The PracTEX Journal
TPJ 2005 No 02, 2005-04-15
Rev. 2005-05-07

Producing beautiful slides with LATEX: An introduction to the HA-prosper package

Tristan Miller
German Research Center for Artificial Intelligence
Postfach 20 80, 67608 Kaiserslautern, Germany
`http://www.dfki.uni-kl.de/~miller/`
`Tristan.Miller@dfki.de`

Abstract

In this paper, we present HA-prosper, a LATEX package for creating overhead slides. We describe the features of the package and give examples of their use. We also discuss what advantages there are to producing slides with LATEX versus the presentation software typically bundled with today's office suites.

1 Introduction

Creating attractive overhead presentations has long been the domain of proprietary software packages such as Harvard Graphics and Microsoft PowerPoint. Though this trend has been changing in recent years with the advent of free[1] presentation programs such as OpenOffice.org Impress and and KPresenter, from a LATEX user's point of view these packages still have a number of shortcomings which make them inadequate or inappropriate for many purposes. Though presentation packages for LATEX have been available since 1993, it was not until relatively recently that these packages have matured to the point where a novice user

[1] "Free" here is used in the sense of the user being allowed to run, copy, distribute, study, change, and improve the software.

could use them to produce layouts matching those of their proprietary counterparts.

In this article we introduce HA-prosper[2], one such LaTeX package for producing overheard slides.[3] Among its many features are overlays, animated slide transitions, an automatically generated table of contents, split slides, hidden author annotations, and internal and external hyperlinks. It can output to PDF for online viewing or to portrait- or landscape-orientation PostScript for printed transparencies. As with any LaTeX class, the layout parameters are near-infinitely configurable, but helpfully a number of prefabricated styles and templates are available.

This article assumes that you have a basic familiarity with editing and compiling LaTeX documents. It also assumes that you either have the requisite packages for HA-prosper (see Appendix A) already installed on your system, or know how to fetch them from CTAN[4] and install them yourself.

2 Why use LaTeX for slides?

The motivation for using a text-based document processor to produce slides may not be readily apparent to those who normally use WYSIWYG presentation software. However, there are a number of important benefits to using LaTeX for presentations, even for those who do not normally use it for writing papers. Many of these benefits are typical of LaTeX in general and will be already familiar to critics of the word processor paradigm (see *e. g.*, [2, 1]). However, for completeness's sake we review them here briefly.

Portability. Most LaTeX slide packages, including HA-prosper, are designed to produce PostScript or PDF files. These formats are standard in the computing industry and viewers for them are widely available for all common platforms. This means, for example, that it's easy to start writing a presentation on your office Unix machine and then take it home to finish and preview on your MS-Windows PC. More

[2]HA-prosper was written by Hendri Adriaens of Tilburg University; it is an extension to Frédéric Goualard and Peter Møller Neergaard's prosper class, which is itself based on the venerable seminar class by Timothy van Zandt.

[3]There exist many other screen presentation tools for LaTeX and friends; see [3] for a near-comprehensive list.

[4]`http://www.ctan.org/`

importantly, having a PDF version of your presentation means that there is no need to arrange for your venue to have PowerPoint, StarOffice, or other uncommon or expensive software; nor do you need to bring along your own laptop with custom software. Most computers nowadays have some sort of PDF viewer installed, but even if not, there are several zero-cost viewers available. In most cases, you can simply copy your PDF presentation to a floppy, CD, or USB keydrive, slip it in your pocket, and head to your presentation venue.

Ease of collaboration. Because LaTeX files are plain text, it is easy for multiple authors to collaborate to write a presentation, even if not all of them have LaTeX. Furthermore, keeping track of versions is easy and storage-efficient using, for example, CVS. Even without sophisticated versioning software, changes from colleagues can be merged in using standard text processing tools such as `diff` and `patch`. Compare this to the situation with most presentation software, which use non-human-readable binary file formats and may not have built-in support for collaboration and versioning.

Free licence and community support. HA-prosper and its attendant programs (LaTeX, Ghostscript, *etc.*) are free software, not just in the sense that you can legally obtain copies without paying for them, but also in that you are free to run, copy, distribute, study, change, and improve them for any purpose. If HA-prosper doesn't work exactly the way you want, you can examine and adjust the code yourself, or hire a knowledgeable programmer to do it for you. Furthermore, LaTeX and its packages have an extensive support network through public mailing lists, user groups, publications, Usenet newsgroups, and private consulting firms.

Content before style. An inherent problem with the WYSIWYG paradigm used by most presentation software is that it conflates the tasks of *composition* (fixing one's ideas into words in a logically and semantically structured document) and *typesetting* (determining the superficial physical appearance of a document via, for example, colour and font settings). LaTeX, however, encourages writers to concentrate on content rather than style. Unlike with WYSIWYG editors, which do not always distinguish between semantic and physical markup, the physical appearance of an HA-prosper presentation is trivial to change even after the document has been written.

