}{\blackrule[width=.5cm]}

\placerightfigure{}{\blackrule[width=1cm]}

\placerightfigure{}{\blackrule[width=2cm]}

\placerightfigure{}{\blackrule[width=4cm]}

\placerightfigure{}{\blackrule[width=8cm]}

\placerightfigure{}{\blackrule[width=16cm]}
```

```
\setupfloat
  [bleedfigure]
  [criterium=.5\textwidth,
   leftmargindistance=-1mm,
   rightmargindistance=-1mm,
   default={backspace,none}]

\placebleedfigure{}{\blackrule[width=.25cm]}


\placebleedfigure{}{\blackrule[width=.5cm]}


\placebleedfigure{}{\blackrule[width=1cm]}


\placebleedfigure{}{\blackrule[width=2cm]}


\placebleedfigure{}{\blackrule[width=4cm]}


\placebleedfigure{}{\blackrule[width=8cm]}


\placebleedfigure{}{\blackrule[width=16cm]}
```

```
\setupfloat
  [bleedfigure]
  [criterium=.5\textwidth,
   leftmargindistance=-1mm,
   rightmargindistance=-1mm,
   default={backspace,none}]

\placebleedfigure{}{\blackrule[width=.25cm]}


\placebleedfigure{}{\blackrule[width=.5cm]}


\placebleedfigure{}{\blackrule[width=1cm]}


\placebleedfigure{}{\blackrule[width=2cm]}


\placebleedfigure{}{\blackrule[width=4cm]}


\placebleedfigure{}{\blackrule[width=8cm]}


\placebleedfigure{}{\blackrule[width=16cm]}
```

```
\setupfloat
  [bleedfigure]
  [criterium=.5\textwidth,
   leftmargindistance=-1mm,
   rightmargindistance=-1mm,
   default={cutspace,none}]

\placebleedfigure{}{\blackrule[width=.25cm]}


\placebleedfigure{}{\blackrule[width=.5cm]}


\placebleedfigure{}{\blackrule[width=1cm]}


\placebleedfigure{}{\blackrule[width=2cm]}


\placebleedfigure{}{\blackrule[width=4cm]}


\placebleedfigure{}{\blackrule[width=8cm]}


\placebleedfigure{}{\blackrule[width=16cm]}
```

```
\setupfloat
  [bleedfigure]
  [criterium=.5\textwidth,
   leftmargindistance=-1mm,
   rightmargindistance=-1mm,
   default={cutspace,none}]

\placebleedfigure{}{\blackrule[width=.25cm]}


\placebleedfigure{}{\blackrule[width=.5cm]}


\placebleedfigure{}{\blackrule[width=1cm]}


\placebleedfigure{}{\blackrule[width=2cm]}


\placebleedfigure{}{\blackrule[width=4cm]}


\placebleedfigure{}{\blackrule[width=8cm]}


\placebleedfigure{}{\blackrule[width=16cm]}
```

In this chapter we will discuss a few more tricks to control float placement. This control is needed if you want to typeset documents in a semi desk top publishing way.

When you combine technical graphics, you may wish to align the content optically. This can be done with the `offset` command. We will demonstrate this with a couple of METAPOST graphics:

```
\startreusableMPgraphic{alpha}
  fill fullsquare xyscaled(  2cm,  2cm) withcolor \MPcolor{red} ;
  fill unitsquare xyscaled(+.5cm,+.5cm) withcolor \MPcolor{gray} ;
\stopreusableMPgraphic

\startreusableMPgraphic{beta}
  fill fullsquare xyscaled(  2cm,  2cm) withcolor \MPcolor{red} ;
  fill unitsquare xyscaled(+.5cm,-.5cm) withcolor \MPcolor{gray} ;
\stopreusableMPgraphic

\startreusableMPgraphic{gamma}
  fill fullsquare xyscaled(  2cm,  2cm) withcolor \MPcolor{red} ;
  fill unitsquare xyscaled(-.5cm,-.5cm) withcolor \MPcolor{gray} ;
\stopreusableMPgraphic

\startuseMPgraphic{delta}
  fill fullsquare xyscaled(  2cm,  2cm) withcolor \MPcolor{red} ;
  fill unitsquare xyscaled(-.5cm,+.5cm) withcolor \MPcolor{gray} ;
\stopuseMPgraphic

\startcombination[2*2]
  {\reuseMPgraphic{alpha}} {alpha}
  {\reuseMPgraphic {beta}}  {beta}
  {\reuseMPgraphic{gamma}} {gamma}
  {\reuseMPgraphic{delta}} {delta}
\stopcombination
```

In figure 6.1 we place these graphics in a `2*2` grid. As you can see, the centers don't align well.

In figure 6.2 the centers of the graphic align well. This is accomplished by adding some space around the graphics.

```
\startcombination[2*2]
  {\offset[rightoffset=1cm]  {\reuseMPgraphic{alpha}}} {alpha}
  {\offset[bottomoffset=.5cm]{\reuseMPgraphic {beta}}}  {beta}
  {\offset[bottomoffset=.5cm]{\reuseMPgraphic{gamma}}} {gamma}
  {\offset[leftoffset=1cm]   {\reuseMPgraphic{delta}}} {delta}
\stopcombination
```

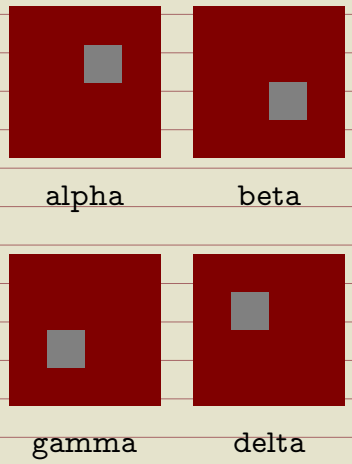


Figure 6.1

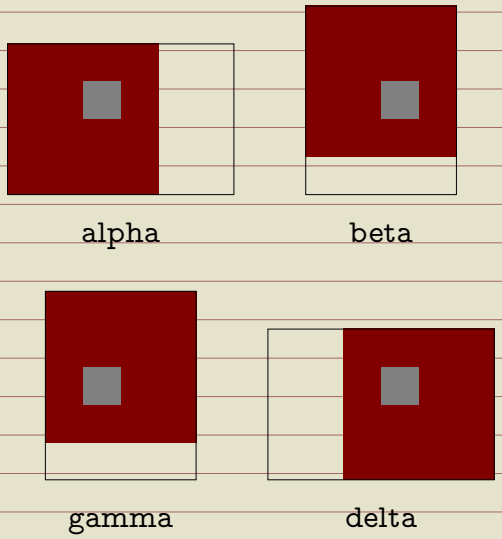


Figure 6.2

If we align the centers vertically, as demonstrated in figure 6.2 we can stick to a few bottom offsets.

```
\startcombination[4*1]
                                {\reuseMPgraphic{alpha}}  {alpha}
  {\offset[bottomoffset=.5cm]{\reuseMPgraphic {beta}}}  {beta}
  {\offset[bottomoffset=.5cm]{\reuseMPgraphic{gamma}}} {gamma}
                                {\reuseMPgraphic{delta}}  {delta}
\stopcombination
```

These examples demonstrate that the dimensions change with the offset. You can retain the dimensions but still align them by using the `x` and `y` parameter. This kind of manipulations will often result in a ugly spacing because the placement macros handle on the original dimensions. Figure 6.4 demonstrates this.

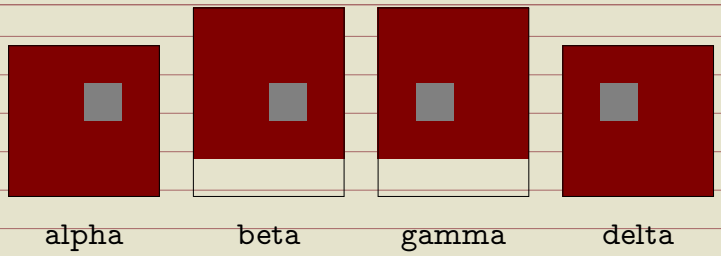


Figure 6.3

