The PracTEX Journal, 2006, No. 2
Article revision 2006/05/04

Writing Your Own Presentation Style in ConTEXt

Thomas A. Schmitz

1. Introduction

If you use any TEX macro package for your work and have to give presentations, you have probably thought about ways of using TEX for these presentations as well. In principle, this is easy: you can produce a pdf file, and many pdf viewers have a fullscreen mode which allows you to show this pdf page by page, thus presenting a series of "slides."

Both LATEX and ConTEXt have dedicated packages and modules for this job: in LATEX there's the `prosper` or the `beamer` packages which allow to produce such pdf presentations (Michael Wiedmann has written an in-depth **comparison**[1] of different solutions). In ConTEXt, there's a number of predefined modules with very advanced visual effects; there's a **sample document**[2] on the pragma website (it's about 3 MB).

While this is wonderful, I found it extremely instructive to create my own style for presentations. The customizability of ConTEXt is so great that it is quite easy to do, and instead of relying on predefined styles and code you may not understand, you're in control of everything. This article will introduce a way of writing your own presentation style. And there will be one huge benefit: it will teach you how to have the manuscript for your lecture and the material for your slides in the same source file, allowing you to produce different output with just one switch.

The article does not presuppose prior knowledge of advanced features; it is suitable for beginners in ConTEXt.

[1] `http://me.in-berlin.de/~miwie/presentations/presentations.html`
[2] `http://www.pragma-ade.com/show-pre.pdf`

2. The Concept

Before we actually start writing our file, we just think about its structure and the ConTEXt tools that we will be using. What we want: we want one single source file which will output either a presentation that can be run in the full-screen mode of a pdf viewer or a manuscript for our lecture. There is one feature in ConTEXt which will allow us to achieve this aim; it's called "modes" (and it has its own **modes manual**[3]). It allows us, as it were, to have two documents within one source: certain parts of the source file are marked as belonging to one mode and are typeset only if this mode is explicitly turned on (there's a similar feature for LATEX; it's called "conditional text", and there's a section on it in the **UK TEX faq**[4], but as far as I can see, it's less advanced than ConTEXt modes). Modes are sort of a preprocessor command: when ConTEXt reads your source file (a process which is called "parsing") and is passed a mode, it will "see" only the parts that are marked as belonging to this mode. Two switches will be especially interesting for us:

```
\startmode[NAME]
...
\stopmode
```

Everything that goes between these commands is only effective if this mode is enabled. This can be commands and setups, definitions of macros, or simply text.

```
\startnotmode[NAME]
...
\stopnotmode
```

Easy to guess: everything between these commands is effective whenever the mode is *not* enabled.

Enabling a mode is easy. You can either have one line in your sourcefile

[3] http://www.pragma-ade.com/general/manuals/mmodes.pdf

[4] http://www.tex.ac.uk/cgi-bin/texfaq2html?label=conditional

```
\enablemode[NAME]
```

and either comment it out (with a % in front of it) or leave it active (remember that this line has to be placed *before* the definition of your modes!). Or you can pass `texexec`, the command-line tool for ConTEXt, the mode as an option:

```
texexec --mode=NAME myfile.tex
```

Now that we know which toolset we are going to use, let's go to the drawing board and plan how we will be using it to achieve our goal. Here is what I suggest:

1. We will define a mode "manuscript" which will contain the text for our lecture notes. This text will of course not be visible on the slides, so it will be wrapped in pairs of `\startmode[manuscript]` and `\stopmode`. The same is true for any setup and definition which we want to be effective for the manuscript only.

2. For the slides, on the other hand, we will have a particular set of setups which enable, say, a color theme, the right size for screen documents, and other bells and whistles. We could do this with `\startnotmode[manuscript]` and `\stopnotmode` commands, but I suggest defining a mode of its own, `\startmode[presentation]`.

3. I suggest that we include miniature versions of our "slides" in the text of our manuscript so we know what our audience is seeing without having to turn around to the screen. That means the text and graphics for our slides will be typeset for the slides themselves as well as for the manuscript.

3. Setups for the Slides

Let's first think about the slides; they need more special setup than the manuscript. So we start defining: we write `\startmode[presentation]` and use the ConTEXt commands to set things up.

