

User's Guide to the Beamer Class, Version 0.82

<http://latex-beamer.sourceforge.net>

Till Tantau
tantau@users.sourceforge.net

October 15, 2003

Contents

1	Introduction	2
1.1	Overview	2
1.2	Getting Started: Installation	3
1.2.1	Installing Debian and Red Hat Packages	4
1.2.2	Temporary Installation	4
1.2.3	Installation in a texmf Tree	4
1.3	Testing the Installation	4
2	Workflow	5
2.1	Step Zero: Know the Time Constraints	5
2.2	Step One: Setup the Files	5
2.3	Step Two: Structure Your Presentation	5
2.4	Step Three: Creating a PDF or PostScript File	6
2.4.1	Creating PDF	6
2.4.2	Creating PostScript	6
2.5	Step Four: Create Frames	7
2.5.1	Guidelines on What to Put on a Frame	7
2.5.2	Guidelines on Text	8
2.5.3	Guidelines on Graphics	8
2.5.4	Guidelines on Colors	8
2.5.5	Guidelines on Animations and Special Effects	9
2.6	Step Five: Test Your Presentation	9
2.7	Step Six: Optionally Create a Handout	9
3	Frames and Overlays	9
3.1	Frame Titles	10
3.2	Commands with Overlay Specifications	10
3.3	Environments with Overlay Specifications	12
3.4	Restricting the Slides of a Frame	13
3.5	Dynamically Changing Text	14
3.6	Making Commands and Environments Overlay-Specification-Aware	15
3.7	Frames and Verbatim Commands	17
4	Structurizing the Text	18
4.1	Lists	18
4.2	Sections and Subsections	19
4.3	Table of Contents	20
4.4	Title Page	20
4.5	Columns	22

4.6	Bibliography	22
4.7	Predefined Environments and Commands	24
5	Graphics, Animations, and Special Effects	25
5.1	Graphics	25
5.1.1	Including External Graphic Files	25
5.1.2	Inlining Graphic Commands	26
5.2	Animations	26
5.3	Slide Transitions	27
6	Creating Handouts, Transparencies, and Notes	29
6.1	Creating Handouts	29
6.2	Creating Transparencies	30
6.3	Adding Notes	30
7	Customization	31
7.1	Fonts	31
7.2	Margins and Sizes	32
7.3	Class Options	32
7.4	Themes	33
7.5	Predefined Templates	35
7.5.1	Backgrounds	36
7.5.2	Head and Foot Lines	37
7.5.3	Itemizing	37
7.5.4	Bibliography	37
7.5.5	Navigation Bars	37
7.6	Changing Templates	38
7.6.1	Title Page	38
7.6.2	Background	38
7.6.3	Table of Contents	38
7.6.4	Frame Titles	39
7.6.5	Headlines, Footlines, and Footnotes	39
7.6.6	Navigation Bars	40
7.6.7	Hilighting Commands	41
7.6.8	Block Environments	41
7.6.9	Lists (Itemizations, Enumerations, Descriptions)	42
7.6.10	Bibliography	44
7.7	Template Inserts	44

1 Introduction

1.1 Overview

This user's guide explains the functionality of the BEAMER class. It is a L^AT_EX class that allows you to create a beamer presentation. It can also be used to create slides. It behaves similarly to other packages like PROSPER, but has the advantage that it works together directly with pdf_lat_ex, but also with dvips.

To use the BEAMER class, proceed as follows:

1. Specify `beamer` as document class instead of `article`.
2. Structure your L^AT_EX text using `section` and `subsection` commands.
3. Place the text of the individual slides inside `frame` commands.
4. Run pdf_lat_ex on the text (or lat_ex, dvips, and ps2pdf).

The BEAMER class has several useful features: You don't need any external programs to use it other than `pdflatex`, but it works also with `dvips`. You can easily and intuitively create sophisticated overlays. Finally, you can easily change the whole slide theme or only parts of it. The following code shows a typical usage of the class.

```
\documentclass{beamer}

\usepackage{beamerthemesplit}

\title{Example Presentation Created with the Beamer Package}
\author{Till Tantau}
\date{\today}

\begin{document}

\frame{\titlepage}

\section[Outline]{}
\frame{\tableofcontents}

\section{Introduction}
\subsection{Overview of the Beamer Class}
\frame
{
  \frametitle{Features of the Beamer Class}

  \begin{itemize}
    \item<1-> Normal LaTeX class.
    \item<2-> Easy overlays.
    \item<3-> No external programs needed.
  \end{itemize}
}
\end{document}
```

Run `pdflatex` on this code (twice) and then use, for example, the Acrobat Reader to present the resulting `.pdf` file in a presentation. You can also, alternatively, use `dvips`; see Section 2.4.2 for details.

As can be seen, the text looks almost like a normal \LaTeX text. The main difference is the usage of the `\frame` command. This command takes one parameter, which is the text that should be shown on the frame. Typically, the contents of a frame is shown on a single slide. However, in case you use overlay commands inside a frame, a single frame command may produce several slides. An example is the last frame in the above example. There, the `\item` commands are followed by *overlay specifications* like `<1->`, which means “from slide 1 on.” Such a specification causes the item to be shown only on the specified slides of the frame (see Section 3 for details). In the above example, a total of five slides are produced: a title page slide, an outline slide, a slide showing only the first of the three items, a slide showing the first two of them, and a slide showing all three items.

To structure your text, you can use the commands `\section` and `\subsection`. These commands will not only create a table of contents, but will also create navigation bars.

1.2 Getting Started: Installation

To use the beamer class, you just need to put the files of the BEAMER package in a directory that is read by \TeX . To uninstall the class, simply remove these files once more. The same is true of the PGF package, which you will also need.

Unfortunately, there are different ways of making \TeX “aware” of the files in the BEAMER package. Which way you should choose depends on how permanently you intend to use the class.

1.2.1 Installing Debian and Red Hat Packages

Currently, there are no out-of-the-box Debian or Red Hat packages of the beamer class available.

