Preparing Slide Presentations with the PPTEX Package

Ulrich Bodenhofer (*ulrich@bodenhofer.com*)

Contents

1 Introduction

In 2000, the author developed a LaTeX document class for his former employer, Software Competence Center Hagenberg, with the aim to emulate the corporate design of this institution's MS PowerPoint template. The given package—`pptex.cls`—is a new version and generalization of this package (that was called `scch-slides.cls` then). The main difference between the former class `scch-slides.cls` and `pptex.cls` is that different layouts, including the SCCH layout, are possible. The set of available layouts is extensible. Moreover, `pptex.cls` now supports a 4:3 page/slide format of 10×7.5in by default (the same as MS PowerPoint), whereas `scch-slides.cls` was only able to use A4 format.

The PPTeX slide document class can be used in the following way:

`\documentclass[`⟨*options*⟩`]{pptex}`

The following options are available:

`display` enables animation features for beamer presentations

`printout` turns off all animation features (recommended for handouts, printed slides, etc.)

`startblank` adds a blank black slide as first page

`endblank` adds a blank black slide as last page

`a4paper` switches back to A4 paper (only for backwards compatibility; not recommended)

Moreover, the name of a layout has to be passed as an option as well. If no layout is specified, LaTeX displays an error message.

NOTE The whole class is designed for use with PDFLaTeX only. Any attempt to process a PPTeX document with ordinary LaTeX results in an error message.

2 Layouts

As mentioned above, a layout has to be specified as option in the `\documentclass` command. Presently, three layouts are available:

`jku` layout resembling the new corporate design of Johannes Kepler University Linz (JKU, 2015)

`bioinf` layout in the corporate design of the Institute of Bioinformatics, Johannes Kepler University Linz

`jkuold` layout resembling the old corporate design of Johannes Kepler University Linz (JKU, around 2003)

`scch` layout corresponding to the MS PowerPoint template of Software Competence Center Hagenberg (SCCH, as of 2006)

`scchh` the same as `scchh`, but without using the Verdana font

`scchold` obsolete layout corresponding to the previous MS PowerPoint template of Software Competence Center Hagenberg (SCCH)

`uda` layout for presentations related to Austrian Partner of International Universities (UDA)

`udaold` obsolete layout for presentations related to previous Unversity of Derby in Austria (UDA)

The layouts `jkuold`, `scchold` and `udaold` are available for 4:3 format (default) and A4 format (when the `a4paper` option is being used) in order to provide a compatibility mode for older presentations. The layouts `jku`, `bioinf`, `scch`, `scchh` and `uda` are newer ones and have been developed when PPTEX was already available, so backwards compatibility is not an issue for these layouts.

It is possible to define new layouts, but it is not yet documented how to do this. If you are interested in writing a new layout, please contact the author of the package.

3 Basic Usage

The `pptex` class is based on the document class `seminar` by Timothy Van Zandt and makes use of the TEXPower bundle written by Stephan Lehmke.

Similar to the classes `seminar` and `slides`, slide text must be enclosed in the `slide` environment. Internally, the `pptex` class works with `picture` environments. In the following, we will discuss the convenience functions that have been implemented in order to make work with these picture environments easier.

`\titleslide[`⟨*author*⟩`]{`⟨*title*⟩`}{`⟨*subtitle*⟩`}`

Produces a title slide, where the meaning of the arguments should be self-explanatory. Please note that the optional argument is not used by all layouts (cf. 4.1).

`\simpletextslide{`⟨*headline*⟩`}{`⟨*text*⟩`}`

Produces a slide with one single large text box. The second argument may contain any valid LATEX text (including mathematical formulas, the environments `itemize`, `enumerate`, `description`, `list`, `verse`, `quote`, `quotation`, `verbatim`, `center`, `flushleft`, `flushright`, and many more).