Reuse LaTeX code. Those who already use TeX or LaTeX to write their papers can leverage their existing knowledge rather than learn a whole new program for making slides. And for conference presentations based on existing papers, using LaTeX to produce the slides can save you a lot of time. For example, it's much easier to copy and paste complex equations, figures, and tables from the source document than it is to tediously re-key them in a GUI editor. The same goes for incorporating citations and a list of references from your paper's BIBTeX bibliography.

3 HA-prosper syntax

3.1 Overview

The basic structure of the source code for an HA-prosper presentation is illustrated in Figure 1. (Mandatory items are printed in **boldface**, and items modifiable by the author are printed in *italics*.) Anyone familiar with LaTeX will immediately note that it's not much different than any of the standard document classes. The only thing new is the `slide` environments, which, as you might guess, define the contents of individual slides.

```
\documentclass[options]{prosper}
\usepackage[options]{HA-prosper}

\title{Title of presentation}
\author{Name of author}

\begin{document}
\maketitle

    \begin{slide}[options]{Slide title}
      Material for the slide
    \end{slide}

\end{document}
```

Figure 1: The basic structure of an HA-prosper document

3.2 Setting up your document

3.2.1 Class options

As mentioned in the introduction, the HA-prosper package is based on the prosper package, which provides a class of the same name. Various class options are used to specify the output format; the most important of these are as follows:

`pdf`	This option tells prosper to produce a PDF file for presentation with a computer and video projector. (This is the default behaviour.)
`ps`	If instead you want to produce printed transparencies, this option can be used to produce a standard PostScript file. Any slide with overlays will be merged or "accumulated" into a single page.
`slideBW`	Use this option when the slides are to be printed on a black and white printer.
`distiller`	This option must be specified when you intend to generate a PDF file Adobe Distiller instead of Ghostscript (*i. e.*, `ps2pdf`).
`draft`	With this option, figures and graphics are replaced with bounding boxes. Because any graphics appearing in the slide template are re-embedded on every page they appear, this option significantly speeds up compilation time.

3.2.2 Package options

The HA-prosper package adds many useful extensions to prosper; among these are a number of options to control your slides' appearance. Some of these options must be specified as options to the `\usepackage` command:

`style`	Use the named layout template. This can be one of the ten templates included in the HA-prosper distribution, or one of your own devising. See Appendix B for a gallery of the prepackaged styles.
`toc`	This option creates a table of contents and displays it on each slide, typically as a sidebar. The table of contents is hyperlinked, making it easy to jump around your presentation with the click of a mouse.

`highlight`	With this option, the current slide is always highlighted on the table of contents. Useful for letting your audience keep track of where you are in your presentation.
`hlsections`	This option specifies that the current section in the table of contents is to be highlighted.
`landscape`	This option tells HA-prosper to create slides with landscape orientation. Since monitor resolutions are typically wider than they are tall, this is the default behaviour.
`portrait`	To override landscape mode, use this option. Only certain slide styles support portrait orientation.
`notes`	HA-prosper allows you to embed authors' notes in your document. They are normally hidden, but specifying the `notes` option includes them in the output.
`notesonly`	Use this option to output notes only; no slides. This is useful for making printed notes to accompany a computer presentation.

3.2.3 Footers

HA-prosper allows you to define left and right footers which appear on every slide. This is accomplished with the `\HAPsetup` command, which takes a list of comma-separated variable–value pairs as arguments. The variables controlling the left and right footers are `lf` and `rf`, respectively. Thus to put a copyright notice in the left footer and the current date in the right footer, you might use something like the following:

```
\HAPsetup{
  lf={\copyright 2005 Gnu Enterprises},
  rf={\today}
}
```

Another useful variable is `sn`. It controls the formatting of the page numbers, which are placed to the right of the right footer. The default value of `sn` is

```
{-~p.~\thepage\ifallPages/\totalpages\fi}
```

where `\thepage` prints the current slide number, `\totalpages` prints the total number of slides, and `\ifallPages...\fi` is a conditional branch which is skipped when the `nototal` document class option is used.