1.2.2 Temporary Installation

If you only wish to install the beamer class for a quick appraisal, do the following: Obtain the latest source version (ending `.tar.gz`) of the BEAMER package from <http://sourceforge.net/projects/latex-beamer/> (most likely, you have already done this). Next, you also need the PGF package, which can be found, for example, at <http://www.ctan.org/tex-archive/graphics/pgf/>.

In both cases, the packages contain a bunch of files (for the BEAMER class, `beamer.cls` is one of these files and happens to be the most important one, for the PGF package `pgf.sty` is the most important file). Place all files in two directories. For example, `/home/tantau/beamer/` and `/home/tantau/pgf/` would work fine for me. Then setup the environment variable called `TEXINPUTS` to be the following string (how exactly this is done depends on your operating system and shell):

```
./home/tantau/beamer:/home/tantau/pgf:
```

Naturally, if the `TEXINPUTS` variable is already defined differently, you should *add* the two directories to the list. Do not forget to place a colon at the end (corresponding to an empty path), which will include all standard directories.

1.2.3 Installation in a texmf Tree

For a more permanent installation, you can place the files of the BEAMER package and of the PGF package (see the previous subsection on how to obtain them) in an appropriate `texmf` tree.

When you ask \TeX to use a certain class or package, it usually looks for the necessary files in so-called `texmf` trees. These trees are simply huge directories that contain these files. By default, \TeX looks for files in three different `texmf` trees:

- The root `texmf` tree, which is usually located at `/usr/share/`, `c:\texmf\`, or `c:\Program Files\TeXLive\texmf\`.
- The local `texmf` tree, which is usually located at `/usr/local/share/`, `c:\localtexmf\`, or `c:\Program Files\TeXLive\texmf-local\`.
- Your personal `texmf` tree, which is located in your home directory.

You should install the packages either in the local tree or in your personal tree, depending on whether you have write access to the local tree. Installation in the root tree can cause problems, since an update of the whole \TeX installation will replace this whole tree.

Inside whatever `texmf` directory you have chosen, create the sub-sub-sub-directories

- `texmf/tex/latex/beamer` and
- `texmf/tex/latex/pgf`

and place all files in these two directories.

Finally, you need to rebuild \TeX 's filename database. This done by running the command `texhash` or `mktexlsr` (they are the same). In MikTeX, there is a menu option to do this.

For a more detailed explanation of the standard installation process of packages, you might wish to consult <http://www.ctan.org/installationadvice/>. However, note that the BEAMER package does not come with a `.ins` file (simply skip that part).

1.3 Testing the Installation

To test your installation, copy the file `beamerexample.tex` from the documentation subdirectory to some place where you usually create presentations. Then run the command `pdflatex` twice on the file and check whether the resulting `beamerexample.pdf` looks correct. If so, you are all set.

2 Workflow

This section presents a possible workflow for creating a beamer presentation and possibly a handout to go along with it. Technical questions are addressed, like which programs to call with which parameters, and hints are given on how to create a presentation. If you have already created numerous presentations, you may wish to skip the first of the following steps and only have a look at how to convert the `.tex` file into a `.pdf` or `.ps` file.

2.1 Step Zero: Know the Time Constraints

When you start to create a presentation, the very first thing you should worry about is the amount of time you have for your presentation. Depending on the occasion, this can be anything between 2 minutes and two hours. A simple rule for the number of frames is that you should have at most one frame per minute.

In most situations, you will have less time for your presentation than you would like. *Do not try to squeeze more into a presentation than time allows for.* No matter how important some detail seems to you, it is better to leave it out, but get the main message across, than getting neither the main message nor the detail across.

In many situations, a quick appraisal of how much time you have will show that you won't be able to mention certain details. Knowing this can save you hours of work on preparing slides that you would have to remove later anyway.

2.2 Step One: Setup the Files

It is advisable that you create a folder for each presentation. Even though your presentation will usually reside in a single file, \TeX produces so many extra files that things can easily get very confusing otherwise. The folder's name should ideally start with the date of your talk in ISO format (like 2003-12-25 for a Christmas talk), followed by some reminder text of what the talk is all about. Putting the date at the front in this format causes your presentation folders to be listed nicely when you have several of them residing in one directory. If you use an extra directory for each presentation, you can call your main file `main.tex`.

To create an initial `main.tex` file for your talk, copy an existing file (like the file `beamerexample.tex` that comes along with the contribution) and delete everything that is not going to be part of your talk. Adjust the `\author{}` and other fields as appropriate.

2.3 Step Two: Structure Your Presentation

With the time constraints in mind, make a mental inventory of the things you can reasonably talk about within the time available. Then categorize the inventory into sections and subsections. Put `\section{}` and `\subsection{}` commands into the (more or less empty) main file. Do not create any frames until you have a first working version of a possible table of contents. Do not feel afraid to change it later on as you work on the talk.

You should not use more than four sections and not less than two. Even four sections are usually too much, unless they follow a very easy pattern. Five and more sections are simply too hard to remember for the audience. After all, when you present the table of contents, the audience will not yet really be able to grasp the importance and relevance of the different sections and will most likely have forgotten them by the time you reach them.

Ideally, a table of contents should be understandable by itself. In particular, it should be comprehensible *before* someone has heard your talk. Keep section and subsection titles self-explaining.

Both the sections and the subsections should follow a logical pattern. Begin with an explanation of what your talk is all about. (Do not assume that everyone knows this. The Ignorant Audience Law states: The audience always knows less than you think it should know, even if you take the Ignorant Audience Law into account.) Then explain what you or someone else has found out concerning the subject matter. Always conclude your talk with a summary that repeats the main message of the talk in a short and simple way. People pay most attention at the beginning and at the end of talks. The summary is your "second chance" to get across a message.