```
\startcombination[4*1]
                    {\reuseMPgraphic{alpha}}  {alpha}
  {\offset[y=-.5cm]{\reuseMPgraphic {beta}}}  {beta}
  {\offset[y=-.5cm]{\reuseMPgraphic{gamma}}} {gamma}
                    {\reuseMPgraphic{delta}}  {delta}
\stopcombination
```

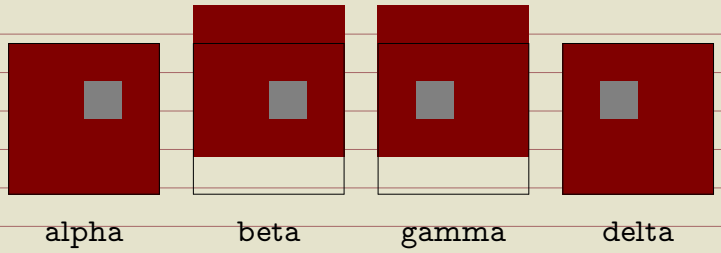


Figure 6.4

In the previous chapter we demonstrated how a side float can be moved up or down by providing a placement directive or by preceding the placement with `\movesidefloat`. Such a move can be used to align a graphic with particular line of text. This command can also be used for alignment purposes similar to the `\offset` command. We will demonstrate this with the following graphics.

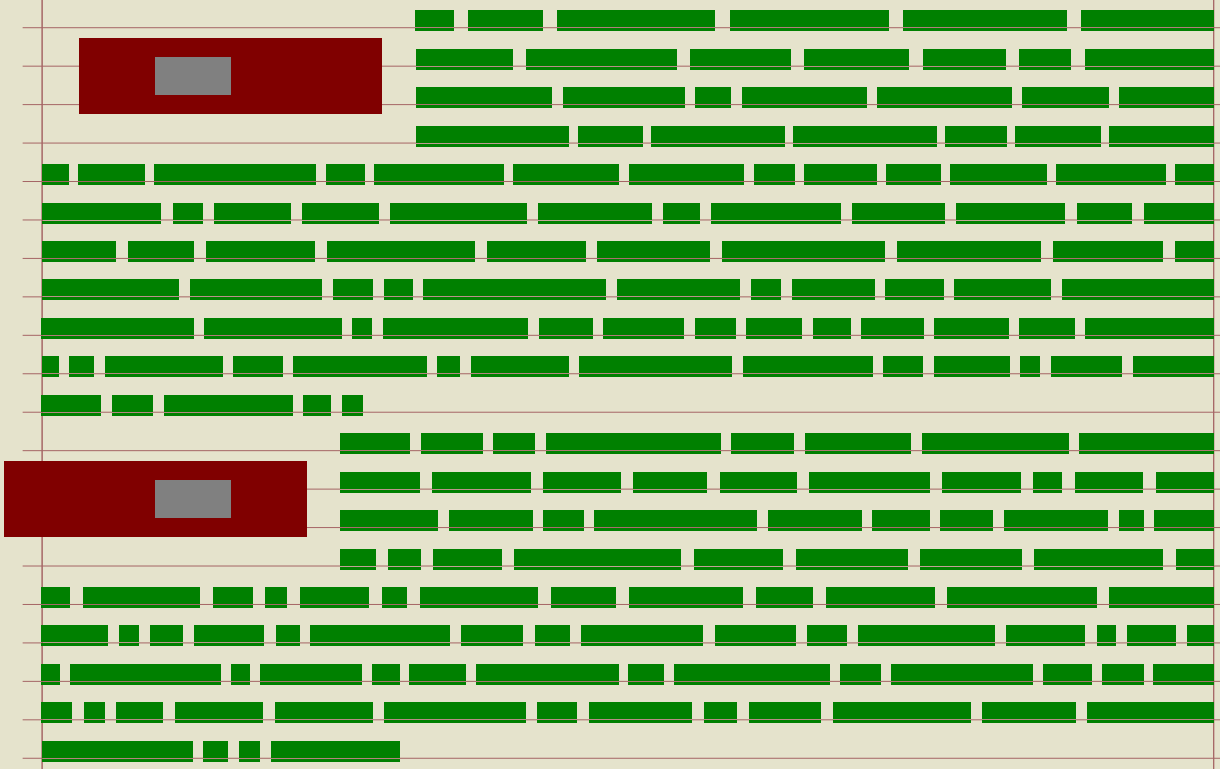
```
\startreusableMPgraphic{gnu}
  fill fullsquare xyscaled( 4cm, 1cm) withcolor \MPcolor{red} ;
  fill unitsquare xyscaled(-1cm,.5cm)
                  shifted (0,-.25cm) withcolor \MPcolor{gray} ;
\stopreusableMPgraphic

\startreusableMPgraphic{gnat}
  fill fullsquare xyscaled( 4cm, 1cm) withcolor \MPcolor{red} ;
  fill unitsquare xyscaled(+1cm,.5cm)
                  shifted (0,-.25cm) withcolor \MPcolor{gray} ;
\stopreusableMPgraphic
```

In the next two examples we shift the gnu and gnat graphics horizontally in order to get them aligned. The move does not change the dimensions of the float, but they do influence the paragraph shape.

```
\movesidefloat [x=.5cm]
\placefigure   [left,none] {} {\reuseMPgraphic{gnu}}
```

```
\movesidefloat [x=-.5cm]
\placefigure   [left,none] {} {\reuseMPgraphic{gnat}}
```



It is possible to shift vertically by setting `y`, but this is often a bad idea and definitely may spoil alignment of graphics to the grid. If you have to revert to this trick, you are probably working in document screw–up mode. This is why in grid mode, we automatically round to an equal number of lines.

If you know what text you're dealing with and also can be sure about the height of a graphic, you can trick ConTEXt to ignore the dimensions of a graphic. Here we use the graphic:

```
\startreusableMPgraphic{gnome}
  fill fullsquare xyscaled(2cm, 1cm) withcolor \MPcolor{red} ;
  fill fullsquare xyscaled(1cm,.5cm) withcolor \MPcolor{gray} ;
\stopreusableMPgraphic

\placefigure[leftmargin,none,reset]{}{\reuseMPgraphic{gnome}}
```

The graphic is moved into the margin (`leftmargin`), has no caption (`none`), and all kind of tricky housekeeping is reset (`reset`).

Now the next graphic is not influenced by the previous one, so we can place them close to each other. Use these tricks with care, especially if your document source is reused and the typeset products are not carefully checked.

```
\placefigure[left,none,high,low]{}{\reuseMPgraphic{gnome}}
```

When ConTEXt tries to determine if a float fits, it makes a couple of assumptions, for instance that the available room equals the text height minus the height of the text so far. You can slightly influence the way these values are interpreted by setting the calculation method. You can set the methods as follows:

```
\setupfloats[textmethod=0,sidemethod=1]
```

Method 0 just looks at the raw dimensions, while method 1 lessens the maximum text height by one percent, thereby playing safe. Method 2 takes a window of 1 point. This may lead to better decisions since we may run into rounding errors of several scaled points (which is small but troublesome). Method 2 is well suited when typesetting on a grid, because there everything has to fit in a rounded number of lines, which leaves no room for rounding errors.

grid mode	yes	no
sidemethod	2	1
textmethod	2	0

As you may know by now, we can use the directives high, low, height, depth and line to influence the spacing around a side float. A real tight spacing can be achieved with fit.

```
\placefigure[left,fit,none]{}{some graphic}
```

This kind of placements only make sense in special situations, because normally you don't want the graphic to touch the text.

If you think that this is all a user may want, you're wrong. It is not imaginary that graphics have small pieces sticking out and/or lots of white space as part of their design. In that case, the boundingbox can be set to a smaller size.

Now, when handling a side float, ConTEXt first places the float, and then starts with typesetting the paragraph, cleverly avoiding the graphic. However, when the graphic is virtually larger than its known size, it may cover part of the preceding paragraph.

How come that the graphic starting this paragraph does not do that? It is because we explicitly moved it to the background. This involves some preparation. At the document level, we define a layer called graphic.

```
\definelayer[graphics][position=yes]
```

The position directive tells ConTEXt that it should honor the position of the graphic. Next we must make sure that this layer is placed.

```
\setupbackgounds[page][background=graphics]
```

Now we're ready to move graphics to this layer:

```
\placefigure
  [left,fit,none]
  {}{\setlayer[graphics]{graphic}}
```

It's now a small step to more advanced movements. Say that you want to move the graphic a little bit to the left. In that case you can tell the layer placement to do so.

```
\placefigure
  [left,fit,none]{}{\setlayer[graphics][hoffset=-12pt]{graphic}}
```

From this you can deduce that there is also a movement in the vertical direction using `voffset`. In addition you can anchor the graphic using the `location` parameter and provide offsets.

As soon as you run into situations where float placement is to be consistently enforced, you will feel the need for dedicate placement macros. For example:

```
\definefloat
  [somefloat]
  [figure]

\setupfloat
  [somefloat]
  [sidespaceafter=,
   sidespacebefore=,
   default={left,none}]
```

Instead of resetting the side spacing, we could have default to `high,low`, but this way we can overload the default placement and still get zero spacing.

Throughout this manual we discuss features related to overlays and layers. These permit you to move content around in ways that either or not depend on the text flow. We have now come to another trick based on these mechanisms: bleeding.

When printing a document, you need to take into account that when graphics go beyond the page boundary, you need to compensate for inaccuracies in cutting the pages. Such graphics are called bleeding graphics and the amount of bleed is often a few millimeters.

The best way to handle such graphics is to use the correct dimensions and play with the edge widths and distances in combination with backspace and cutspace. In a properly set up layout and by using a well designed set of predefined graphic placements, you can handle this quite well. A bleeding figure can be defined as follows:

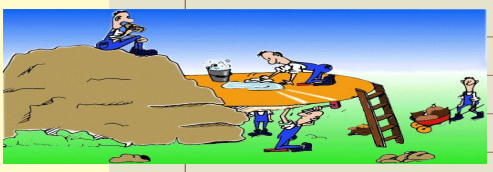
```
\definefloat
  [edgefigure]
  [figure]
```

```
\setupfloat
  [edgefigure]
  [default={inner,height,high,low,none},
   maxwidth=4cm]

\defineexternalfigure
  [edgefigure]
  [width=\dimexpr(\backspace+4cm-1mm),
   lines=4]
```

The default placement is preconfigured to have no additional vertical space and align on the height of a line (this is default behaviour so the height key is redundant here. The 1mm in the previous definition simulates what happens when a page is cut off slightly wrong: we get an anoying gap.