3.1 Paper Size

ConTEXt has predefined "paper" sizes for screen documents (they are in landscape and match the usual ratio of computer screens). These sizes are defined and described in the ConTEXt module `page-lay.tex`; the width and height are roughly 4:3. The actual dimensions (defined in the ConTEXt module `page-lay.tex`) are:

```
[S3] [width=300pt,height=225pt]
[S4] [width=400pt,height=300pt]
[S5] [width=500pt,height=375pt]
[S6] [width=600pt,height=450pt]
[S8] [width=800pt,height=600pt]
[SW] [width=800pt,height=450pt]
[SM] [width=720pt,height=450pt]
```

S6 has approxiately the width of a sheet of A4 paper, so we write:

```
\setuppapersize[S6][S6]
```

3.2 Page Layout

Margins and space for headers/footers are a bit special for a screen document. Here's an example of the layout I use for my own slides:

```
\setuplayout [width=fit,
              rightmargin=1.5cm,
              leftmargin=1.5cm,
              leftmargindistance=0pt,
              rightmargindistance=0pt,
              height=fit,
              header=0pt,
              footer=5pt,
              topspace=2.5cm,
              backspace=1.5cm,
```

```
bottomspace=.8cm,
bottom=12pt,
location=singlesided]
```

I will not go into the gory details of page composition (for which you can consult chapter 3 of the **ConTEXt manual**[5]). Just a few words about the values above: the text area of our slides is calculated automatically; that's why we have `width=fit, height=fit`. This text body is located 1.5cm from the right and left edge (`backspace=1.5cm`), 2.5cm from the top (`topspace=2.5cm`) and 0.8cm from the bottom edge (`bottomspace=.8cm`). ConTEXt offers the possibility to place elements into this `backspace` area; if we want that (and we might!), we have to assign a percentage of this area to the "margin". In our case, we want to use the entire `backspace` left and right for such "marginal" material (`rightmargin=1.5cm,leftmargin=1.5cm`). We will not be using a header, but later, we will be typesetting elements in the bottom part of our slides; that's why we have (`footer=5pt,bottom=12pt`).

These values are merely empirical; I find that my presentations look good with them, but feel free to experiment.

3.3 Page Numbers and Colors

We'll want colors for our presentation slides, and we do not want pagenumbers appearing on them. So we type:

```
\setupcolors[state=start]
\setuppagenumber[state=stop]
```

Now we have to go for a color scheme for our slides. This is the moment to bring your personal style to your slides—but don't go too wild, your audience will be grateful. In classes I taught, I have made completely unrepresentative polls, and most people thought that white text on a dark blue background was most legible. So let's go for this look (again, experiment to see if you like other combinations better):

[5] `http://www.pragma-ade.com/general/manuals/cont-eni.pdf`

```
\setupbackgrounds[page][background=color,backgroundcolor=darkblue]
\startcolor[white]
```

3.4 Defining the Slides

This was the general structure for our documents. We now define the specific look of our slides. We want a "normal" bodyfont of a size of about 24 pt (just how big you want your bodyfont will of course depend on a number of factors: the font you are using, the amount of text you want on your slides, the size of the screen, etc.). In order to have a uniform look for our slides, we will put this into an environment defined by a `\start...\stop` pair. Of course, every slide will be on a "page" of its own; that's what the command `before=\page` does.

```
\definestartstop
   [Slide]
   [commands={\switchtobodyfont[24pt]},
    before=\page]
```

Finally, we want (almost) every slide to start with a centered title in a somewhat larger font and with a blank line after it. So we define a command that will do just that:

```
\define[1]\SlideTitle
  {\midaligned{\tfb #1}
   \blank[line]}
```

We now close the setups for our slides by typing `\stopmode`

4. Setups for the Manuscript

Everything we have done so far will merely modify the look of our slides; for the manuscript, we have left everything at its default settings (which are perfectly alright).

There's only one thing that is absolutely necessary to have: we have defined an environment `\startSlide` and a command `\SlideTitle`, but only for the presentation mode. Since we want the material of the slides to be typeset in the manuscript as well, we must define both for manuscript mode as well, or TEX will complain about "undefined control sequences." In manuscript mode, we don't want `\startSlide` to start a new page and to change to large font sizes. One way to incorporate the material on the slides into your manuscript would be to have it in a little frame in the middle of the page, with an empty line preceding and following it. ConTEXt offers an environment to do this, `\startframedtext`. So all we have to do is set this up and make our own environment `\startSlide` call this command:[6]

```
\startmode[manuscript]
\setupframedtexts[location=middle,
                  before={\blank[line]},
                  after={\blank[line]}]

\let\startSlide\startframedtext
\let\stopSlide\stopframedtext
```

Moreover, we need to redefine the command `\SlideTitle`. In our manuscript, we just want the title to be centered and followed by a small blank space. So we define:

```
\define[1]\SlideTitle{\midaligned{#1}\blank[small]}
```

This is the basic setup. The line `\stopmode` finishes the setup for our manuscript mode.