2.4 Step Three: Creating a PDF or PostScript File

Once a first version of the structure is finished, you should create a first PDF or PostScript file of your (still empty) talk. This file will only contain the title page and the table of contents. The file might look like this:

```
\documentclass{beamer}
% This is the file main.tex

\usepackage{beamerthemesplit}

\title{Example Presentation Created with the Beamer Package}
\author{Till Tantau}
\date{\today}

\begin{document}

\frame{\titlepage}

\section[Outline]{}
\frame{\tableofcontents}

\section{Introduction}
\subsection{Overview of the Beamer Class}
\subsection{Overview of Similar Classes}

\section{Usage}
\subsection{...}
\subsection{...}

\section{Examples}
\subsection{...}
\subsection{...}

\end{document}
```

2.4.1 Creating PDF

To create a PDF version of this file, run the program `pdflatex` on `main.tex` at least twice. You need to run it twice, so that `TeX` can create the table of contents. In the following example, the greater-than sign is the prompt.

```
> pdflatex main.tex
... lots of output ...
> pdflatex main.tex
... lots of output ...
```

You can next use a program like the Acrobat Reader or `xpdf` to view the resulting presentation.

```
> acroread main.pdf
```

When printing a presentation, make sure that the option “expand small pages to paper size” is enabled. This is necessary, because slides are only 128mm times 96mm.

2.4.2 Creating PostScript

To create a PostScript version, you should first ascertain that the `HYPERRREF` package (which is automatically loaded by the `BEAMER` class) uses the option `dvips` or some compatible option, see the documentation of the

HYPERREF package for details. Whether this is the case depends on the contents of your local `hyperref.cfg` file. You can enforce the usage of this option by passing `dvips` or a compatible option to the BEAMER class (write `\documentclass[dvips]{beamer}`), which will pass this option on to the HYPERREF package.

You can then run `latex` twice, followed by `dvips`.

```
> latex main.tex
... lots of output ...
> latex main.tex
... lots of output ...
> dvips -P pdf main.dvi
```

The option `(-P pdf)` tells `dvips` to use Type 1 outline fonts instead of the usual Type 3 bitmap fonts. You may wish to omit this option if there is a problem with it.

If you wish each slide to completely fill a letter-sized page, use the following commands instead:

```
> dvips -P pdf -tletter main.dvi -o main.temp.ps
> psnup -1 -W128mm -H96mm -pletter main.temp.ps main.ps
```

For A4-sized paper, use:

```
> dvips -P pdf -ta4 main.dvi -o main.temp.ps
> psnup -1 -W128mm -H96mm -pa4 main.temp.ps main.ps
```

In order to create a white margin around the whole page (which is sometimes useful for printing), add the option `-m 1cm` to the options of `psnup`.

To put two or four slides on one page, use `-2`, respectively `-4` instead of `-1` as the first parameter for `psnup`. In this case, you may wish to add the option `-b 1cm` to add a bit of space around the individual slides.

You can convert a PostScript file to a pdf file using

```
> ps2pdf main.ps main.pdf
```

2.5 Step Four: Create Frames

Once the table of contents looks satisfactory, start creating frames for your presentation. In the following, some guidelines that I stick to are given on what to put on slides and what not to put. You can certainly ignore any of these guideline, but you should be aware of it when you ignore a rule and you should be able to justify it to yourself.

2.5.1 Guidelines on What to Put on a Frame

- A frame with too little on it is better than a frame with too much on it.
- Do not assume that everyone in the audience is an expert on the subject matter. (Remember the Ignorant Audience Law.) Even if the people listening to you should be experts, they may last have heard about things you consider obvious several years ago. You should always have the time for a quick reminder of what exactly a “semantical complexity class” or an “ ω -complete partial ordering” is.
- Never put anything on a slide that you are not going to explain during the talk, not even to impress anyone with how complicated your subject matter really is. However, you may explain things that are not on a slide.
- Keep it simple. Typically, your audience will see a slide for less than 50 seconds. They will not have the time to puzzle through long sentences or complicated formulas.

2.5.2 Guidelines on Text

- Put a title on each frame. The title explains the contents of the frame to people who did not follow all details on the slide.
- Ideally, titles on consecutive frames should “tell a story” all by themselves.
- *Never* use a smaller font size to “squeeze more on a frame.”
- Prefer enumerations and itemize environments over plain text. Do not use long sentences.
- Text and numbers in figures should have the *same* size as normal text. Illegible numbers on axes usually ruin a chart and its message.

2.5.3 Guidelines on Graphics

- Put (at least) one graphic on each slide, whenever possible. Visualizations help an audience enormously.
- Usually, place graphics to the left of the text. (Use the `columns` environment.)
- Graphics should have the same typographic parameters as the text: Use the same fonts (at the same size) in graphics as in the main text. A small dot in a graphic should have exactly the same size as a small dot in a text. The line width should be the same as the stroke width used in creating the glyphs of the font. For example, an 11pt non-bold Computer Modern font has a stroke width of 0.4pt.
- While bitmap graphics, like photos, can be much more colorful than the rest of the text, vector graphics should follow the same “color logic” as the main text (like black = normal lines, red = highlighted parts, green = examples, blue = structure).
- Like text, you should explain everything that is shown on a graphic. Unexplained details make the audience puzzle whether this was something important that they have missed. Be careful when importing graphics from a paper or some other source. They usually have much more detail than you will be able to explain.

For technical hints on how to create graphics, see Section 5.1.

2.5.4 Guidelines on Colors

- Use colors sparsely. The prepared themes are already quite colorful (blue = structure, red = alert, green = example). If you add more colors, you should have a *very* good reason.
- Be careful when using bright colors on white background, *especially* when using green. What looks good on your monitor may look bad during a presentation due to the different ways monitors, beamers, and printers reproduce colors. Add lots of black to pure colors when you use them on bright backgrounds.
- Maximize contrast. Normal text should be black on white or at least something very dark on something very bright. *Never* do things like “light green text on not-so-light green background.”
- Background shadings decrease the legibility without increasing the information content. Do not add a background shading just because it “somehow looks nicer.”
- Inverse video (bright text on dark background) can be a problem during presentations in bright environments since only a small percentage of the presentation area is light up by the beamer. Inverse video is harder to reproduce on printouts and on transparencies.

2.5.5 Guidelines on Animations and Special Effects

- Use animations to explain the dynamics of systems, algorithms, etc.
- Do *not* use animations just to attract the attention of your audience. This often distracts attention away from the main topic of the slide.
- Do *not* use distracting special effects like “dissolving” slides unless you have a very good reason for using them. If you use them, use them sparsely.