Here is a simple example how to use `\titleslide` and `\simpletextslide`:

```
\documentclass[display,jku]{pptex}

\setfootline{Ulrich Bodenhofer}

\begin{document}

\begin{slide}
\titleslide{PP\TeX}{% title
A Short Introduction} % subtitle
\end{slide}

\begin{slide}
\simpletextslide{Overview}{%
\begin{itemize}
\item Motivation
\item Basic usage
\item Using layouts
\item Miscellaneous issues
\end{itemize}}
```

```
\end{slide}

\end{document}
```

`\simpleslide{`⟨*headline*⟩`}{`⟨*body*⟩`}`

Produces an empty slide. The argument ⟨*body*⟩ may only contain `\put` commands, since the surrounding environment is a `picture` environment. However, the `pptex` class contains several convenience macros for several often used structures which will be described next.

3.1 Convenience Macros for Positioning Text and Pictures Freely

The following macros may only be used inside the ⟨*body*⟩ argument of the `\simpleslide` command.

`\textbox[`⟨*align*⟩`]{`⟨*xpos,ypos*⟩`}{`⟨*w*⟩`}{`⟨*text*⟩`}`

This command puts a text box of width ⟨*w*⟩ with content ⟨*text*⟩ at position (⟨*xpos,ypos*⟩). Normally, the text box is placed such that its left upper corner is at position (⟨*xpos,ypos*⟩). This can be overridden with the optional argument ⟨*align*⟩, the effect of which is the same as of the optional argument of the `\makebox` command inside `picture` environments:

`lt` **(default):** upper left corner at (⟨*xpos,ypos*⟩)

`l`**:** middle of left edge at (⟨*xpos,ypos*⟩)

`lb`**:** lower left corner at (⟨*xpos,ypos*⟩)

`rt`**:** upper right corner at (⟨*xpos,ypos*⟩)

`r`**:** middle of right edge at (⟨*xpos,ypos*⟩)

`rb`**:** lower right corner at (⟨*xpos,ypos*⟩)

`t`**:** middle of top edge at (⟨*xpos,ypos*⟩)

`b`**:** middle of bottom edge at (⟨*xpos,ypos*⟩)

Note that the coordinates (⟨*xpos,ypos*⟩) need to be scalars (no units!). The internal unit of measure is cm relative to the lower left corner of the slide (total size 10×7.5 in $= 25.4\times19.05$ cm; 29.7×21cm if the option `a4paper` is used); analogous for the width ⟨*w*⟩. Concerning the argument ⟨*text*⟩, the same rules as for the second argument of the `\simpletextslide` command apply (see above).

`\itembox[`⟨*align*⟩`]{`⟨*xpos,ypos*⟩`}{`⟨*w*⟩`}{`⟨*items*⟩`}`
`\enumbox[`⟨*align*⟩`]{`⟨*xpos,ypos*⟩`}{`⟨*w*⟩`}{`⟨*items*⟩`}`

These two convenience functions are basically the same as `\textbox` except that they already contain `itemize` and `enumerate` environments, respectively.

`\clearbox[`⟨*align*⟩`]{`⟨*xpos,ypos*⟩`}{`⟨*content*⟩`}`

Unlike the above three macros, this command allows to create a simple box without any internal pre-assumption about its content. This command is mainly useful for inserting graphics.

Examples for most of these macros can be found in the enclosed demo file `slidedemo.tex`.

3.2 Further Convenience Macros for Positioning Text and Pictures

Though the macros in the previous subsection are very flexible, they are not very user-friendly. In order to provide the user with some simpler, but more flexible convenience functions, the layouts `jku` and `bioinf` provide the following macro:

`\defaulttextbox{`⟨*text*⟩`}`

This macro, in conjunction with `\simpleslide`, is equivalent to the functionality of `\simpletextslide`. However, now the entire box can be animated and complemented by additional boxes (e.g. with additional text or figures).

For two-column layouts, the following two macros are available:

`\defaulttextboxleft{`⟨*text*⟩`}`
`\defaulttextboxright{`⟨*text*⟩`}`

3.3 The Foot Line

`\setfootline{`⟨*footline*⟩`}`

Determines the content of foot line. This command can be supplied in the preamble of the document or inside the document whenever a change of the content of the foot line is desired.