5. Making it Work

Instead of including a lengthy example in this article, I append a **ConTEXt file** which will give you some ideas. Of course, you can use all the concepts ConTEXt offers on

[6] For the curious: `\let\a\b` "clones" a control sequence `\b` to a new name `\a`; you can find an in-depth explanation in the *TEXbook*, p. 206–7.

these slides: environments, lists, graphics, figures, math, just about anything. Two pdf files are obtained from the same source; one has been compiled in **manuscript mode**, the other in **presentation mode**. If you have a ConTEXt-installation on your computer, you can try it yourself: just compile the attached file. The first line reads `\enablemode[presentation]`; if you change it to `\enablemode[manuscript]`, you will typeset the manuscript for an imaginary lecture, with the slides included as small framed images. Play with the file and test things on the slides.

6. More Bells and Whistles

The slides you get from compiling our example file are very basic; they are just meant as a first encouragement to get you started. If you want your slides to look a bit more refined, here are some ideas:

6.1 Fonts

ConTEXt uses the Latin Modern font by default; this is a replica of Knuth's original Computer Modern typeface (see **Will Robertson's article**[7]). While it is a marvelous font, I find it's not quite suitable for a screen presentation. There's no problem enabling a different font for your slides. One possibility would be to go for a sans serif font like Helvetica. In that case, just put these two lines into the `\startmode[presentation]` section:

```
\usetypescript[helvetica] [ec]
\setupbodyfont[helvetica,ss,24pt]
```

and your slides will be in Helvetica (while your manuscript will be unchanged). A font that I find very legible on such slides is Palatino. This would be enabled with these two lines:

[7] `http://www.tug.org/pracjourn/2006-1/robertson/robertson.pdf`

```
\usetypescript[palatino] [ec]
\setupbodyfont[palatino,24pt] % or maybe 20pt, Palatino is big!
```

You can learn more about the free fonts that are probably already available in your TEX installation at the **ConTEXt wiki**[8].

6.2 Backgrounds and Colors

Our blue background is certainly nice, but with ConTEXt, we can do better. You can define almost anything as a background to your slides: solid colors, a picture, a graphics file. Something I find very impressive is a background with two colors that are interpolated, i.e., fade into each other. This interpolation can be achieved with MetaFun, a macro package for METAPOST that enhances the graphics capabilities of ConTEXt. So instead of just setting the background to `darkblue`, we put some MetaFun code into the background of our slides:

```
\definecolor[lightblue][r=0,g=0,b=1]
\definecolor[darkblue][r=0,g=0,b=0.05]

\startuniqueMPgraphic{LinearShade}
path p ;
p := unitsquare xscaled \overlaywidth yscaled \overlayheight ;
linear_shade(p,6,\MPcolor{darkblue},\MPcolor{lightblue}) ;
\stopuniqueMPgraphic

\defineoverlay[shaded][\useMPgraphic{LinearShade}]

\setupbackgrounds[page][background={shaded}]
```

We have defined two colors; `lightblue` is a very light, `darkblue` a very dark blue. We let MetaFun calculate an interpolation between both colors. If you want to know more about the details of how this works, you can read section 8.1 in the **MetaFun manual**[9],

[8] `http://wiki.contextgarden.net/Psnfss`

or you can just experiment with other values for the colors or for the interpolation (try all numbers from 0 to 9 for the `p,6` variable).[10]

You can have even fancier effects; have a look at the attached file `example2.tex` to see what can be done!

6.3 Pictures

ConTEXt's figures mechanism makes it easy to integrate pictures into your slides. There's a number of points we have to consider:

1. ConTEXt relies on pdfetex, so the supported formats (in addition to embedded metapost code) are `.pdf, .png, .jpg`.[11] So if you have a picture in a different format, you will need to convert it.

2. You can either have all the pictures in the same directory as your source file, and TEX will find them automatically, or you can name a path explicitly via the command `\setupexternalfigures[directory=/PATH/TO/PICTURES]`.

3. Our pictures will be shown on the slides as well as in the manuscript. Since these modes have different setups in terms of textwidth and location of slides, we will have to make two definitions for the dimensions of the picture, in `\startmode ... \stopmode` environments.

How do you want to present your graphics on your slides? This will, of course, depend on a number of factors: the nature of the pictures you want to show, their size, and the relation between graphics and surrounding text. Here are some possibilities:

---

[9] `http://www.pragma-ade.com/general/manuals/metafun-s.pdf`

[10] If you want this metapost code to be executed as you typeset your document, you may need to adjust two configuration files in order for this to work: in `texmf.cnf`, you have to have the line `shell_escape = t`; in your `cont-sys.tex`, uncoment the lines `\runMPgraphicstrue` and `\runMPTEXgraphicstrue`. If you don't you will have to run `texexec` twice to see the results.

[11] A number of documents claims that `.tif` is supported as well, but this is not the case for current versions of pdfetex.

1. If you want your picture to take an entire "paragraph" of your slide, without any additional material next to it, you can simply place it via the command `\externalfigure[NAME][height=.75\textheight]`.