2.6 Step Five: Test Your Presentation

Always test your presentation. For this, you should vocalize or subvocalize your talk in a quiet environment. Typically, this will show that your talk is too long. You should then remove parts of the presentation, such that it fits into the allotted time slot. Do *not* attempt to talk faster in order to squeeze the talk into the given amount of time. You are almost sure to lose your audience this way.

Do not try to create the “perfect” presentation immediately. Rather, test and retest the talk and modify it as needed.

2.7 Step Six: Optionally Create a Handout

Once your talk is fixed, you can create a handout, if this seems appropriate. For this, use the class option `handout` as explained in Section 6.1. Typically, you might wish to put several handout slides on one page. See Section 2.4.2 on how to do this.

3 Frames and Overlays

A presentation consists of a series of frames. Each frame consists of a series of slides. You create a frame using the command `\frame`. This command takes one parameter, namely the contents of the frame. All of this text that is not tagged by overlay specifications is shown on all slides of the frame.

Command `frame`

Parameters:

1. optional parameter in square brackets: a specification of slides to be shown, see subsection 3.4 for details.
2. the frame’s contents.

Example:

```
\frame
{
  Some text...

  Some more...
}
```

Command `pictureframe`

Parameters:

1. optional parameter in square brackets: a specification of slides to be shown, see subsection 3.4 for details.
2. the frame’s contents.

Description: This command creates a frame that can be used to show a big picture that completely fills the frame. All head and foot lines will be suppressed.

Example:

```
\pgfdeclareimage{bigimage}{9.6cm}{bigimagefilename}
\pictureframe{\hfill\pgfuseimage{bigimage}\hfill}
```

3.1 Frame Titles

Each frame typically has an individual frame title, which is shown prominently at the top of the frame. This title is created using the command `\frametitle`.

Command `frametitle`

Parameters:

1. a title for the frame.

Example: `\frametitle{A Frame Title is Important.}`

3.2 Commands with Overlay Specifications

The number of slides in a frame is calculated automatically from the overlay specifications inside the frame, but see below for how to specify this number explicitly. The number is chosen to be the maximum number mentioned in any overlay specification. An overlay specification is a comma-separated list of slides and ranges. Ranges are specified like this: 2-5, which means slide two through to five. The start or the beginning of a range can be omitted. For example, 3- means “slides three, four, five, and so on” and -5 means the same as 1-5. A complicated example is -3,6-8,10,12-15, which selected the slides 1, 2, 3, 6, 7, 8, 10, 12, 13, 14, and 15.

Overlay specifications can be written behind certain commands. If such an overlay specification is present, the command will only “take effect” on the specified slides. What exactly “take effect” means depends on the command. Consider the following example.

```
\frame
{
  \textbf{This line is bold on all three slides.}
  \textbf<2>{This line is bold only on the second slide.}
  \textbf<3>{This line is bold only on the third slide.}
}
```

For the command `\textbf`, the overlay specification causes the text to be set in boldface only on the specified slides. On all other slides, the text is set in a normal font.

You cannot add an overlay specification to every command, but only to those listed below. However, it is quite easy to redefine a command such that it becomes “overlay specification aware.”

For the following commands, adding an overlay specification causes the command to be simply ignored on slides that are not included in the specification: `\textbf`, `\textit`, `\textsl`, `\textrm`, `\textsf`, `\color`, `\alert`, `\structure`. If a command takes several arguments, like `\color`, the specification directly follows the command as in the following example.

```
\frame
{
  \color<2-3>[rgb]{1,0,0} This text is red on slides 2 and 3, otherwise black.
}
```

For the following commands, the effect of an overlay specification is special:

Command only

Parameters:

1. a text

Description: If an overlay specification is present, the text is inserted only into the specified slides. For other slides, the text is simply thrown away. In particular, it occupies no space.

Example: `\only<3->\{Text inserted from slide 3 on.\}`

There exists a variant of `\only`, namely `\pgfonly`, that should be used inside PGF pictures instead of `\only`. The command `\pgfonly` inserts appropriate `\ignorespaces` commands that are needed by PGF.

Command `uncover`

Parameters:

1. a text

Description: If an overlay specification is present, the text is shown (“uncovered”) only on the specified slides. On other slides, the text still occupies space and it is still typeset, but it is not shown.

Example: `\uncover<3->\{Text shown from slide 3 on.\}`

Command `alt`

Parameters:

1. a slide specification in pointed brackets.
2. a main text
3. an alternative text

Description: The main text is shown on the specified slides, otherwise the alternative text. The specification must always be present.

Example: `\alt<2>\{On Slide 2\}\{Not on slide 2.\}`

Command `temporal`

Parameters:

1. a slide specification in pointed brackets.
2. a text to be put on all slides before the specified slides
3. a text to be put on the specified slides
4. a text to be put on all slides after the specified slides

Description: This command alternates between three different texts, depending on whether the current slide is temporally before the specified slides, is one of the specified slides, or comes after them. If the specification is not an interval (that is, if it has a “hole”), the “hole” is considered to be part of the before slides.

Example:

```
\temporal<3-4>\{Shown on 1, 2\}\{Shown on 3, 4\}\{Shown 5, 6, 7, ...}
\temporal<3,5>\{Shown on 1, 2, 4\}\{Shown on 3, 5\}\{Shown 6, 7, 8, ...}
```

For a useful application of the `\temporal` command consider the following example:

```
\def\colorize<#1>\{%
  \temporal<#1>\{ \color{structureshaded} \} \{ \color{black} \} \{ \color{shaded} \} \}

\frame{
  \begin{itemize}
    \colorize<1> \item First item.
    \colorize<2> \item Second item.
    \colorize<3> \item Third item.
    \colorize<4> \item Fourth item.
  \end{itemize}
}
```

Command `item`

Description: Adding an overlay specification to an item in a list causes this item to be uncovered only on the specified slides. This is useful for creating lists that are uncovered piecewise. Note that you are not required to stick to an order in which items are uncovered.