3.4 Lengths

Since the `seminar` class works with magnifications, an alternative model of lengths is used internally. Therefore, unfortunately, the usual font-independent LaTeX lengths (units `cm`, `mm`, and `in`) cannot be used in the normal way. Instead, the macros `\semcm` and `\semin` need to be used in order to obtain the desired results. Fortunately, no restrictions apply to the font-dependent units `em` and `ex`. NOTE

3.5 Graphics Inclusion

Graphics inclusion works as usual. However, the user should take the following two points into account:

1. Since the `scch-slides` class is designed for use with PDFLaTeX, only JPEG, PDF, and PNG files can be included, but no encapsulated PostScript (EPS) files. EPS files must converted to PDF first. This can be done by the GhostScript-based tool `epstopdf` which is available on the SCCH common software server.

2. As described in 3.4, font-independent lengths cannot be used in a straightforward way. Use the macros `\semcm` and `\semin` instead!

It is recommended to place pictures inside `\clearbox` environments, although the other box commands and `\simpletextslide` also work. A simple example:

```
\clearbox{3,14}{\includegraphics[width=7\semcm]{dummy.jpg}}
```

3.6 Animation

The `pptex` class makes use of the TEXPower bundle's animation features. Consequently, the reader is referred to the enclosed manual for more details. Be aware, however, that the `pptex` class makes
NOTE excessive use of the `picture` environment. Therefore, the `\pause` command does not work. Instead, the constructs `\stepwise`, `\step`, and all the related macros must be used in order to make animation work properly. More information is available in the enclosed TEXPower manual and the example files.

3.7 Hyperlinks

The package `hyperref` is included by default. Therefore, all its features can be used in PPTEX slide presentations without any restrictions. See the `hyperref` manual for more details.

3.8 Packages

The following packages are included by `pptex.cls` and need not be included in the preamble of the document:

1. `color.sty`
2. `graphicx.sty`
3. `hyperref.sty`
4. `url.sty`
5. `texpower.sty`

The `pptex` document class should work properly with most standard packages, in particular those belonging to the $\mathcal{AMS}$LATEX bundle.

4 Layout-Dependent Behavior

While PPTEX tries to maintain layout independence where possible, there are some functions/issues that depend on the layout that is actually being used.

4.1 Optional argument of `titleslide` command

In the layouts `jku`, `bioinf`, `uda` and `udaold`, the `titleslide` command ignores the optional argu-
NOTE ment, whereas the `scch` layout uses it to typeset the typical four-line author box to the right of the logo.

4.2 Page formats

Although the page/slide size is the same independent of the layout, the actually available space for text and drawings may be different for different layouts. The reason is simple that logos and graphical elements occupy differently much space. So it is almost sure that a PPTEX document that has been
NOTE written using the above documented commands compiles well with different layouts. It is likely, however, that adaptations of positions of text and graphics or adaptations of font sizes are necessary if the layout is changed.

4.3 Additional fonts

The layout `jku` requires the font Arial Black and the layout `scch` requires Verdana to work correctly. The

4.4 Colors

Different layouts may define different colors. These colors can be used as usual with the commands `\color`, `\textcolor`, `\colorbox`, and `\fcolorbox` (see documentation on the `color.sty` package for more information). The layouts may also define different convenience macros for using the colors.

4.4.1 Layout `jku`

Internally, the colors `jkublue`, `jkupurple`, `jkured`, `jkuyellow`, `jkugreen`, and `jkulightblue` are pre-defined according to the new JKU Corporate Design (2015).

The following commands can be used to switch to the colors for highlighting. They can be used in the same way as `\bf`:

```
\blue
\purple
\red
\yellow
\green
\lightblue
```

4.4.2 Layout `bioinf`

Internally, the colors `bioaz` (dark blue of the square in the logo), `biove` (dark green, like the darker ends of the DNA strands in the logo), `bioli` (light green, like the lighter ends of the DNA strands in the logo), `biowh` (white), and `biobl` (black) are pre-defined.

```
\green
\blue
```

These commands can be used to switch to the colors for highlighting. They can be used in the same way as `\bf`.

4.4.3 Layout `jkuold`

Internally, the colors `jkure` (ruby red), `jkugr` (medium gray), `jkulg` (light gray), `jkuwh` (white), and `jkubl` (black) are pre-defined.

```
\red
```

This command can be used to switch to the colors for highlighting. It can be used in the same way as `\bf`.

4.4.4 Layout `scch`

Internally, the colors `scchre` (ruby red), `scchlg` (light gray), `scchdg` (dark gray), `scchwh` (white), and `scchbl` (black) are pre-defined, corresponding to the standard colors of the new SCCH corporate identity.