3.2.4 Title and author

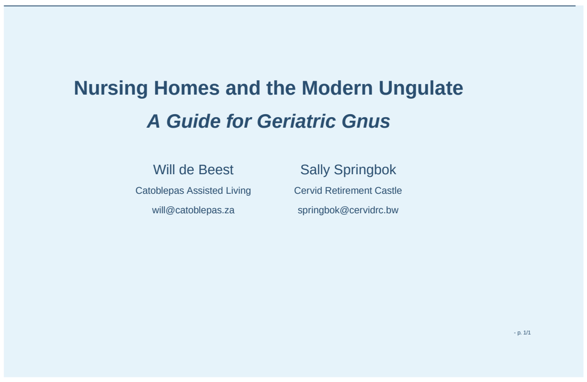
Just as with the standard document classes, the title and author of your slide presentation are specified with the `\title` and `\author` commands, respectively. HA-prosper also allows you to specify a subtitle to your presentation with the `\subtitle` command. More than one author can be specified by separating them with the `\and` command inside the argument to `\author`. Also within the `\author` command you can specify an `\institution` and `\email` for each author. For example:

```
\title{Nursing Homes and the Modern Ungulate}
\subtitle{A Guide for Geriatric Gnus}
\author{
  Will de Beest\\
  \institution{Catoblepas Assisted Living}\\
  \email{will@catoblepas.za}
  \and
  Sally Springbok\\
  \institution{Cervid Retirement Castle}\\
  \email{springbok@cervidrc.bw}
}
```

The output of `\maketitle` with the above example is illustrated in Figure 2.

3.3 The `slide` environment

Now that the basics of your layout have been established, you can start producing your slides. Generally speaking, all you need to do is enclose the material for each slide in its own `slide` environment, the only mandatory argument to which is the slide's title. Almost any sort of text or LaTeX environment, including figures, can appear inside a slide. Figure 3 shows a sample `slide` environment along with its output.



Nursing Homes and the Modern Ungulate

A Guide for Geriatric Gnus

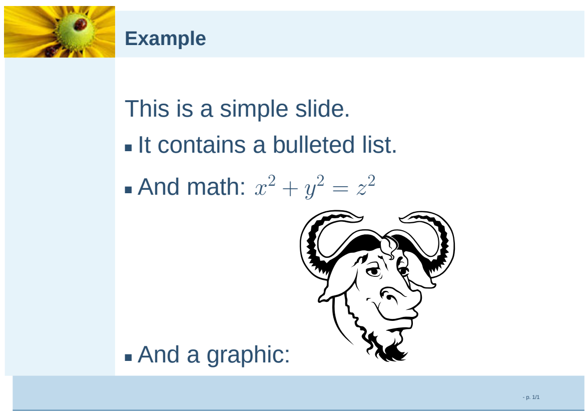
Will de Beest
Catoblepas Assisted Living
will@catoblepas.za

Sally Springbok
Cervid Retirement Castle
springbok@cervidrc.bw

- p. 1/1

Figure 2: A sample title slide

```
\begin{slide}{Example}
  This is a simple slide.
  \begin{itemize}
  \item It contains a bulleted list.
  \item And math: $x^2 + y^2 = z^2$
  \item And a graphic:
  \includegraphics[width=4cm]{GNU}
  \end{itemize}
\end{slide}
```



Example

This is a simple slide.

- It contains a bulleted list.
- And math: $x^2 + y^2 = z^2$
- And a graphic:

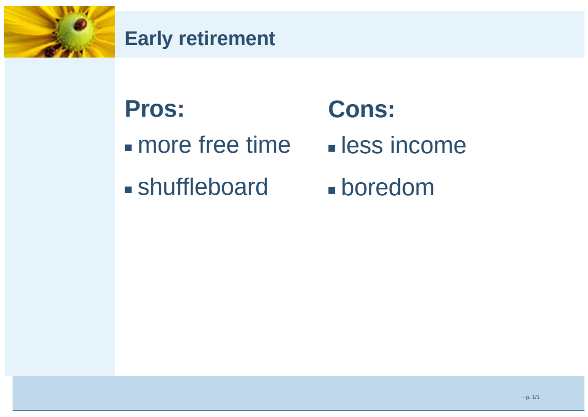
- p. 1/1

Figure 3: A simple `slide` environment and its output

3.4 Dual slides

It is often necessary for slides to show two blocks of text side-by-side—for example, a program plus its output, or two contrasting lists. Fortunately, HA-prosper provides the `\dualslide` command, a convenient mechanism for producing two-column slides. This command, which always appears *inside* a `slide` environment, takes three mandatory arguments: a list of options in the form of comma-separated variable–value pairs, the contents of the left column, and the contents of the right column. The list of options controls such things as the width and position of the columns; you can refer to the HA-prosper documentation for the full list.[5] Figure 4 shows an example dual slide and its output.

```
\begin{slide}{Early retirement}
  \dualslide{}{
    \textbf{Pros:}
    \begin{itemize}
    \item more free time
    \item shuffleboard
    \end{itemize}
  }{
    \textbf{Cons:}
    \begin{itemize}
    \item less income
    \item boredom
    \end{itemize}
  }
\end{slide}
```