Example:

```
\frame
{
  \begin{itemize}
    \item<1-> First point, shown on all slides.
    \item<2-> Second point, shown on slide 2 and later.
    \item<2-> Third point, also shown on slide 2 and later.
    \item<3-> Fourth point, shown on slide 3.
  \end{itemize}
}

\frame
{
  \begin{enumerate}
    \item<3->[0.] A zeroth point, shown at the very end.
    \item<1-> The first an main point.
    \item<2-> The second point.
  \end{enumerate}
}
```

In the following concluding example, a list is uncovered item-wise. The last uncovered item is furthermore highlighted.

```
\frame
{
  The advantages of the beamer class are
  \begin{enumerate}
    \item<1-> \alert<1>{It is easy to use.}
    \item<2-> \alert<2>{It is easy to extend.}
    \item<3-> \alert<3>{It works together with \texttt{pdflatex}.}
    \item<4-> \alert<4>{It has nice overlays.}
  \end{enumerate}
}
```

3.3 Environments with Overlay Specifications

Environments can also be equipped with overlay specifications. For most of the predefined environments, see subsection 4.7, adding an overlay specifications causes the whole environment to be uncovered only on the specified slides. This is useful for showing things incrementally as in the following example.

```
\frame
{
  \frametitle{A Theorem on Infinite Sets}

  \begin{theorem}<1->
    There exists an infinite set.
  \end{theorem}

  \begin{proof}<3->
    This follows from the axiom of infinity.
  \end{proof}
}
```

```

\end{proof}

\begin{example}<2->
  The set of natural numbers is infinite.
\end{example}
}

```

In the example, the first slide only contains the theorem, on the second slide an example is added, and on the third slide the proof is also shown.

The two special environments `onlyenv` and `uncoverenv` are “environment versions” of the commands `\only` and `\uncover`.

Environment `onlyenv`

Description: If an overlay specification is given, the contents of the environment is inserted into the text only on the specified slides.

Example:

```

\frame
{
  This line is always shown.
  \begin{onlyenv}<2>
    This line is inserted on slide 2.
  \end{onlyenv}
}

```

Environment `uncoverenv`

Description: If an overlay specification is given, the contents of the environment is shown only on the specified slides. It still occupies space on the other slides.

Example:

```

\frame
{
  This word is
  \begin{uncoverenv}<2>
    visible
  \end{uncoverenv}
  only on slide 2.
}

```

3.4 Restricting the Slides of a Frame

As mentioned above, the number of slides in a frame is automatically calculated. If the largest number mentioned in any specification is 4, four slides are introduced (despite the fact that a specification like `<4->` might suggest that more than four slides would be possible).

You can also specify the number of slides in the frame “by hand.” To do so, you pass an optional argument to the `\frame` command, given in *square* brackets. This argument is also a slide specification. The frame will contain only the slides specified in this argument. Consider the following example.

```

\frame[1-2,4-]
{
  This is slide number \only<1>{1}\only<2>{2}\only<3>{3}%
  \only<4>{4}\only<5>{5}.
}

```

This command will create a frame containing four slides. The first will contain the text “This is slide number 1,” the second “This is slide number 2,” the third “This is slide number 4,” and the fourth “This is slide number 5.”

3.5 Dynamically Changing Text

You may sometimes wish to have some part of a frame change dynamically from slide to slide. On each slide of the frame, something different should be shown inside this area. You could achieve the effect of dynamically changing text by giving a list of `\only` commands like this:

```
\only<1>{Initial text.}
\only<2>{Replaced by this on second slide.}
\only<3>{Replaced again by this on third slide.}
```

The trouble with this approach is that it may lead to slight, but annoying differences in the heights of the lines, which may cause the whole frame to “whobble” from slide to slide. This problem becomes much more severe if the replacement text is several lines long.

So solve this problem, you can use two environments: `overlayarea` and `overprint`. The first is more flexible, but less user-friendly.

Environment `overlayarea`

Parameters:

1. The width of the area.
2. The height of the area.

Description: Everything within the environment will be placed in a rectangular area of the specified size. The area will have the same size on all slides of a frame, regardless of its actual contents.

Example:

```
\begin{overlayarea}{\textwidth}{3cm}
\only<1>{Some text for the first slide.\\Possibly several lines long.}
\only<2>{Replacement on the second slide.}
\end{overlayarea}
```

Environment `overprint`

Parameters:

1. Optional parameter in square brackets: width of the overprint area. Default: text width.

Description: Inside the environment, use `onslide` commands to specify different things that should be shown for this environment on different slides. The `onslide` commands are used like `item` commands. Everything within the environment will be placed in a rectangular area of the specified width. The height and depth of the area are chosen large enough to accommodate the largest contents of the area. The overlay specifications of the `onslide` commands must be disjoint.

Example:

```
\begin{overprint}
\onslide<1>
Some text for the first slide.\\
Possibly several lines long.
\onslide<2>
Replacement on the second slide.
\end{overprint}
```

3.6 Making Commands and Environments Overlay-Specification-Aware

This subsection explains how you can make your own commands overlay-specification-aware. Also, it explains how to setup counters correctly that should be increased from frame to frame (like equation numbering), but not from slide to slide. You may wish to skip this section, unless you want to write your own extensions to the BEAMER class.

You can define a new command that is overlay-specification-aware using the following command.

Command `newoverlaycommand`

Parameters:

1. name of the command
2. commands to be executed on the specified slides
3. commands to be executed otherwise

Description: Declares a new command. If this command is encountered, it is checked whether an overlay specification follows. If not, the commands given in the second parameter are executed. If there is a specification, the second parameter is executed if the current slide is specified, otherwise the third parameter is executed.

Example:

```
\newoverlaycommand{\SelectRedAsColor}{\color[rgb]{1,0,0}}{}
...
\frame
{
  \SelectRedAsColor<2>
  The second slide of this frame is all in red.
}
```

Command `renewoverlaycommand`

Parameters:

1. name of a command to be redefined
2. commands to be executed on the specified slides
3. commands to be executed otherwise

Description: Reddeclares a command that already exists in the same way as `newoverlaycommand`. Inside the parameters, you can still access to original definitions using the command `original`, see the example.