`\red`

This command can be used to switch to the color that is standard for highlighting in the corresponding PowerPoint template. It can be used like a font switch (e.g. `\bf`).

`\scchcopyright`

This command prints the SCCH copyright notice "© Software Competence Center Hagenberg GmbH". Note that this is also the default foot line.

NOTE The standard SCCH PowerPoint layout uses Verdana as default font for title and subtitle on title pages and for slide headings. The SCCH layout cannot be used if this font is not available to PDFLaTeX. For instructions how to make Verdana usable for PDFLaTeX, see Appendix A.

4.4.5 Layout `scchh`

Works in the same way as the layout `scch` with the difference that it does *not* use Verdana. It uses Helvetica instead, so files using the `scch` layout should run smoothly on every up-to-date LaTeX installation, regardless of whether Verdana is available or not.

4.4.6 Layout `oldscch`

Internally, the colors `scchre` (ruby red), `scchye` ("Österreichische Post" yellow), `scchgr` (medium gray), `scchwh` (white), and `scchbl` (black) are pre-defined, corresponding to the standard colors of the previous SCCH corporate identity.

`\red`
`\yellow`

These commands can be used to switch to the colors which are standard for highlighting in the corresponding PowerPoint template. They can be used like font switches (e.g. `\bf`).

4.4.7 Layout `uda`

The colors `udaaz` (dark blue), `udagr` (medium gray), `udabg` (very light steel blue), and `udabl` (black) are pre-defined.

`\blue`
`\grey`

As above, these commands can be used to switch to the colors for highlighting. It can be used in the same way as `\bf`.

4.4.8 Layout `udaold`

The colors `udati` (similar to moss-green), `udagr` (medium gray), `udalg` (light gray), `udawh` (white), and `udabl` (black) are pre-defined.

`\green`

As above, this command can be used to switch to the colors for highlighting. It can be used in the same way as `\bf`.

4.5 Layout-specific convenience macros

In the `scch-slides` package, the macros described above had different names, all starting with `\scch`. In order to be independent of the chosen layout, this has been changed in PPTEX. However, the three layouts still define all these macros in order to maintain backwards compatibility. A presentation that was written using the `scch-slides` class should still produce the same result if compiled with `\documentclass[scch,a4paper]{pptex}`.

5 Further Caveats

1. Animation is a delicate thing; if some arrangement of `\step` statements does not work, try different ones until the document is processed without error message. In particular, the first `\item` of a list environment (including `itemize`, `enumerate`, and `description`) may not be surrounded by a `\step` statement.

2. Slides in portrait format (corresponding to the `slides*` environment in the `seminar` class) are yet supported.

3. When processing a PPTEX document, a lot of garbage warnings and other messages are produced.

4. The status of the TEXPower bundle is still experimental, and so is the status of the `pptex` document class.

5. The use of more than one layout within one presentation is currently not supported.

6. Note that you cannot use optional arguments in optional arguments; bear that in mind when you use the optional argument of `\titleslide`.

6 Related Documents

This document, more or less, only describes the specific features of the class `pptex.cls` , which makes excessive use of other packages and classes, such as `seminar.cls`, `texpower.sty`, and `hyperref.sty`. For detailed descriptions, see the enclosed manuals.

7 License Issues

All files included in this distribution that were written by Ulrich Bodenhofer are completely free. They may be downloaded, used, copied, distributed, modified, without any restrictions and without the prior acceptance of the author.

However, this distribution also contains material from other authors, in particular, version 0.0.8f of Stephan Lehmke's TeXPower package. For these files, license restrictions may apply. Please check the headers of these files for further information.

A Making TrueType Fonts Usable for LaTeX and PDFLaTeX

In this appendix, we give instructions on how Verdana can be made available to LaTeX/PDFLaTeX. These instructions are based on the procedure described by Jens Weißenburg,[1] an update (and German translation) of the method previously described by Damir Rakityansky.[2] This procedure has worked perfectly on the author's system and has *not* been tested on other systems. So the author does, in no sense, guarantee that the procedure described in the following will work on other systems.