Early retirement

Pros:	Cons:
▪ more free time	▪ less income
▪ shuffleboard	▪ boredom

- p. 1/1

Figure 4: Source code and output for a dual slide

3.5 Overlays

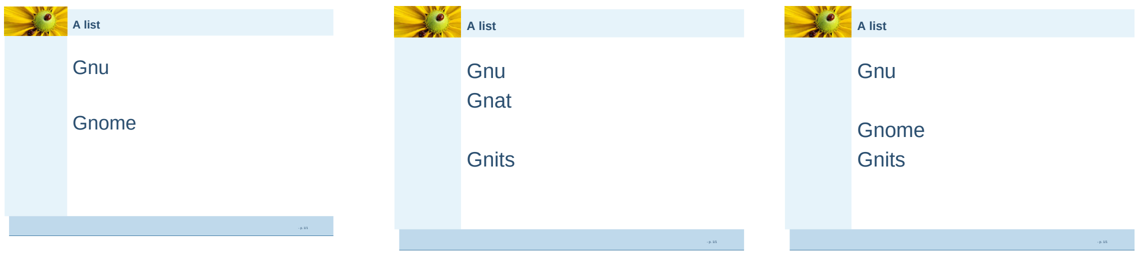
Overlays are created by wrapping a `slide` environment in the `\overlays` command, and then using the `\onSlide` command to specify which material appears

[5]The `\dualslide` command also takes three optional arguments which can use PSTricks commands to further modify the appearance of the two columns and the divider between them. We don't use them here, but they're explained in further detail in the HA-prosper and PSTricks documentation.

on which overlay. `\onSlide` takes two mandatory arguments: the first is a comma-delimited list of overlay numbers or ranges (in the form x-y, x-, or -x), and the second is the content to be placed on those overlays. This is best illustrated with an example:

```
\overlays{3}{%
  \begin{slide}{A list}
    Gnu\\
    \onSlide{2}{Gnat\\}
    \onSlide{1,3}{Gnome\\}
    \onSlide{2-}{Gnits\\}
  \end{slide}
}
```

This code defines a slide with three overlays.[6] The slide title, "A list", plus the text "Gnu" appear on all three overlays. The text "Gnat" appears only on the second overlay, its place being occupied by a blank line on the first and third overlays. The line "Gnome" appears on the first and third overlays, but is blank on the second. Finally, the line "Gnits" is displayed on the second and third slides, but not on the first. (The output is shown in Figure 5.)



A list

Gnu

Gnome

A list

Gnu
Gnat

Gnits

A list

Gnu

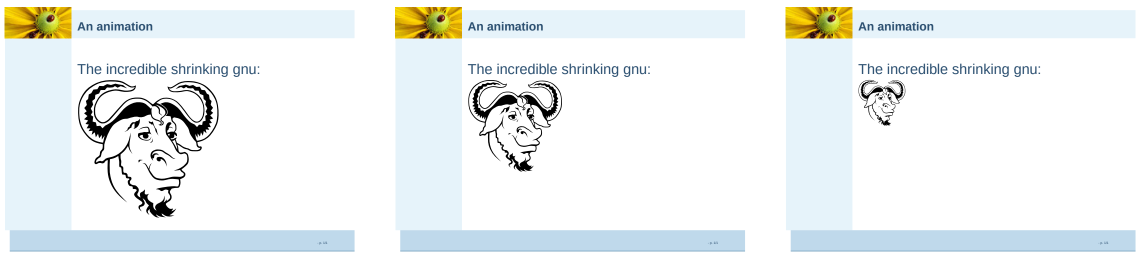
Gnome
Gnits

Figure 5: A slide with three overlays

There also exists a "starred" variant, `\onSlide*`, which does not leave blank space where hidden content will appear or has appeared. This is useful for creating animated graphics where variants of a certain image must appear in exactly the same place. This is demonstrated with the following code; the output is shown in Figure 6.

[6]More properly, the slide consists of a basic invariant background plus two overlays, but to simplify things we refer to the background as the first "overlay".

```
\overlays{3}{%
  \begin{slide}{An animation}
    The incredible shrinking gnu:\\
    \onSlide*{1}{\includegraphics[width=6cm]{GNU}}
    \onSlide*{2}{\includegraphics[width=4cm]{GNU}}
    \onSlide*{3}{\includegraphics[width=2cm]{GNU}}
  \end{slide}
}
```



An animation

The incredible shrinking gnu:

An animation

The incredible shrinking gnu:

An animation

The incredible shrinking gnu:

Figure 6: An animation using `\onSlide*`

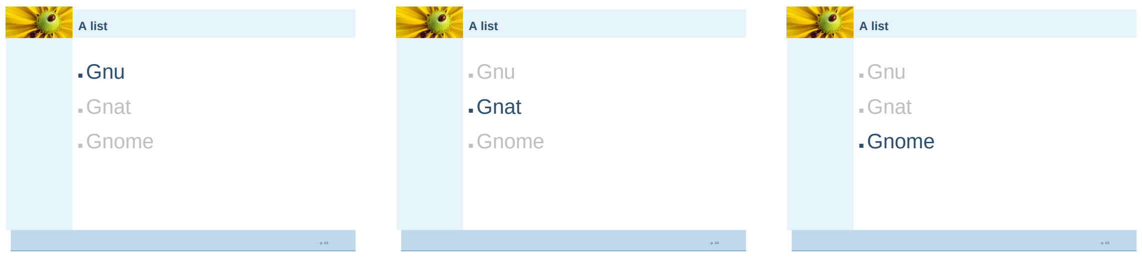
3.5.1 Animated lists

One common use of overlays is to produce animated item and enumeration lists, where each item in the list is displayed on a separate overlay. Since it's rather tedious to use `\onSlide` for this, HA-prosper provides the `itemstep` and `enumstep` environments to automate the process. These two environments function like `itemize` and `enumerate`, except that the list items are specified with the command `\xitem` instead of `\item`, and the items are revealed one at a time on separate overlays. Both environments take an optional argument consisting of variable–value pairs which modify the list's appearance. For example, the `stype` variable determines whether and how past and future items are displayed. With a value of `0`, items appear one by one and remain on the slide after their first appearance. With `stype=1`, items appear one at a time, but past items are grayed out when a newer one appears. A value of `2` causes all the list items to be displayed at once, but only the current item is not grayed out.

Other useful variables include `sstart`, which indicates which overlay the list is to begin on, and `iacolor`, which can be used to specify an "inactive" item colour other than gray.

A sample animated list follows; the output appears in Figure 7. Note that the `\overlay` command is still required, and moreover that its first argument must correspond to the number of items in the list (plus any other overlays the slide may contain).

```
\overlays{3}{%
  \begin{slide}{A list}
    \begin{itemstep}[stype=2]
      \xitem Gnu
      \xitem Gnat
      \xitem Gnome
    \end{itemstep}
  \end{slide}
}
```


A list

- Gnu
- Gnat
- Gnome

A list

- Gnu
- Gnat
- Gnome

A list

- Gnu
- Gnat
- Gnome

Figure 7: An animated list using the `itemstep` environment

3.6 Structuring your presentation

Documents written with the standard LaTeX classes are divided into logical sections using commands such as `\part`, `\section`, and `\subsection`. With HA-prosper, the corresponding sectioning commands are as follows:

`\part{`*`title`*`}` This command inserts a "title card" slide between separate parts of your presentation.

\tsection{*title*}
: This command simply indicates that the next slide marks the beginning of a new section in the table of contents. No separate title slide is created.

\tsectionandpart{*title*}
: To start a new section in the table of contents *and* create a title card, use this command.

3.7 Notes

As briefly mentioned earlier, HA-prosper provides a mechanism to embed notes in your presentation. These notes are normally suppressed from the output, though using the package options in §3.2.2 you can print out the notes separately and refer to them when giving your talk.

To create notes for a given slide, simply start a `notes` environment after said slide. Like the `slide` environment, `notes` requires a title as an argument:

```
\begin{note}{Notes for the first slide}
  Remember to grunt greetings to the herd.
\end{note}
```

4 Compiling your document

Compiling an HA-prosper presentation is not much different than compiling any other LaTeX document. For producing PostScript transparencies from an HA-prosper file `foo.tex`, the following command sequence is used:

```
latex foo.tex
dvips foo.dvi
```

The output then resides in the file `foo.ps`. For producing a PDF file `foo.pdf`, the command sequence will typically look as follows:

```
latex foo.tex
dvips foo.dvi
ps2pdf foo.ps
```

(Note that HA-prosper is not designed to work with `pdflatex`; the PostScript file generated by `latex` + `dvips` must be converted to PDF using Ghostscript (`ps2pdf`) or Adobe Distiller.) It bears mentioning that HA-prosper is designed to produce slides for US letter paper (8.5″×11″) only. If you have configured `dvips` to produce A4 or other-size output by default, you will need to use the `-t letter` command-line option.

4.1 Automating the process with AUCTEX

Most editors which support LATEX or programming languages will allow you to automate the compilation process through macros or configuration settings. If you are using the popular AUCTEX package with Emacs or XEmacs, perform the following steps to add a new "Prosper PDF" entry to your LATEX command menu:[7]

1. First, make sure AUCTEX is loaded by switching to LATEX mode: type `M-x latex-mode` and hit Enter.
2. Type `M-x customize-variable` and hit Enter.
3. At the `Customize variable:` prompt, type `TeX-command-list` and hit Enter.
4. A new buffer listing the commands is displayed. Scroll down to where in the list you'd like the new command to be displayed and press Enter overtop of the appropriate **`[INS]`** hyperlink.
5. A new form pops up. Enter the field data as follows:

   `Name:` `Prosper PDF`

   `Command:` `dvips -t letter %d; ps2pdf %f`

   `How:` `TeX-run-command`

   `Prompt:` `off (nil)`

6. Scroll back up to the top of the buffer and click on the "Set" button or hit Enter overtop the **`[Set for Current Session]`** hyperlink.