2. It's a bit more complicated to have two pictures or text and pictures side by side. One way to achieve this is via the `\startcombination` and `\stopcombination` environment.[12] It allows you to have a picture and text or another picture on two "columns" on your slide. Instead of a lengthy explanation, see the example in the attached file.

3. Or, the `\setlayer` command allows you to place pictures at arbitrary positions on your slides, independent of surrounding text. Layers are a powerful, yet complex feature of ConTEXt; they are documented in chapter 6 of the **details manual**[13].

It's difficult to give any general receipe; everything depends on your needs and your individual workflow. Have a look at the attached files to get some inspiration of what is possible.

6.4 Progress Meter

One feature I find very nice is a predefined module in ConTEXt. It provides, at the bottom of your "slides," a progress meter that shows how many slides have already been shown and how many more will follow. The feature is called "interactionbar." In order to get it, just put this in the definition of the "slides" mode:

```
\setupinteraction
  [page=yes,
   color=InteractionColor,
   contrastcolor=ContrastColor,
   menu=on,
   state=start]
```

[12] If you want to know more about this environment, consult section 12.3 in the ConTEXt manual.

[13] `http://www.pragma-ade.com/general/manuals/details.pdf`

```
\startinteractionmenu[bottom]
\rightaligned{\interactionbar[alternative=f,width=\makeupwidth,height=1ex]}
\stopinteractionmenu
```

If you want to have such an "interactionbar," you will have to modify the definition of our `\startSlide`: the `\start ... \stop` defines a grouped command and prevents ConTEXt from "seeing" the page breaks. We now define collected setups for the commands `\startSlide` and `\stopSlide`:[14]

```
\def\startSlide{\directsetup{slide:start}}
\def\stopSlide {\directsetup{slide:stop}}

\startsetups slide:start
 \start
 \switchtobodyfont[24pt]
 \setupinteractionbar[state=start]
\stopsetups
\startsetups slide:stop
 \page
 \egroup \stopsetups
```

This will put a progress meter on every slide. I have attached an example with these fancier settings, both in the **source** and the compiled **output**.

6.5 Handout

Now let's suppose that after typesetting your document, you have the idea of giving the material of your slides to your audience as a handout. That's easy to do: you just have to define another mode; let's call it `handout`. What we want: typesetting the text and the graphics on the slides, but without the colored background. Every slide will appear in a small frame, and there will be two columns on every A4 (or letter) page.

[14] Hans Hagen has contributed this code; for this and for his tireless efforts in developing ConTEXt, I am extremely grateful.

Here's the setup:

```
\startmode[handout]
\setuppapersize[A4][A4]
\setuplayout [width=fit,
              rightmargin=0cm,
              leftmargin=0cm,
              backspace=1.4cm]
\setupframedtexts[location=left,
                  width=\textwidth,
                  height=.75\textwidth,
                  before={\blank[line]},
                  after={\blank[line]}]
\let\startSlide\startframedtext
\let\stopSlide\stopframedtext
\define[1]\SlideTitle{\midaligned{#1}\blank[small]}
\usetypescript[palatino] [ec]
\setupbodyfont[palatino,11pt]
\setupcolumns[2,distance=2mm]
\stopmode
```

We can be brief in describing this since you've encountered most of the commands already: we use A4 paper, and we set the margins pretty narrow since every frame carries its own margin. The framed "slides" have a blank line before and after the frame. For our manuscript, we simply let the content of the embedded slides determine the height; for the handout, we give a fixed height which is ¾ of the slide's width (`height=.75\textwidth`). `\startSlide` just triggers the frame. The command `\SlideTitle` is defined to center the title and leave a small blank space after it, and we choose a bodyfont of 11 pt. Finally, we define two columns with a space of only 2 mm between them (again, because every "slide" has its own margins). After setting this up, we have to put this at the beginning of the document, just after the line that says `\starttext`

```
\startmode[handout]
\startcolumns
\stopmode
```

to start setting the handout in two columns. And of course, you will need to end this columnset at the end of the document:

```
\startmode[handout]
\stopcolumns
\stopmode
```

Now typesetting a handout of your slides is as easy as modifying the first line of our examples to `\enablemode[handout]`. Again, you'll find an example attached to this article: it is derived from the same source file, compiled in **handout mode**.

7. Conclusion

Of course, an almost infinite number of optical improvements is possible; you really should have a look at the predefined presentation styles and let this inspire you. But the method described here should allow you to understand the mechanism behind it all and to develop your own personal style. And it should demonstrate how powerful the mechanism behind "modes" in ConTEXt is; it allows for almost unlimited possibilities. You will certainly find more use for it now that you've seen what it can do for you.