```
\placeedgefigure
  {}
  {\externalfigure[hacker][edgefigure]}
```

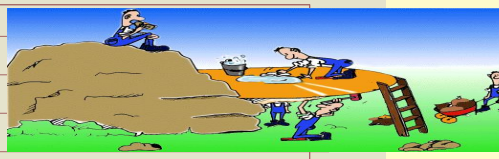


One of the nice things about TEX is that you can fine tune dimensions pretty well. So, instead of the previous placement, which turns out rather ugly, we can come up with a better one:

```
\setupfloat
  [edgefigure]
  [default={inner,height,high,low,none},
   maxwidth=4cm,
   margin=\strutdepth]

\defineexternalfigure
  [edgefigure]
  [width=\dimexpr(\backspace+4cm+2mm),
   height=\dimexpr(3\lineheight+\strutheight)]
```

This time we take no risk and add 2mm to the dimensions so thta we can be sure that the edge of the graphic falls outside the page boundary.



The ConTEXt resourse library modules provide means to report back the dimensions of graphics used in a document, so that you can develop (tune) them with the proper dimensions. In practice a slightly wider than normal graphic (scaling it horizontally a few millimeters more) does not harm the visual appearance that much, so adapting a graphic to this kind of bleeding is not really needed.

In addition to this (rather natural) way of adding bleed to a graphic, you can apply the `\bleed` macro. In the previously discussed method the figure placement mechanisms work with the real dimensions. The `bleed` macro is using scaling in a different way: from the prespective of ConTEXt the graphic remains its original dimensions and the figure placement mechanisms will act accordingly. We will give a couple of examples of using this macro.

Permitted bleeding locations are `l`, `r`, `t`, `b`, `lr`, `bl`, `br`, `tl` and `tr`.

```
\placesomefloat
  [left,none,fit]
  {}
  {\setupbleeding[offset=5mm]%
   \bleed[width=5cm,height=1cm,location=l]
     {\externalfigure[mill][bleed]}}
```

```
\placesomefloat
  [left,none,fit]
  {}
  {\setupbleeding[offset=2mm]%
   \bleed[width=5cm,height=1cm,location=l]
     {\externalfigure[mill][bleed]}}
```

The amount of bleeding depends on the postprocessing. In the previous paragraph we used a bleed offset of 5mm, and here we used 2mm. Because the graphic is scaled in order to match the bleed, it will be slightly destorted. With small values this will go unnoticed. You can set the offset with:

```
\setupbleeding[offset=5mm]
```

Bleeding itself is accomplished by the `\bleed` macro as in:

```
\bleed
  [width=5cm,height=1cm,location=l]
  {\externalfigure[mill][width=\bleedwidth,height=\bleedheight]}
```

It is kind of awkward to pass those two dimensions so here is a shorter way of doing the same:

```
\bleed
  [width=5cm,height=1cm,location=l]
  {\externalfigure[mill][bleed]}
```

In fact, this uses the following definition:

```
\defineexternalfigure[bleed][width=\bleedwidth,height=\bleedheight]
```

You can influence the scaling of a graphic by setting the `stretch` parameters. The location parameter determines the direction of the stretch: `l` (left), `r` (right), `t` (top), `b` (bottom) or a combination of these. We will now combine the previous example code with this knowledge.

```
\placefigure
  [left]
  {}
  {\bleed
     [stretch=no,voffset=0pt,hoffset=1cm]
     {\externalfigure[detcow][bleed]}}
```

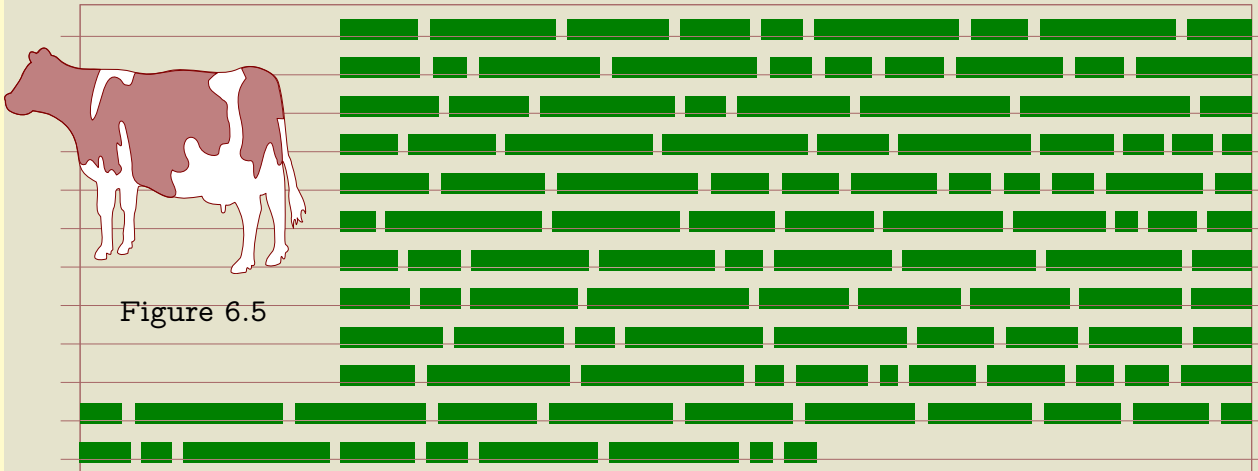


Figure 6.5

```
\placefigure
  [left]
  {}
  {\bleed
     [width=5cm,height=3cm,location=l]
     {\externalfigure[detcow][bleed]}}
```

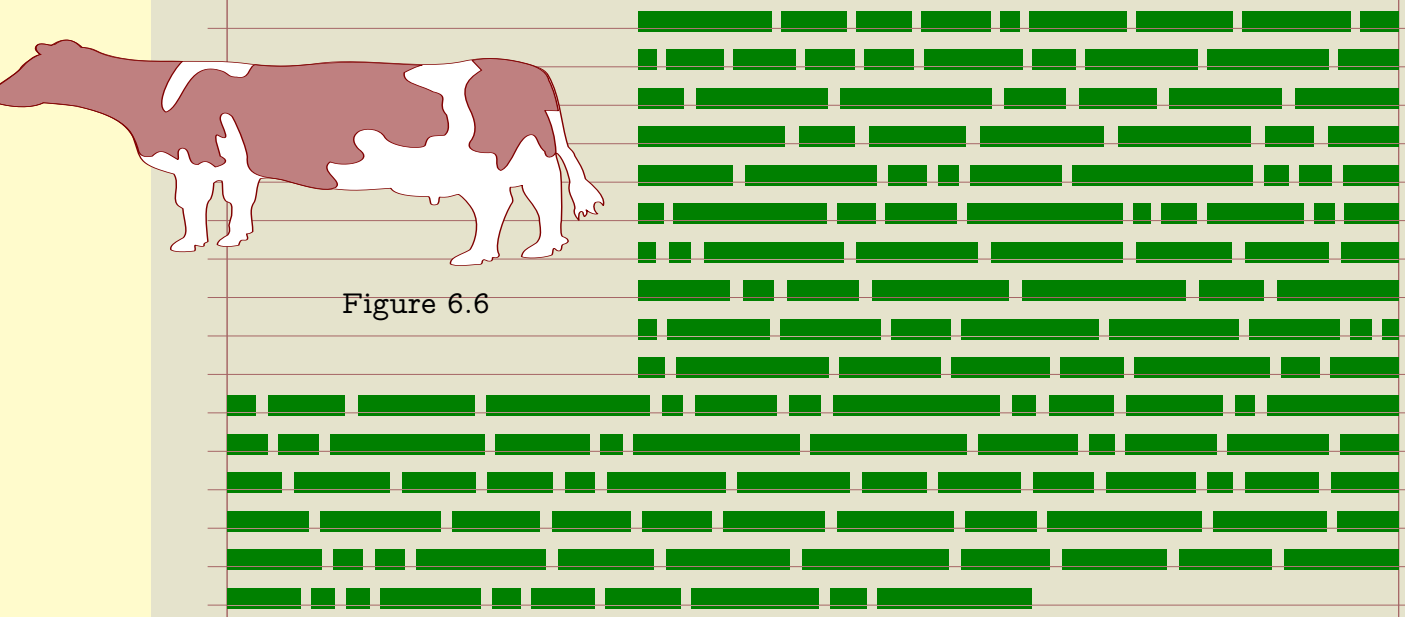


Figure 6.6

```
\placefigure
  [right]
  {}
  {\bleed
     [width=5cm,height=3cm,location=r]
     {\externalfigure[detcow][bleed]}}
```

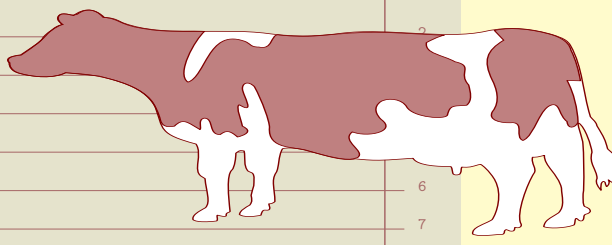


Figure 6.7

You can combine this feature with layers. We will now show a few applications which may look like magic at first glance, but will become natural to your repertoire once you have played with them.

The next example moves the graphic to a layer associated with the (current) page.