[7]As per Emacs convention, the notation `M-x` means to use the meta modifier key with `x`—*i. e.*, hold down your META or ALT key while pressing `x`, or hit ESC and then `x`.

7. Click on the "Save" button or hit Enter overtop the **`[Save for Future Sessions]`** hyperlink.

Another useful customization makes AUCTEX recognize when you're editing an HA-prosper presentation and automatically suggest the appropriate viewer when you invoke the `View` command. To set this up, perform the following:

1. First, make sure AUCTEX is loaded by switching to LATEX mode: type `M-x latex-mode` and hit Enter.

2. Type `M-x customize-variable` and hit Enter.

3. At the `Customize variable:` prompt, type `TeX-view-style` and hit Enter.

4. Again, a new buffer listing the commands is displayed. Position your cursor over the first **`[INS]`** hyperlink and hit Enter.

5. A new form pops up. Enter the field data as follows:

   `Regexp:` `^HA-prosper$`

   `Command:` `acroread %s.pdf`

   (You can substitute `acroread` with the command name of your preferred PDF viewer.)

6. Set and save the configuration as explained previously.

4.2 Troubleshooting

A problem sometimes encountered with HA-prosper is that ligatures such as 'fi' and 'ff' are not typeset correctly. This is actually due to a bug with how Ghostscript (*i. e.*, `ps2pdf`) up to and including version 7.21 produces level 1.2 PDF files. To avoid the problem, you can either upgrade to a newer version of Ghostscript, or force Ghostscript to generate level 1.3 output by invoking it as `ps2pdf13`.

4.3 Conclusion

In this article we have endeavoured to give the reader a general idea of the capabilities of LaTeX for producing high-quality presentation slides. However, the techniques presented in this article, while certainly enough to generate attractive and structured presentations, are actually only a subset of HA-prosper's full features. The package offers many more commands and configuration options to control your slides' appearance and navigation capabilities. For example, there are advanced features for creating "hidden" sections, PDF bookmarks, bibliography slides, and embedded sound files. All these and more are outlined in more detail in the official HA-prosper documentation.

A Installing HA-prosper

HA-prosper is bundled with the MiKTeX and TeX Live distributions, so users of those systems likely already have it installed or can install it from their source media. For most other distributions, including teTeX, it must be installed manually. This entails downloading the latest version of the package from CTAN or directly from the author's home page. HA-prosper also depends on a number of other packages, though most of them are fairly standard and will probably be already present on any reasonably recent system. However, a few of them, most likely xkeyval and prosper, may require manual installation. The list below indicates the most important packages and other tools you require, the minimum version number (if any), and where they can be found.

- HA-prosper $\geq$ 4.21
  - CTAN directory: `macros/latex/contrib/ha-prosper/`
  - `http://stuwww.uvt.nl/~hendri/Downloads/haprosper.html`
- prosper $\geq$ 1.5[8]
  - CTAN directory: `macros/latex/contrib/prosper/`

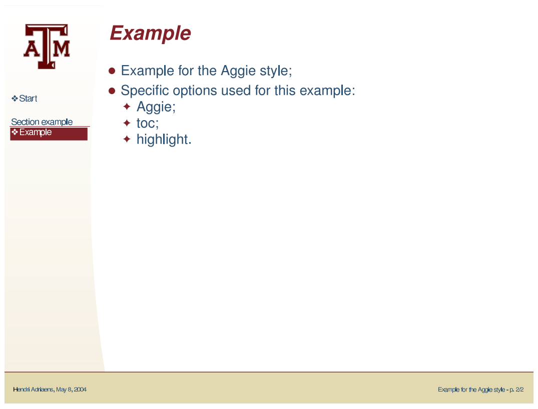
[8]At the time of this writing, the prosper package distributed on the official prosper website, `http://prosper.sourceforge.net/`, is the obsolete version 1.1. This version is incompatible with HA-prosper.

- xkeyval
  - CTAN directory: `macros/latex/contrib/xkeyval/`
  - `http://stuwww.uvt.nl/~hendri/Downloads/xkeyval.html`

For teTEX users with operating systems that support RPM, the present author has made available RPMs for each of the above three packages at `http://www.nothingisreal.com/tetex`.

B Slide styles

The screenshots in this section are reproduced from the HA-prosper documentation.