Prerequisites

Your system needs TeX tools for handling TrueType fonts, such as, `ttf2tfm`, `ttf2pk`, etc. These programs are included in recent versions of MikTeX and also in distributions for other platforms. Further make sure that the file `T1-WGL4.enc` is found on your system (search in the `ttf2tfm` subfolder of your (local) `texmf` directory. If it is not there, download it from the Internet.

Creating the font-specific files

Start from an empty folder and copy the relevant TTF files to this folder. For Verdana, these are the files `verdana.ttf` (regular), `verdanab.ttf` (bold), `verdanai.ttf` (italic), and `verdanaz.ttf` (bold italic). For Arial Black, this is the file `ariblk.ttf`. Open a command/shell window and change to this directory containing the copy/copies of the TrueType file(s).

For producing the Verdana files, enter the following 12 commands (or put them into a shell script/batch program):

```
ttf2tfm verdana.ttf -q -T T1-WGL4.enc -v ecvrd.vpl recvrd.tfm >ttfonts.map
ttf2tfm verdanab.ttf -q -T T1-WGL4.enc -v ecvrdb.vpl recvrdb.tfm >>ttfonts.map
ttf2tfm verdanai.ttf -q -T T1-WGL4.enc -v ecvrdi.vpl recvrdi.tfm >>ttfonts.map
ttf2tfm verdanaz.ttf -q -T T1-WGL4.enc -v ecvrdbi.vpl recvrdbi.tfm >>ttfonts.map
ttf2tfm verdana.ttf -q -T T1-WGL4.enc -s .167 -v ecvrdo.vpl recvrdo.tfm >>ttfonts.map
ttf2tfm verdanab.ttf -q -T T1-WGL4.enc -s .167 -v ecvrdbo.vpl recvrdbo.tfm >>ttfonts.map

vptovf ecvrd.vpl ecvrd.vf ecvrd.tfm
vptovf ecvrdb.vpl ecvrdb.vf ecvrdb.tfm
vptovf ecvrdi.vpl ecvrdi.vf ecvrdi.tfm
vptovf ecvrdbi.vpl ecvrdbi.vf ecvrdbi.tfm
vptovf ecvrdo.vpl ecvrdo.vf ecvrdo.tfm
vptovf ecvrdbo.vpl ecvrdbo.vf ecvrdbo.tfm
```

[1] `http://www.weissenburger.de/content/latex5/`

[2] `http://www.radamir.com/tex/`

For producing the Arial Black files, enter the following commands (or put them into a shell script/batch program):

```
ttf2tfm ariblk.ttf -q -T T1-WGL4.enc -v ecarb.vpl recarb.tfm >ttfonts.map
vptovf ecarb.vpl ecarb.vf ecarb.tfm
```

Note that it is essential that you use exactly the same file names for the destination files, otherwise the font definitions in the files `t1tvrd.fd` and `t1tarb.fd` that are supplied with PPTEX will not work. NOTE

After having done this, you can delete all files ending with `.vpl`.

Installing the files such that TEX finds them

1. Place all `.tfm` files in a folder, where TEX searches for `.tfm` files; recommended: create a sub-folder of `fonts/tfm/` in your `localtexmf` directory and move the files there.
2. Place the `.vf` files in a folder, where TEX searches for `.vf` files; recommended: create a sub-folder of `fonts/vf/` in your `localtexmf` directory and move the files there.
3. Place the `.ttf` files in a folder, where `ttf2pk` can find them. I do not know enough about this, but at least for me `localtexmf/fonts/ttf` worked.
4. Append the lines of the above created `ttfonts.map` file to your system-wide `ttfonts.map` file (most probably in the `ttf2tfm/base` folder in your `texmf` directory)
5. Once again, make sure that the file `T1-WGL4.enc` is correctly found on your system.
6. Refresh the file name database.

Caveats

- The fonts' resolution may not be too good. If you zoom into the resulting PDFs, you will see that they are actually quite edgy. For screen/beamer presentations, however, the anti-aliasing capabilities of up-to-date PDF viewers are sufficient not to make the low resolution appear too annoying.
- As you may have seen from the output of the `ttf2tfm` commands above, not all special characters can be translated. In particular, some ligatures (e.g. ff vs. ff) are not resolved and are treated as single characters by the new font.