```
\placefigure
  [right,none]
  {}
  {\setlayer
     [graphics]
     {\bleed
        [width=5cm,height=3cm,location=rb]
        {\externalfigure[detcow][bleed]}}}
```

You can also predefine locations where graphics (or other content) needs to be anchored. A direct call to anchor looks as follows:

```
\placefigure
  [left,none]
  {}
  {\anchor
     [text-1]
     [location=lt,hoffset=max,voffset=max]
     [width=3cm,height=3cm,frame=on]%
     {\externalfigure[detcow][width=5cm,frame=on]}}
```

This will anchor a graphic in one of the text layers, but at the cost of specifying this in the document source. One way around this is to predefine anchors.

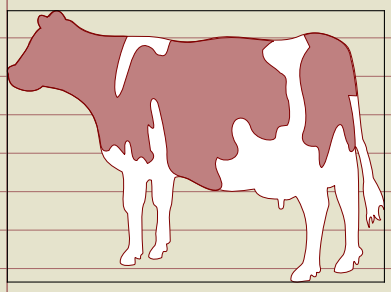
```
\defineanchor[rightbottom][text-1][location=lt,hoffset=max,voffset=max]
\defineanchor[righttop]    [text-1][location=lb,hoffset=max]
\defineanchor[leftbottom]  [text-1][location=rt,voffset=max]
\defineanchor[lefttop]     [text-1][location=rb]
```

We will apply this to a predefined float type.

```
\definefloat[myfigure][figure]
\setupfloat[myfigure][sidespaceafter=,sidespacebefore=]
```

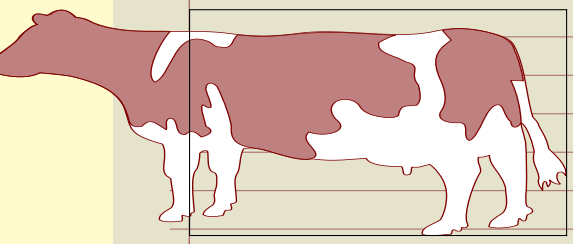
Our previous example can now be reduced to:

```
\placemyfigure
  [left,none]
  {}
  {\anchor[rightbottom]
     {\externalfigure[detcow][width=5cm,frame=on]}}
```

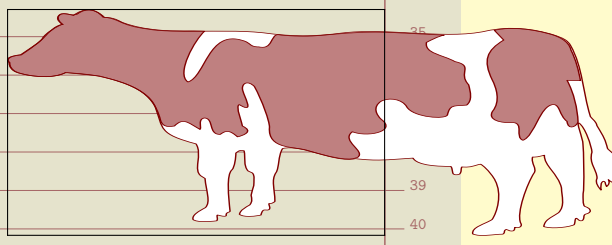


You can still specify dimensions and anchors can be combined with bleeding. Of course this kin dof mixed usage means that you need to have some feeling fo rwhat these mechanisms do.

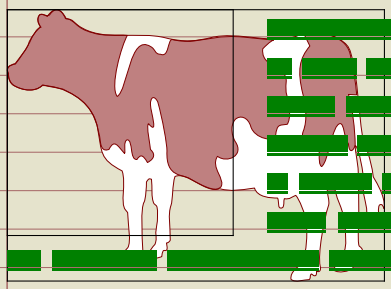
```
\placemyfigure
  [left,none]
  {}
  {\anchor
     [rightbottom]
     [width=5cm,height=3cm,frame=on]
     {\bleed
        [width=5cm,height=3cm,location=l]
        {\externalfigure[detcow][bleed]}}}
```



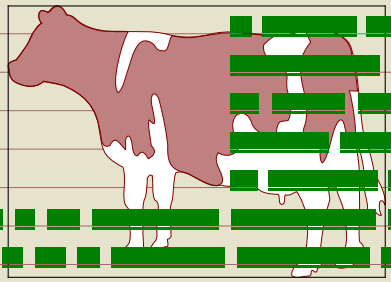
```
\placemyfigure
  [right,none]
  {}
  {\anchor
     [rightbottom]
     [width=5cm,height=3cm,frame=on]
     {\bleed
        [width=5cm,height=3cm,location=r]
        {\externalfigure[detcow][bleed]}}}
```



```
\placemyfigure
  [left,none]
  {}
  {\anchor
     [lefttop]
     [width=3cm,height=3cm,frame=on]
     {\externalfigure[detcow][width=5cm,frame=on]}}
```



```
\placemyfigure
  [left,none]
  {}
  {\anchor
     [lefttop]
     [width=3cm,height=3cm,frame=on]
     [offset=.5cm]
     {\externalfigure[detcow][width=5cm,frame=on]}}
```



Todo: parameter specifications of all those macros.

The background mechanisms present in ConTEXt have evolved over time and with computers becoming faster, you can expect new functionality to show up and existing functionality to start using this technology. A simple background consist of a colored area. Many commands accept settings like:

```
...[background=color,backgroundcolor=red,backgroundoffset=3pt]
```

Instead of such an area you can define one or more so called overlays:

```
\defineoverlay[one][...]
\defineoverlay[two][...]

...[background={one,two}]
```

The name overlay comes from the fact that you stack them on top of each other. A special overlay is foreground, and deep down in ConTEXt there are more predefined overlays.

In the MetaFun manual you will find example of usage, so here we stick to a simple code snippet for testing this functionality:

```
\defineoverlay[one][\green A]
\defineoverlay[two][\red   B]

\framed[background=one]        {1}
\framed[background={one,two}] {1---2}
```

The rather ugly result is:

A 1-B-2

You can construct overlays by using TEX boxing primitives or commands like \framed. Alternatively you can use another mechanism: layers. Layers collect content and flush that when asked, for instance when an overlay is constructed. Layers can be independent of a page, or bound to a specific page number, left or right hand pages. Here we look at independent layers.

All these mechanisms are fine tuned for cooperating with the output routine (the part of TEX that deals with composing pages) and are well interact quite well with METAPOST graphics. Details of usage and tricks are revealed in this manual as well as in styles that come with ConTEXt. In this chapter we will apply layers to graphics. For this we need a few setups, like:

```
\setupbackgrounds
  [page]
  [background=pagegraphics]
```

Here we have set up the page background to use an overlay called

pagegraphics. However, instead of an overlay, we will use a layer. This layer will collect content that goes into the page background. Whenever a layer is defined, an overlay is automatically defined as well.

```
\definelayer
  [pagegraphics]
  [x=-2mm,
   y=-2mm,
   width=\paperwidth,
   height=\paperheight]
```

When you fill a layer with content, you can influence the placement with the x and y parameters as well as hoffset and voffset, whichever you prefer. The reference point and alignment are set with corner and location.

Live can be made easier by using presets, especially for our intended usage. The following presets are predefined.

```
\definelayerpreset
  [lefttop] [corner={left,top},location={right,bottom}]
\definelayerpreset
  [righttop] [corner={right,top},location={left,bottom}]
\definelayerpreset
  [leftbottom] [corner={left,bottom},location={right,top}]
\definelayerpreset
  [rightbottom] [corner={right,bottom},location={left,top}]
```

Because for this layer we have also preset the x and y, those corners are laying a few millimeters outside the page area. We have preset the size as well, otherwise all corners would end up in the top left corner.

We will now fill this layer. Because the layer is hooked into the page, it will be flushed when the page is constructed. After the page is written to the output file, the layer is emptied, unless its state is set to repeat.