A|M

✧ Start

Section example
✧ Example

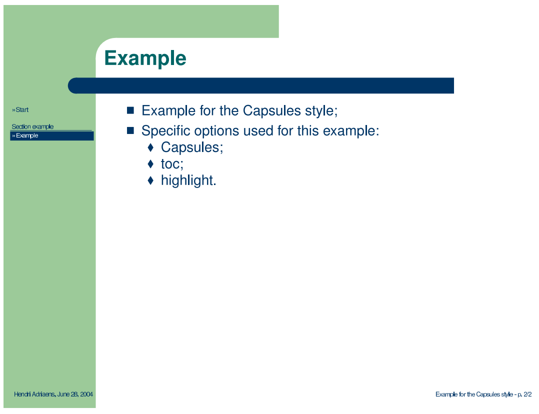
Example

- Example for the Aggie style;
- Specific options used for this example:
  - Aggie;
  - toc;
  - highlight.

Hendri Adriaens, May 8, 2004

Example for the Aggie style - p. 2/2

Figure 8: The `Aggie` style



»Start

Section example
»Example

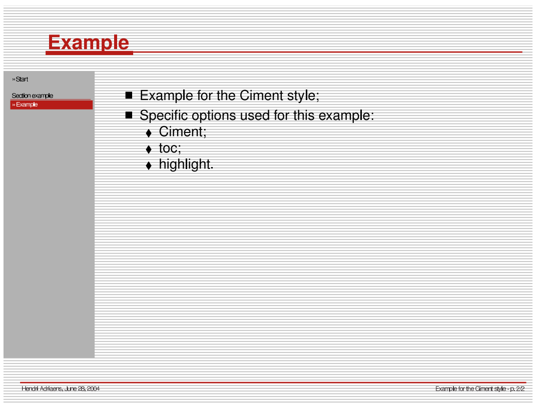
Example

- Example for the Capsules style;
- Specific options used for this example:
  - Capsules;
  - toc;
  - highlight.

Hendri Adriaens, June 28, 2004

Example for the Capsules style - p. 2/2

Figure 9: The `Capsules` style



Example

- Start

Section example

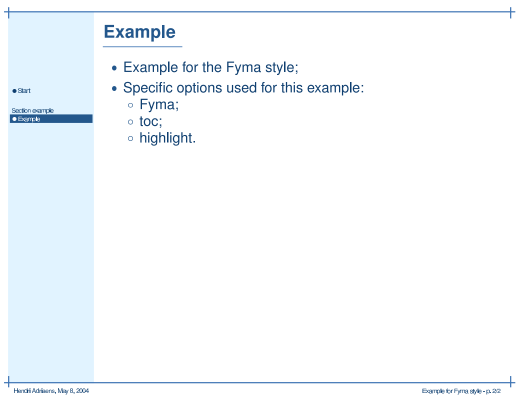
- Example

- Example for the Ciment style;
- Specific options used for this example:
  - Ciment;
  - toc;
  - highlight.

Hendri Adriaens, June 28, 2004

Example for the Ciment style - p. 2/2

Figure 10: The `Ciment` style



Example

- Start

Section example

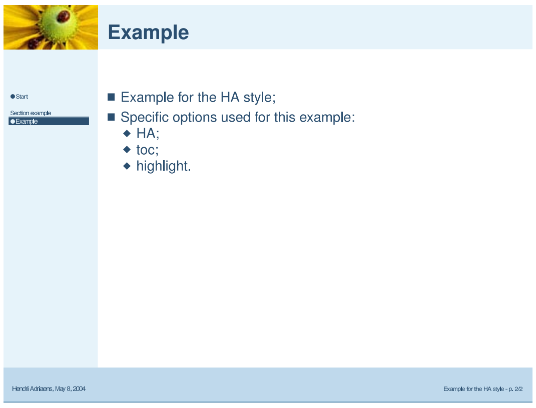
- Example

- Example for the Fyma style;
- Specific options used for this example:
  - Fyma;
  - toc;
  - highlight.

Hendri Adriaens, May 8, 2004

Example for Fyma style - p. 2/2

Figure 11: The `Fyma` style



Example

- Start

Section example

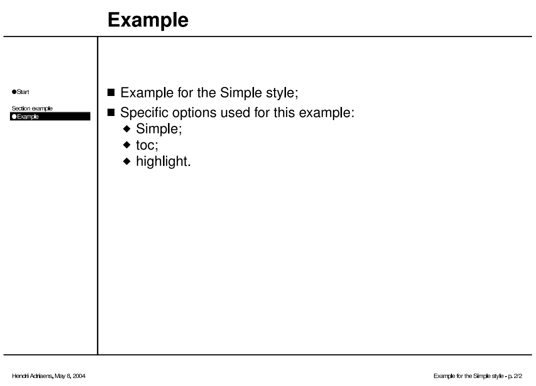
- Example

- Example for the HA style;
- Specific options used for this example:
  - HA;
  - toc;
  - highlight.