Example:

```
\renewoverlaycommand{\tiny}{\original{\tiny}}{}
...
\frame
{
  \tiny<2>This text is tiny on slide 2.
}
```

Command `newoverlayenvironment`

Parameters:

1. name of the environment
2. begin commands to be executed on the specified slides
3. end commands to be executed on the specified slides
4. begin commands to be executed otherwise

5. end commands to be executed otherwise

Description: Declares a new environment that is overlay specification aware. If this environment encountered, it is checked whether an overlay specification follows. If not or if it is found and the current slide is specified, the second and third parameters form the beginning and end of the environment. Otherwise, the fourth and fifth parameters are used.

This command can take one optional parameter, given in square brackets after the first parameter. If this parameter is specified, it must currently be 1. In this case, the begin commands must take one parameter. This parameter will *precede* the overlay specification, see the examples.

Example:

```
\newoverlayenvironment{mytheorem}{\alert{Theorem}:}{\Theorem:}{}

\frame
{
  \begin{mytheorem}<2>
    This theorem is hilighted on the second slide.
  \end{mytheorem}
}

\newoverlayenvironment{mytheorem}[1]{\alert{Theorem #1}:}{\Theorem #1:}{}

\frame
{
  \begin{mytheorem}{of Tantau}<2>
    This theorem is hilighted on the second slide.
  \end{mytheorem}
}
```

The following two commands can be used to ensure that a certain counter is automatically reset on subsequent slides of a frame. This is necessary for example for the equation count. You might want this count to be increased from frame to frame, but certainly not from overlay slide to overlay slide. For equation counters and footnote counters (you should not use footnotes), these commands have already been invoked.

Command `resetcounteronoverlays`

Parameters:

1. name of a \LaTeX counter

Description: After you have invoked this command, the value of the specified counter will be the same on all slides of every frame.

Example: `\resetcounteronoverlays{equation}`

Command `resetcountonoverlays`

Parameters:

1. name of a \TeX count register

Description: The same as `resetcounteronoverlays`, except that this command should be used with counts that have been created using the \TeX primitive `newcount` instead of \LaTeX 's `definecounter`.

Example:

```
\newcount\mycount
\resetcountonoverlays{mycount}
```


3.7 Frames and Verbatim Commands

The `\verb` command and the verbatim environment work only in frames that contain a single slide. Furthermore, you must explicitly specify that the frame contains only one slide; like this:

```
\frame[all:1]
{
  \frametitle{Our Search Procedure}

  \begin{verbatim}
int find(int* a, int n, int x)
{
  for (int i = 0; i<n; i++)
    if (a[i] == x)
      return i;
}
  \end{verbatim}
}
```

Instead of `\frame[all:1]` you could also have specified `\frame[1]`, but this works only for the presentation version of the talk, not for the handout version. To make verbatim accessible also in the handout version, you would have to specify `\frame[1| handout: 1]` and even more if you also have a transparencies version. The specification `\frame[all:1]` states that the frame has just one slide in all versions.

If you need to use verbatim commands in frames that contain several slides, you must *declare* your verbatim texts before the frame starts. This is done using two special commands:

Command `\defverb`

Parameters:

1. command name (including a backslash)
2. a one-line verbatim text, delimited by a special symbol (works like the `\verb` command). Adding a star before the second parameter make spaces visible.

Description: Declares a verbatim text for later use. The declaration should be done outside the frame. Once declared, the text can be used in overlays like normal text.

Example:

```
\defverb\mytext!int main (void) { ...!
\defverb\mytextspaces*!int  main  (void ){ ...!

\frame
{
  \begin{itemize}
  \item<1-> In C you need a main function.
  \item<2-> It is declare like this: \mytext
  \item<3-> Spaces are not important: \mytextspaces
  \end{itemize}
}
```

Command `\defverbatim`

Parameters:

1. command name (including a backslash)
2. a normal parameter that contains a `verbatim` or `verbatim*` environment.

Description: Declares a verbatim environment for later use. The declaration should be done outside the frame. Once declared, the text can be used in overlays like normal text.

Example:

```
\defverbatim\algorithm{
\begin{verbatim}
int main (void)
{
    cout << "Hello world." << endl;
    return 0;
}
\end{verbatim}}

\frame
{
    Our algorithm:

    \alert<1>{\algorithm}

    \uncover<2>{Note the return value.}
}
```

4 Structurizing the Text

4.1 Lists

There are three predefined environments for creating lists, namely `enumerate`, `itemize`, and `description`. The first two of there can be nested to depth two, but not further (this would create totally unreadable slides).

The `\item` command is overlay-specification-aware. If an overlay specification is provided, the item will only be shown on the specified slides, see the following example. If the `\item` command is to take an optional argument and an overlay specification, the overlay specification comes first as in `\item<1>[Cat]`.

```
\frame
{
    There are three important points:
    \begin{enumerate}
    \item<1-> A first one,
    \item<2-> a second one with a bunch of subpoints,
        \begin{itemize}
        \item first subpoint. (Only shown from second slide on!).
        \item<3-> second subpoint added on third slide.
        \item<4-> third subpoint added on fourth slide.
        \end{itemize}
    \item<5-> and a third one.
    \end{enumerate}
}
```

Environment `itemize`

Description: Used to display a list of items that do not have a special ordering. Inside the environment, use an `item` command for each topic. The appearance of the items can be changed using templates, see Section 7.5.

Example:

```
\begin{itemize}
\item This is important.
\item This is also important.
\end{itemize}
```

Environment `enumerate`

Description: Used to display an ordered list of items. Inside the environment, use an `item` command for each topic. The appearance of the items can be changed using templates, see Section 7.5.

Example:

```
\begin{enumerate}
\item This is important.
\item This is also important.
\end{enumerate}
```

Environment `description`

Parameters:

1. Some text, given as an optional parameter in square brackets. The width of the labels will be set to the width of this text. Normally, you choose the widest label in the description and copy it here.

Description: Used to display a list that explains or defines labels. Inside the environment, use an `item` with an argument in square brackets for each topic. The appearance of the items can be changed using templates, see Section 7.5.

Example:

```
\begin{description}
\item[Lion] King of the savanna.
\item[Tiger] King of the jungle.
\end{description}

\begin{description}[longest label]
\item<1->[short] Some text.
\item<2->[longest label] Some text.
\item<3->[long label] Some text.
\end{description}
```

4.2 Sections and Subsections

You can structure your text using the commands `\section` and `\subsection`. Unlike standard \LaTeX , these commands will not create a heading at the position where you use them. Rather, they will add an entry to the table of contents and also to the navigation bars.