```
\setlayer [extras] [preset=lefttop]     {\externalfigure[hacker]}
\setlayer [extras] [preset=righttop]    {\externalfigure[hacker]}
\setlayer [extras] [preset=leftbottom]  {\externalfigure[hacker]}
\setlayer [extras] [preset=rightbottom] {\externalfigure[hacker]}
```

Once you got the picture of layering, you will start using this mechanism for all kind of tasks. Instead of putting layers in a background, you can also directly place them, by using one of the two (equivalent) commands:

```
\composedlayer{identifier}
\placelayer[identifier]
```

Layer are quite convenient for defining title pages, colofons, and special section heads, especially in combination with \framed.

On top of the layer mechanism we have build a few more mechanisms, like

ornaments. You can use ornaments to annotate graphics in such a way that the dimensions stay unchanged.

```
\defineornament
  [affiliation]
  [rotation=90,corner={right,bottom},location={right,top},
   hoffset=-.25ex]
  [frame=on,background=color,backgroundcolor=red,offset=0pt]
```

The negative offset will overlay the text outside the graphic. The meaning of the sign of coordinates and offsets depends on the corner. Figure 7.1 shows the result. We have put the reference point in the right bottom corner. The ornament is anchored at the right top corner of the dot you can picture at the reference point. The ornament is shifted .25ex outwards.

```
\placefigure
  {}
  {\affiliation{graphic}{\externalfigure[hacker][width=3cm]}}
```

Figure 7.1 Number 1

There are two ways to handle the placement. Alternative a will change the dimensions of the graphic according to the size of the ornament, while alternative b acts as a pure overlay. In figure 7.2 the ornament is not taken into account when calculating the dimensions of the graphic. This is often the prefered placement, because this way the (often small) ornament will not it will not spoil visual alignment of similar graphics.

```
\defineornament
  [affiliation]
  [rotation=90,corner={right,bottom},location={right,top},
   hoffset=-.25ex,alternative=b]
  [frame=on,background=color,backgroundcolor=red,offset=0pt]
```

Figure 7.2 Number 2

A positive offset will place the ornament on top of the graphic (see figure 7.3).

```
\defineornament
  [affiliation]
  [rotation=90,corner={right,bottom},location={left,top},
   hoffset=.25ex,voffset=.25ex,alternative=a]
  [background=color,style=\ss\tfxx,backgroundcolor=white,offset=0pt]
```

Figure 7.3 Number 3

You need to play a bit with this mechanism in order to get a feeling for what the parameters do.

```
\defineornament
  [affiliation]
  [rotation=90,corner={right,bottom},location={left,top},
   hoffset=.25ex,voffset=.25ex,alternative=b]
  [background=color,style=\ss\tfxx,backgroundcolor=white,offset=0pt]
```

Figure 7.4 Number 4

Because the text is normally typeset quite small, you'd better use a font that can be scaled down a lot.

```
\definefont[AffiliationFont][Sans sa .25]

\defineornament
  [SomeAffiliation]
  [rotation=90,corner={right,bottom},location={right,top},
   hoffset=-.125ex,alternative=b]
  [style=AffiliationFont,offset=0pt]
```

This affiliation is used as:

```
\placefigure
  {Affiliations normally are typeset pretty small.}
  {\SomeAffiliation
     {author: Hester De Weert}
     {\externalfigure[hacker]}}
```

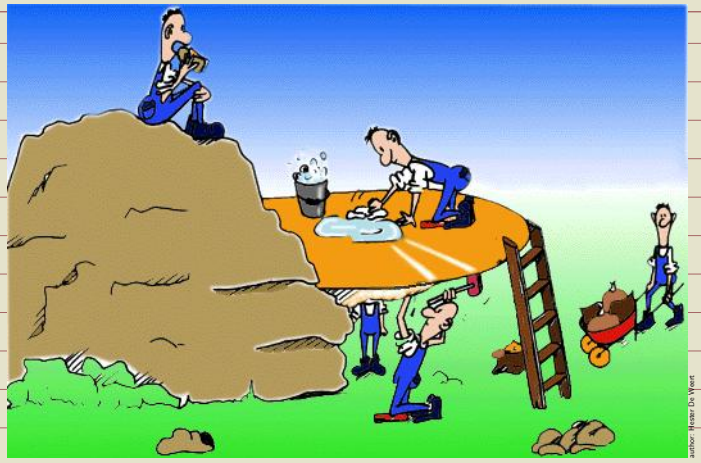
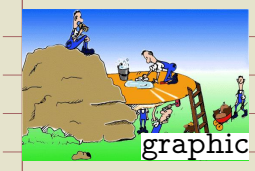


Figure 7.5 Affiliations normally are typeset pretty small.

Ornaments are implemented in terms of layers and collectors. A few examples demonstrate how these can be used.

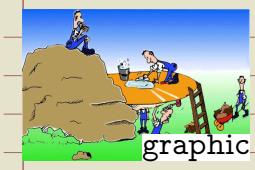
```
\layeredtext
  [corner={right,bottom},location={left,top}]
  [background=color,backgroundcolor=white,offset=0pt]
  {graphic}
  {\externalfigure[hacker][width=3cm]}
```




```
\layeredtext
  [rotation=90,corner={right,bottom},location={right,top}]
  [frame=on,offset=0pt]
  {graphic}
  {\externalfigure[hacker][width=3cm]}
```


```
\layeredtext
  [rotation=90,corner={left,bottom},location={left,top}]
  [frame=on,offset=0pt]
  {graphic}
  {\externalfigure[hacker][width=3cm]}
```


```
\collectedtext
  [corner={right,bottom},location={left,top}]
  [background=color,backgroundcolor=white,offset=0pt]
  {graphic}
  {\externalfigure[hacker][width=3cm]}
```




```
\collectedtext
  [rotation=90,corner={right,bottom},location={right,top}]
  [frame=on,offset=0pt]
  {graphic}
  {\externalfigure[hacker][width=3cm]}
```


```
\collectedtext
  [rotation=90,corner={left,bottom},location={left,top}]
  [frame=on,offset=0pt]
  {graphic}
  {\externalfigure[hacker][width=3cm]}
```

There are several methods to construct titlepages, headers, and other compositions. Of course there are the low level box constructors like `\hbox`, `\vbox` and positioners like `\hskip`, `\hfill` and alike.

Another option is to fall back on the low level box macros in the ConTEXt support file `supp-box` or the higher level `\framed` macro. You can use `\framed` nested and by cleverly using the offsets and dimensions you can do a lot.

Layers are another means. You can or instance construct a title page in the

following way:

```
\definelayer
  [titlepage]
  [width=\textwidth,
   height=\textheight]

\setlayer
  [titlepage]
  [preset=righttop,location={left,bottom},y=1cm,x=1cm]
  {\definedfont[Regular at 60pt]Welcome}

\setlayer
  [titlepage]
  [preset=rightbottom,location={right,top},y=2cm,x=2cm]
  {\definedfont[Regular at 30pt]By Me}
```

This just fills the layer. Placement is done with:

```
\startstandardmakeup
  \flushlayer[titlepage]
\stopstandardmakeup
```

or alternatively:

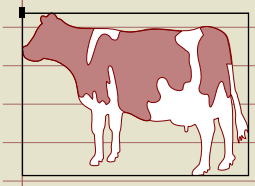
```
\setupbackgrounds[text][background=titlepage]
\startstandardmakeup \stopstandardmakeup
\setupbackgrounds[text][background=]
```

Another way to collect content is to use a collector. A collector starts out empty with:

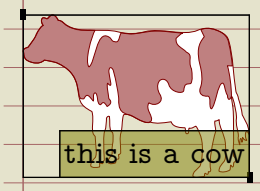
```
\definecollector[test][state=repeat]
```

We can now stepwise fill this collector. For educational purposes we've turn of tracing so that you can see what the anchor points.

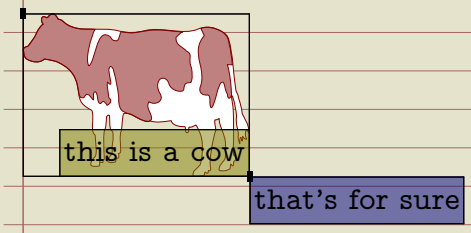
```
\setcollector[test]
  [location={right,bottom}]
  {\externalfigure[detcow][frame=on,width=3cm]}
```



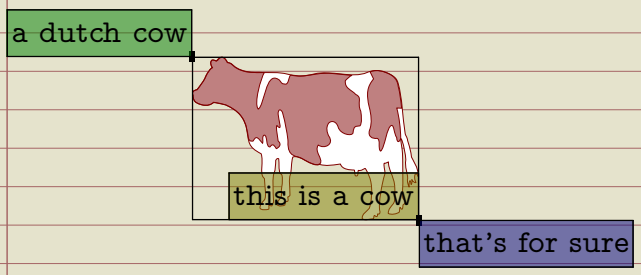
```
\setcollector[test]
  [corner={right,bottom},location={left,top}]
  {\framed[background=color,backgroundcolor=tyellow]{this is a
cow}}
```



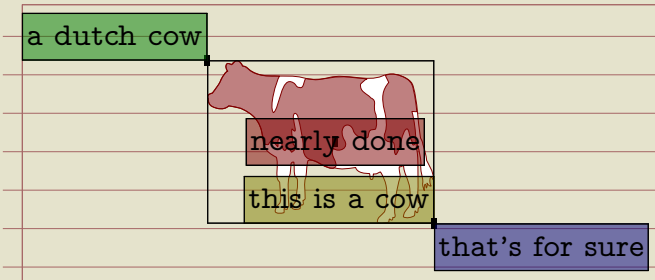