Hendri Adriaens, May 8, 2004

Example for the HA style - p. 2/2

Figure 12: The `HA` style



Example

- Start

Section example

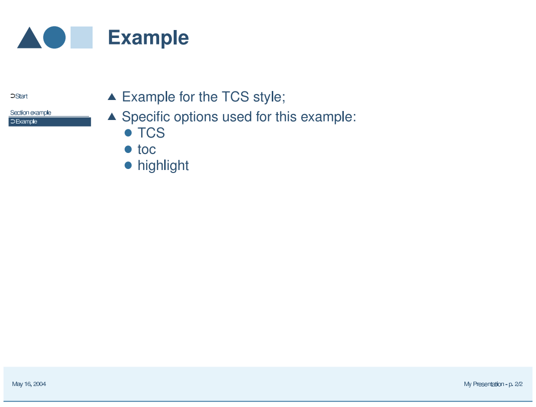
- Example

- Example for the Simple style;
- Specific options used for this example:
  - Simple;
  - toc;
  - highlight.

Hendri Adriaens, May 8, 2004

Example for the Simple style - p. 2/2

Figure 13: The `Simple` style

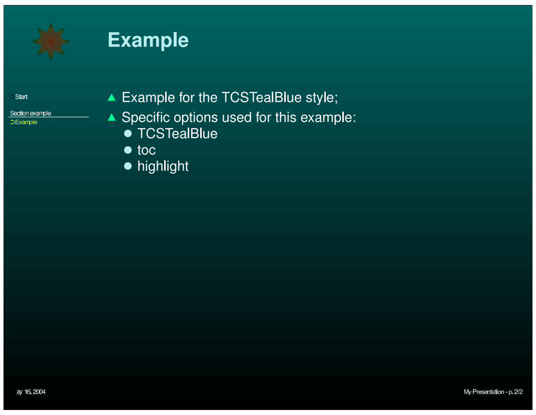

Example

⊃Start
Section example
⊃Example

- ▲ Example for the TCS style;
- ▲ Specific options used for this example:
  - ● TCS
  - ● toc
  - ● highlight

May 16, 2004 My Presentation - p. 2/2

Figure 14: The `TCS` style

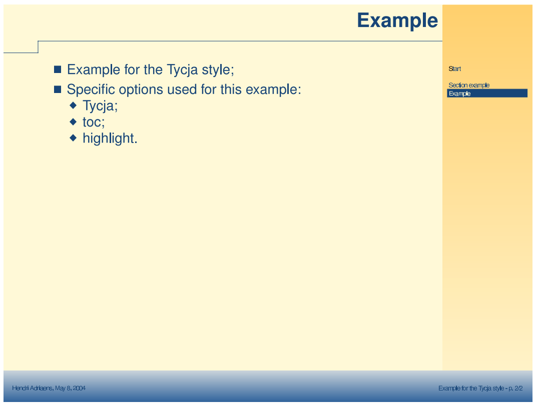

Example

Start
Section example
⊃Example

- ▲ Example for the TCSTealBlue style;
- ▲ Specific options used for this example:
  - ● TCSTealBlue
  - ● toc
  - ● highlight

ay 16, 2004 My Presentation - p. 2/2

Figure 15: The `TCSTealBlue` style

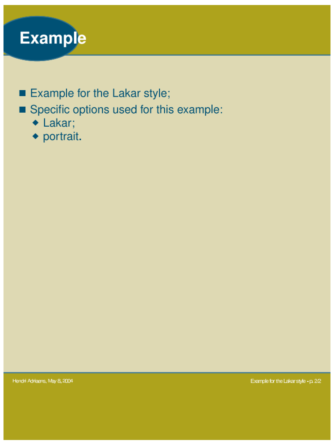

Example

Start
Section example
Example

- ■ Example for the Tycja style;
- ■ Specific options used for this example:
  - ◆ Tycja;
  - ◆ toc;
  - ◆ highlight.

Hendri Adriaens, May 8, 2004 Example for the Tycja style - p. 2/2

Figure 16: The `Tycja` style


Example

- ■ Example for the Lakar style;
- ■ Specific options used for this example:
  - ◆ Lakar;
  - ◆ portrait.

Hendri Adriaens, May 8, 2004 Example for the Lakar style - p. 2/2

Figure 17: The `Lakar` style

Acknowledgments

"A GNU head" from Figures 3 and 6 is probably originally by Etienne Suvasa. The version used in this article is by Aurélio A. Heckert and is reproduced under the terms of the Free Art license (`http://artlibre.org/`).

References

[1] Allin Cottrell. Word processors: Stupid and inefficient. Available at `http://ricardo.ecn.wfu.edu/~cottrell/wp.html`, June 1999.

[2] Conrad Taylor. What has WYSIWYG done to us? *The Seybold Report on Publishing Systems*, 26(2):3–12, September 1996.

[3] Michael Wiedmann. Screen presentation tools: Tools for creating screen or online presentations. Available at `http://www.miwie.org/presentations/`, January 2005.