In order to create a line break in the table of contents (usually not a good idea), you can use the command `\breakhere`. Note that the standard command `\\` does not work.

Command `section`

Parameters:

1. (optional, in square brackets) text to be shown in horizontal navigation bars
2. text to be shown in the table of contents; if empty, no entry is created.

Description: Starts a section. No heading is created, the section name is only shown in the table of contents and in the navigation bar. If the main parameter is empty, but the parameter in square brackets is not, a navigation entry is created, but no entry in the table of contents. This is useful for sections like a “table of contents section.”

Example: `\section[Summary]{Summary of Main Results}` or `\section[Outline]{}`

Command subsection

Parameters:

1. (optional, in square brackets) text to be shown in horizontal navigation bars
2. text to be shown in the table of contents; if empty, no entry is created.

Description: Starts a subsection. No heading is created, the subsection name is only shown in the table of contents and in the navigation bar. If the main parameter is empty, but the parameter in square brackets is not, a navigation entry is created, but no entry in the table of contents.

Example: `\subsection{Some Subsection}`

4.3 Table of Contents

The two commands for creating a table of contents are `\tableofcontents` and `\tableofcontentscurrent`. The difference between these commands is small: the first inserts the main table of contents into the current frame, the second inserts a table of contents in which only the current section is highlighted. An example usage is given in the following example:

```
\section[Outline]{  
\frame{\tableofcontents}  
  
\section{Introduction}  
\frame{\tableofcontentscurrent}  
\subsection{Why?}  
\frame{...}  
\frame{...}  
\subsection{Where?}  
\frame{...}  
  
\section{Results}  
\frame{\tableofcontentscurrent}  
\subsection{Because}  
\frame{...}  
\subsection{Here}  
\frame{...}
```

4.4 Title Page

You can use the `\titlepage` command to insert a title page into a frame.

The `\titlepage` command will arrange the following elements on the title page: the document title, the author(s)’s names, their affiliation, a title graphic, and a date. Before you invoke the title page command, you must specify all elements you wish to be shown. This is done using the following commands:

Command title

Parameters:

1. A shorter version of the title for inclusion in head lines and foot lines. This parameter is optional and given in square brackets.

2. A title for the document. Line breaks can be inserted using the double-backslash command.

Example:

```
\title{The Beamer Class}
```

```
\title[Short Version]{A Very Long Title\\Over Several Lines}
```

Command `author`

Parameters:

1. A shorter version of the authors for inclusion in head lines and foot lines. This parameter is optional and given in square brackets.
2. Names of the authors.

Description: The names should be separated using the command `and`. In case authors have different affiliations, they should be suffixed by the command `inst` with different parameters.

Example: `\author[Hemaspaandra et al.]{Lane Hemaspaandra\inst{1} \and Till Tantau\inst{2}}`

Command `institute`

Parameters:

1. A shorter version of the institute's name for inclusion in head lines and foot lines. This parameter is optional and given in square brackets.
2. Institute(s) where the authors work.

Description: If more than one institute is given, they should be separated using the command `and` and they should be prefixed by the command `inst` with different parameters.

Example:

```
\institute[Universities of Rochester and Berlin]{  
  \inst{1}Department of Computer Science\\  
  University of Rochester  
  \and  
  \inst{2}Fakult\"at f\"ur Elektrotechnik und Informatik\\  
  Technical University of Berlin}
```

Command `date`

Parameters:

1. A shorter version of the date for inclusion in head lines and foot lines. This parameter is optional and given in square brackets.
2. A text to be shown as date or occasion at which the talk was held.

Example: `\date{\today}` or `\date[STACS 2003]{STACS Conference, 2003}`.

Command `titlegraphic`

Parameters:

1. A text to be shown as title graphic. Typically, a picture environment is used as text.

Example: `\titlegraphic{\pgfuseimage{titlegraphic}}`

4.5 Columns

Three environments are used to create columns on a slide. Columns are especially useful for placing a graphic next to a description/explanation. The main environment for creating columns is called `columns`. Inside this environment, you can place several `column` environments. Each will create a new column.

Environment `columns`

Description: A multi-column area. Inside the environment you should place only `column` environments.

Example:

```
\begin{columns}
  \begin{column}{5cm}
    First column.
  \end{column}
  \begin{column}{5cm}
    Second column.
  \end{column}
\end{columns}
```

Environment `columnsonlytextwidth`

Description: This command has the same effect as `columns`, except that the columns will not occupy the whole page width, but only the text width.

Environment `column`

Parameters:

1. The width of the column.

Description: Creates a single column of the specified width. The column is centered vertically relative to the other columns.

4.6 Bibliography

You can use the bibliography environment and the `\cite` commands of \LaTeX in a BEAMER presentation. However, there are a few things to keep in mind:

- It is a bad idea to present a long bibliography in a beamer presentation. Present only very few references.
- Present references only if they are intended as “further reading,” for example at the end of a lecture.
- Using the `\cite` commands can be confusing since the audience has little chance of remembering the citations. If you cite the references, always cite them with full author name and year like “[Tantau, 2003]” instead of something like “[2,4]” or “[Tan01,NT02]”.

Keeping the above warnings in mind, proceed as follows to create the bibliography:

For a beamer presentation, you will typically have to typeset your bibliography items partly “by hand.” Nevertheless, you *can* use `bibtex` to create a “first approximation” of the bibliography. Copy the content of the file `main.bbl` into your presentation. If you are not familiar with `bibtex`, you may wish to consult its documentation. It is a powerful tool for creating high-quality citations.

Using `bibtex` or just your editor, you place your bibliographic references into an environment called `thebibliography`. This (standard \LaTeX) environment takes one parameter, which should be the longest `bibitem` label in the following list of bibliographic entries.

Environment `thebibliography`

Parameters:

1. Text of the longest label. Inside the environment, use one `\bibitem` command for each reference.

Description: Inserts a bibliography into the current frame. Must be placed inside a frame. If the bibliography does not fit on one frame, you should split it (create a new frame and a second `\thebibliography` environment). Even better, you should reconsider whether it is a good idea to present so many references.