```
\setcollector[test]
  [corner={right,bottom},location={right,bottom}]
  {\framed[background=color,backgroundcolor=tblue]{that's for
sure}}
```




```
\setcollector[test]
  [corner={left,top},location={left,top}]
  {\framed[background=color,backgroundcolor=tgreen]{a dutch cow}}
```

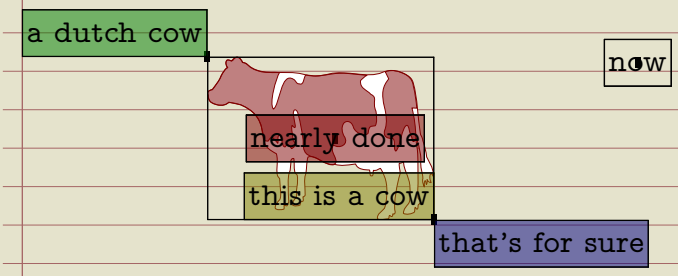



```
\setcollector[test]
  [corner=middle,
   location=middle]
  {\framed[background=color,backgroundcolor=tred]{nearly done}}
```



In addition to the parameters shown here, you can also provide additional ones: x, y, offset, hoffset and voffset for positioning and rotation for (as expected) rotating the content in steps of 90 degrees. As with layers, the coordinates and offsets can be used intermixed.

```
\setcollector[test]
  [hoffset=4cm,
   voffset=-1cm,
   corner=middle,
   location=middle]
  {\framed{now}}
```



We can show the intermediate results because we have set the state of this collector to repeat. In this case you need to erase the content manually, using:

```
\resetcollector[test]
```

The chapter titles of this document are (as usual in a ConTEXt document) typeset by the \chapter macro. When thinking about implementing a non standard head, those familiar with ConTEXt's head macros will probably first think of using one of the hooks, like:

```
\setuphead[chapter][command=\MyChapterHead]
```

Here we have followed a different approach. First we set up the chapter head. The empty directive instructs ConTEXt not to place the head itself, but still to include the associated data in the text stream. This means that we will not see a chapter title, but that there will be an entry in the table of contents, that references will be set up, that so called marks will be available, etc.

```
\setuphead
  [chapter]
  [placehead=empty,
   header=chapter,
   style=\BigText,
   numberstyle=\BigNumber]
```

The header parameters instructs the head handler to mark this page as special with regards to header texts. This text is set up as follows:

```
\definetext
  [chapter]
  [header]
  [\setups{chapter}]
  []
```

The setups are just series of typesetting instructions. For the sake of readability, we have split them up.

```
\startsetups chapter
  \setups[chapter:title]
  \setups[chapter:number]
  \setups[chapter:finish]
\stopsetups
```

The setups will use a dedicated layer for the chapter title:

```
\definelayer
  [chapter]
  [width=\dimexpr(\makeupwidth+\cutspace),
   height=\headerheight]
```

The following code uses a macro \setlayerframed. This is a combination between \setlayer and \framed. We use two placement macros to typeset the title and number. When doing so, we need to take care of both numbered chapters and unnumbered titles.

```
\startsetups chapter:title

  \setlayerframed
    [chapter]
    [x=\dimexpr(\makeupwidth+\cutspace),location={left,bottom}]
    [height=\headerheight,
     foregroundcolor=white,
     background=color,
     backgroundcolor=blue,
     frame=off,
     offset=none,
```

```
        align={right,lohi}]
      {\hbox spread .5\cutspace
         {\hss
          \doiftextelse{\placeheadtext[chapter]}%
            {\placeheadtext[chapter]}%
            {\placeheadtext[title]}%
          \hss}\space
        \vskip.5cm}

\stopsetups
```

Definitions like these may look complicated but in practice you will construct them piecewise.

```
\startsetups chapter:number

  \setlayerframed
    [chapter]
    [x=\dimexpr(\makeupwidth+\cutspace),
     y=\vsize,
     location={left,bottom}]
    [width=\dimexpr(\cutspace-\rightmargindistance),
     height=\dimexpr(\cutspace-\rightmargindistance),
     foregroundcolor=white,
     background=color,
     backgroundcolor=red,
     frame=off,
     offset=none,
     align={middle,lohi}]
    {\hbox to \hsize
       {\hskip.5cm\hss
        \doifmode{*bodypart}{\placeheadnumber[chapter]}%
        \hss}}

\stopsetups
```

The finishing touch is just a dummy frame with the chapter background. We could have used the headertext background instead.

```
\startsetups chapter:finish

  \framed
    [width=\makeupwidth,
     height=\headerheight,
     background=chapter,
     frame=off]
    {}
```

```
\stopsetups
```

As the title of this manual suggests: it's in the details. Most of our time is spent in optimizing spacing issues. If you're designing the layout yourself, for a large part you can fall back on the consistent spacing provided by TEX, i.e. think in terms of em's, ex's and fractions or multiples of \bodyfontsize, as well as base you're dimensions on those provided by the layout. When dealing with translating a dtp layout into something TEX, definitions like the above will often look more messy.

In this manual we pay quite some words on ways to snap your content on a grid. When dealing with grids, we often run into conflicting situations where we have to make the best of it. Let's again deal with an aspect of graphics.

One of the strong points of TEX is that it can deal with graphics automatically, which means that you seldom have to tweak dimensions or placements unless . . . you're dealing with grids. In that case you need to make sure that the height of graphics consistently match the height of lines (or multiples of lines). It is for this purpose that the graphic inclusion macro has a grid entry.

We will illustrate its usage using a dedicated figure class where we have set the space between figure and caption to zero.

```
\definefloat[tightfigure][tightfigures][figure]
\setupcaption[tightfigure][inbetween=]
```

The grid parameter controls rounding of the height of a graphic in the following way:

yes	safe rounding to an equal number of lines
fit	tight rounding to an equal number of lines
height	same as yes but incremented by linedepth

On the next pages we demonstrate the effects of these settings. At the bottom of a page we show the placement commands. On the last pages we've hidden the captions with:

```
\setupfloat[tightfigure][default={here,none}]
```

As you will notice, the height option is handy when the caption is positioned directly under the graphic.

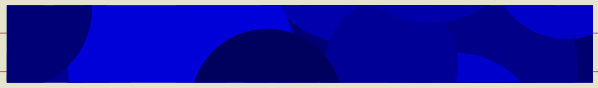


Figure 8.1

Figure 8.2

Figure 8.3

Figure 8.4

Figure 8.5

```
\placetightfigure{}{\externalfigure[dummy][lines=1.3,grid=yes]}
\placetightfigure{}{\externalfigure[dummy][lines=1.4,grid=yes]}
\placetightfigure{}{\externalfigure[dummy][lines=1.5,grid=yes]}
\placetightfigure{}{\externalfigure[dummy][lines=1.6,grid=yes]}
\placetightfigure{}{\externalfigure[dummy][lines=1.7,grid=yes]}
```

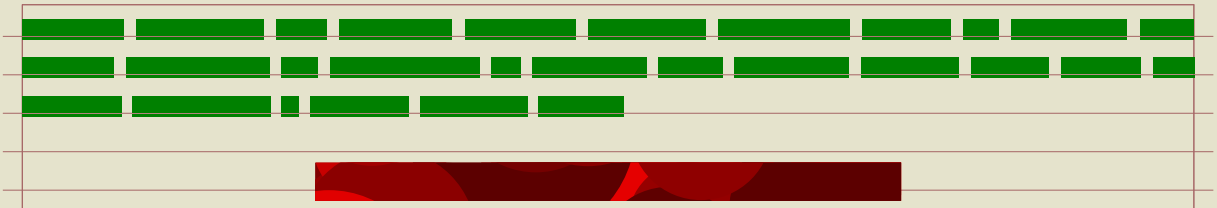


Figure 8.6

Figure 8.7

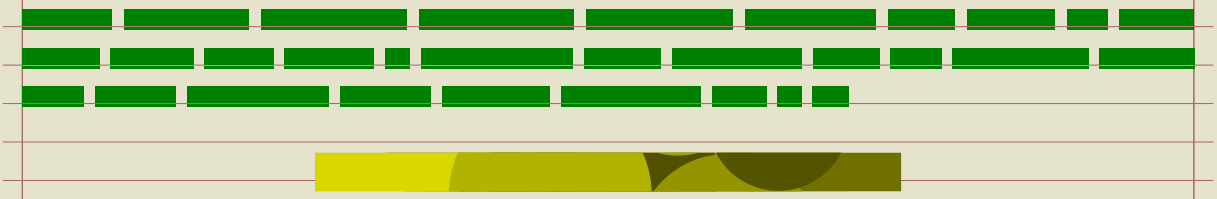


Figure 8.8

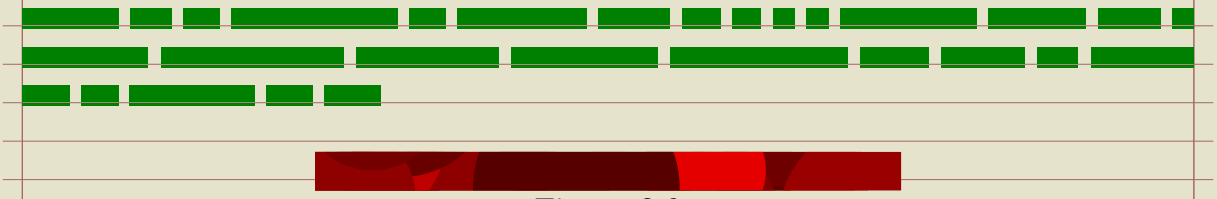


Figure 8.9

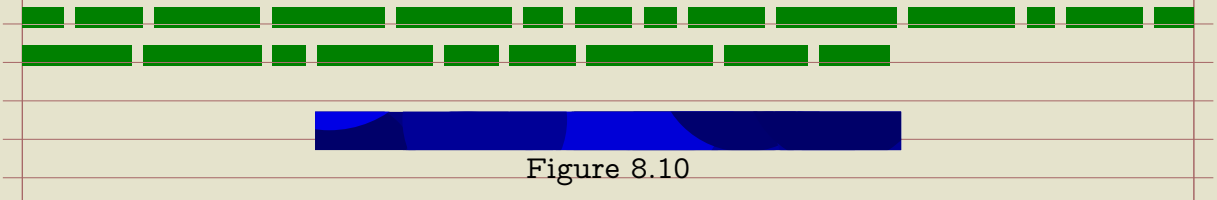
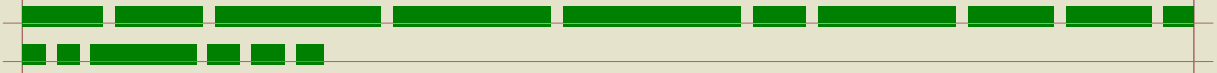


Figure 8.10



```
\placetightfigure{}{\externalfigure[dummy][lines=1.3,grid=fit]}
\placetightfigure{}{\externalfigure[dummy][lines=1.4,grid=fit]}
\placetightfigure{}{\externalfigure[dummy][lines=1.5,grid=fit]}
\placetightfigure{}{\externalfigure[dummy][lines=1.6,grid=fit]}
\placetightfigure{}{\externalfigure[dummy][lines=1.7,grid=fit]}
```

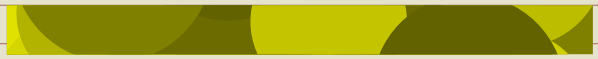


Figure 8.11

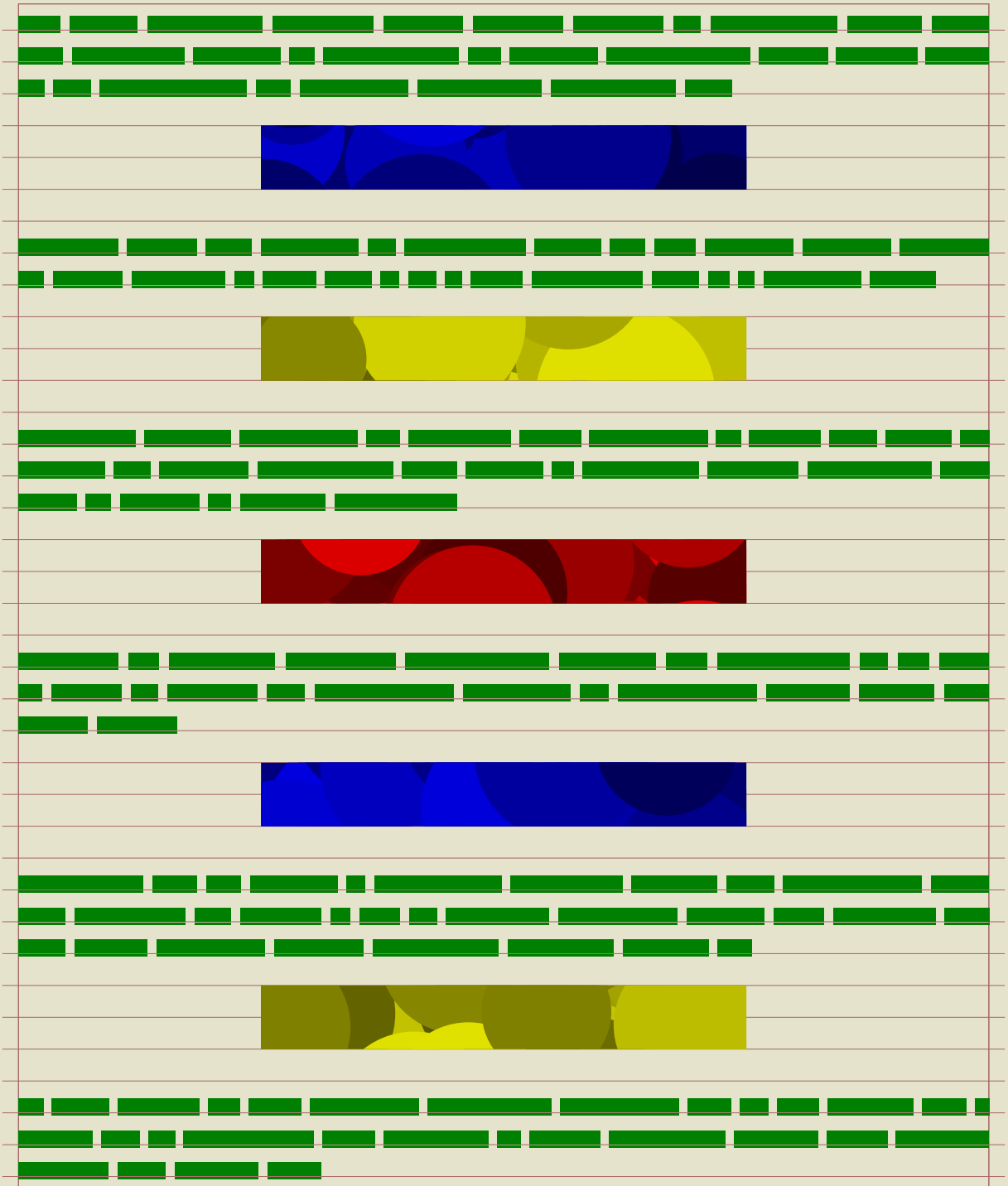
Figure 8.12

Figure 8.13

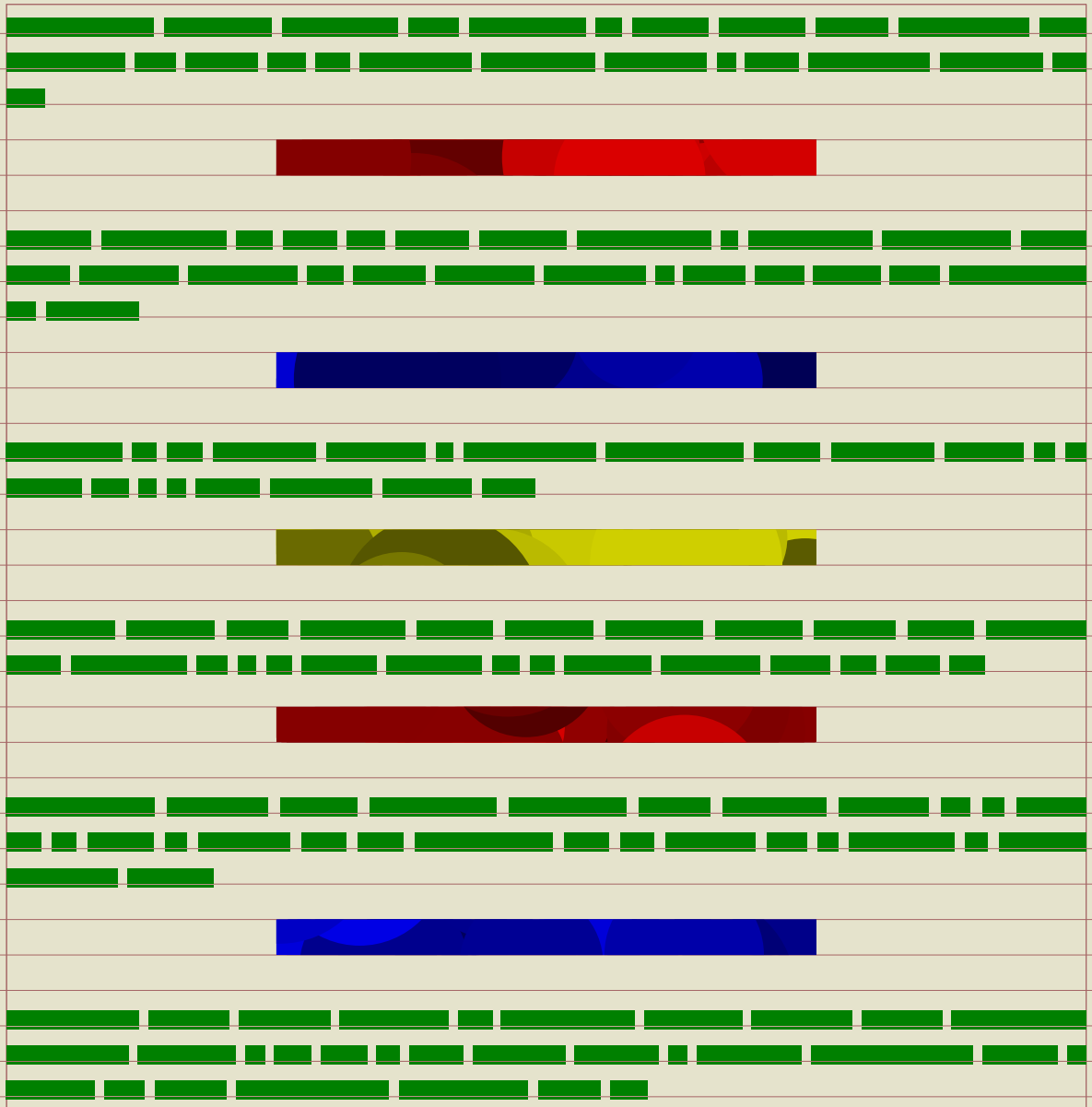
Figure 8.14

Figure 8.15

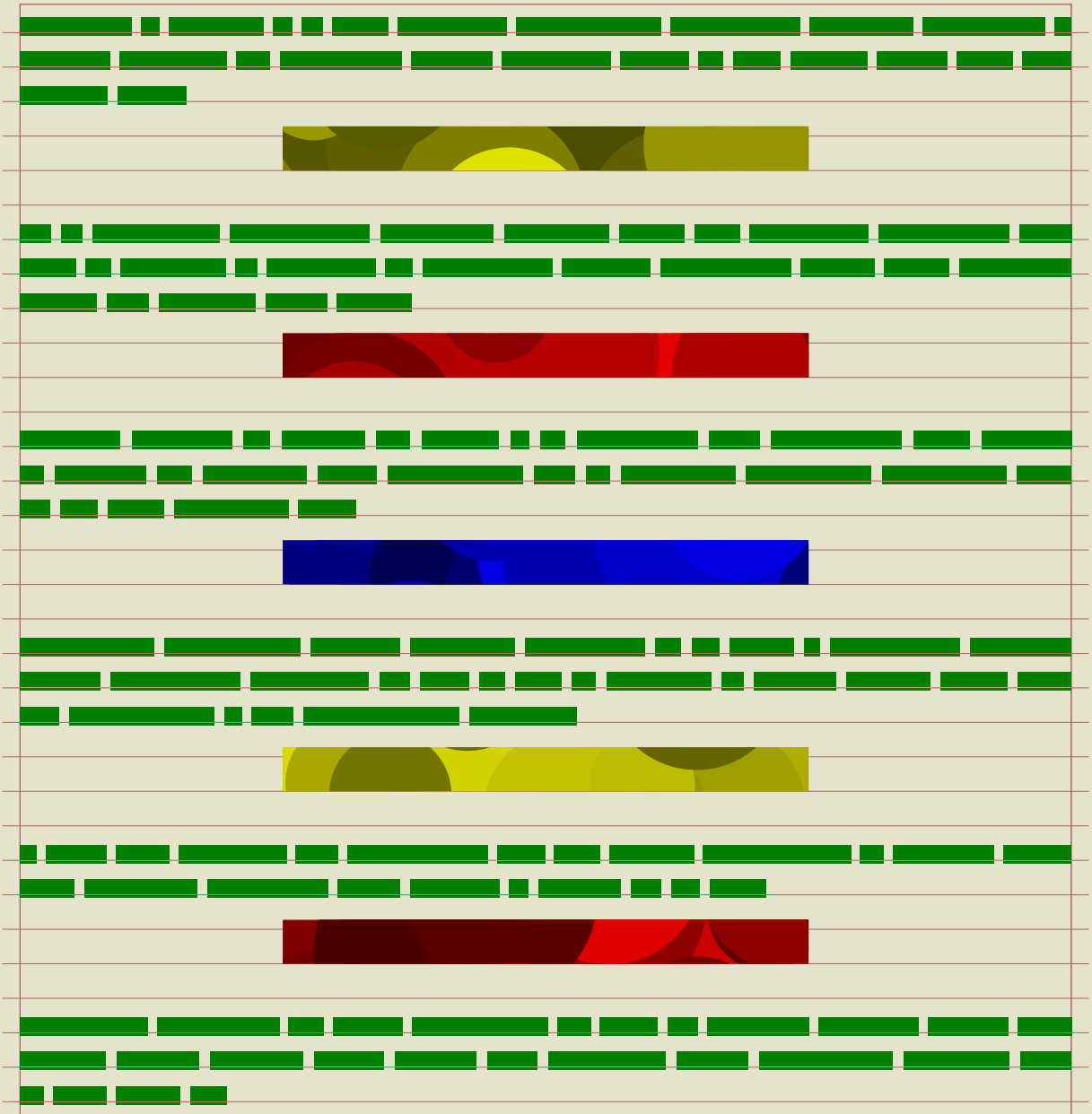
```
\placetightfigure{}{\externalfigure[dummy][lines=1.3,grid=height]}
\placetightfigure{}{\externalfigure[dummy][lines=1.4,grid=height]}
\placetightfigure{}{\externalfigure[dummy][lines=1.5,grid=height]}
\placetightfigure{}{\externalfigure[dummy][lines=1.6,grid=height]}
\placetightfigure{}{\externalfigure[dummy][lines=1.7,grid=height]}
```



```
\placetightfigure{}{\externalfigure[dummy][lines=1.3,grid=yes]}
\placetightfigure{}{\externalfigure[dummy][lines=1.4,grid=yes]}
\placetightfigure{}{\externalfigure[dummy][lines=1.5,grid=yes]}
\placetightfigure{}{\externalfigure[dummy][lines=1.6,grid=yes]}
\placetightfigure{}{\externalfigure[dummy][lines=1.7,grid=yes]}
```



```
\placetightfigure{}{\externalfigure[dummy][lines=1.3,grid=fit]}
\placetightfigure{}{\externalfigure[dummy][lines=1.4,grid=fit]}
\placetightfigure{}{\externalfigure[dummy][lines=1.5,grid=fit]}
\placetightfigure{}{\externalfigure[dummy][lines=1.6,grid=fit]}
\placetightfigure{}{\externalfigure[dummy][lines=1.7,grid=fit]}
```



```
\placetightfigure{}{\externalfigure[dummy][lines=1.3,grid=height]}
\placetightfigure{}{\externalfigure[dummy][lines=1.4,grid=height]}
\placetightfigure{}{\externalfigure[dummy][lines=1.5,grid=height]}
\placetightfigure{}{\externalfigure[dummy][lines=1.6,grid=height]}
\placetightfigure{}{\externalfigure[dummy][lines=1.7,grid=height]}
```

About this document

This document is typeset in ConTEXt using pdf-ε-TEX and METAPOST. We use only one font: the Computer Modern Typewriter. The verbatim portions of the text are typeset in its mono spaced variant. One of the reasons that I chose this font is that we need a mono spaced font to typeset the example code, and the Computer Modern Typewriter is one the best there is. This font combines well with many other typefaces, but the sometimes excessive use of different fonts (and sizes) in the styles that I have to implement made me long for simplicity. And so I decided to stick to one font. A careful reader will notice that this document has character protruding enabled (resulting in hanging punctuation).

We use a couple of colors. Again, I went for simplicity and use rather primary colors, although I do use them in transparent variants as well.

There is not much more to say, apart from that I want to thank our customers as well as ConTEXt users for asking me to implement dtp competing styles and features. Their demands drive ConTEXt in directions we could not have foreseen when we started its development.

We use a (transparent) gray background behind the text so that we have an indication where the text area is positioned relative to the page. It also enables us to comfortably turn on the grid.

Some features shown here are relatively new and therefore they occasionally are improved. As a result some aspects of their functionality may change.

CONTEXT