Example:

```
\frame{
  \frametitle{For Further Reading}

  \begin{thebibliography}{Dijkstra, 1982}
  \bibitem[Solomaa, 1973]{Solomaa1973}
    A.~Salomaa.
    \newblock {\em Formal Languages}.
    \newblock Academic Press, 1973.

  \bibitem[Dijkstra, 1982]{Dijkstra1982}
    E.~Dijkstra.
    \newblock Smoothsort, an alternative for sorting in situ.
    \newblock {\em Science of Computer Programming}, 1(3):223--233, 1982.
  \end{thebibliography}
}
```

The parameter of the `\thebibliography` environment is used to determine the indent of the list. However, several templates for the typesetting of the bibliography (see Section 7.5.4) ignore this parameter since they replace the references by a symbol.

Inside the `\thebibliography` environment, use a (standard L^AT_EX) `\bibitem` command for each reference item. Inside each item, use a (standard L^AT_EX) `\newblock` command to separate the authors's names, the title, the book/journal reference, and any notes. Each of these commands may introduce a new line or color or other formatting, as specified by the template for bibliographies.

Command `\bibitem`

Parameters:

1. The text to be inserted into the text when the item is cited in the presentation (optional in square brackets). For a beamer presentation, this should usually be as long as possible.
2. A label to be used with the `\cite` commands.

Description: Adds a reference item to the bibliography. Use `\newblock` commands to separate the authors's names, the title, the book/journal reference, and any notes.

Example:

```
\bibitem[Dijkstra, 1982]{Dijkstra1982}
  E.~Dijkstra.
  \newblock Smoothsort, an alternative for sorting in situ.
  \newblock {\em Science of Computer Programming}, 1(3):223--233, 1982.
```

Note that, unlike normal L^AT_EX, the default template for the bibliography does not repeat the citation text (like “[Dijkstra, 1982]”) before each item in the bibliography. Instead, a cute, small article symbol is drawn. The rationale is that the audience will not be able to remember any abbreviated citation texts till the end of the talk. If you really insist on using abbreviations, you can use the command `\beamertemplate{textbibitems}` to restore the default behavior, see also Section 7.5.4.

4.7 Predefined Environments and Commands

The BEAMER class predefines a number of useful environments and commands. Using these commands makes it easy to change the appearance of a document by changing the theme.

Command `alert`

Parameters:

1. a text to be highlighted.

Description: The given text is highlighted, typically by coloring the text red. If an overlay specification is given, the command only has an effect on the specified slides.

Example: This is `\alert{important}`.

Command `structure`

Parameters:

1. a text to be marked as part of the structure of the text.

Description: The given text is marked as part of the structure, typically by coloring the text in the structure color. If an overlay specification is given, the command only has an effect on the specified slides.

Example: `\structure{Paragraph Heading.}`

Environment `block`

Parameters:

1. a block title

Description: Inserts a block, like a definition or a theorem, with a title. If an overlay specification is given, the block is shown only on the specified slides. In the example, the definition is shown only from slide 3 onwards.

Example:

```
\begin{block}{Definition}<3->
  A \alert{set} consists of elements.
\end{block}
```

Environment `alertblock`

Parameters:

1. a block title

Description: Inserts a block whose title is highlighted. If an overlay specification is given, the block is shown only on the specified slides.

Example:

```
\begin{alertblock}{Wrong Theorem}
  $1=2$.
\end{alertblock}
```

Environment `exampleblock`

Parameters:

1. a block title

Description: Inserts a block that is supposed to be an example. If an overlay specification is given, the block is shown only on the specified slides.

Example:

```
\begin{exampleblock}{Example}
  The set  $\{1,2,3,5\}$  has four elements.
\end{exampleblock}
```

Predefined English block environments, that is, block environments with fixed title, are: **Theorem**, **Proof**, **Corollary**, **Fact**, **Example**, and **Examples**. You can also use these environments with a lowercase first letter, the result is the same. The following German block environments are also predefined: **Problem**, **Loesung**, **Definition**, **Satz**, **Beweis**, **Folgerung**, **Lemma**, **Fakt**, **Beispiel**, and **Beispiele**. See the following example for their usage

```
\frame
{
  \frametitle{A Theorem on Infinite Sets}

  \begin{theorem}<1->
    There exists an infinite set.
  \end{theorem}

  \begin{proof}<2->
    This follows from the axiom of infinity.
  \end{proof}

  \begin{example}<3->
    The set of natural numbers is infinite.
  \end{example}
}
```

5 Graphics, Animations, and Special Effects

5.1 Graphics

Graphics can often convey concepts or ideas much more efficiently than text: A picture can say more than a thousand words. (Although, sometimes a word can say more than a thousand pictures.) In the following, the advantages and disadvantages of different possible ways of creating graphics for beamer presentations are discussed.

5.1.1 Including External Graphic Files

One way of creating graphics for a presentation is to use an external program, like **xfig** or the Gimp. These programs have an option to *export* a graphic files in a format that can then be inserted into the presentation.

The main advantage is:

- You can use a powerful program to create a high-quality graphic.

The main disadvantages are:

- You have to worry about many files. Typically there are at least two for each presentation, namely the program's graphic data file and the exported graphic file in a format that can be read by \LaTeX .
- Changing the graphic using the program does not automatically change the graphic in the presentation. Rather, you must reexport the graphic and rerun \LaTeX .
- It may be difficult to get the line width, fonts, and font sizes right.

- Creating formulas as part of graphics is often difficult or impossible.

In principle, you can use all the standard \LaTeX commands for inserting graphics, like the command `\includegraphic`. However, it may be advisable to use the special commands from the PGF package instead for this particular purpose. The reason is that `\includegraphic` will put a copy of the graphic into the file upon each invocation. In particular, if a frame includes a graphic and shows this graphic on ten slides, then ten copies of the possibly large graphic file will be inserted into the presentation file. This can result in huge files.

The PGF package offers a solution to this (but, currently, only if you use `pdflatex`): There, you must first *declare* every graphic. Once you have done this, you can *use* the graphic as often as you want and the graphic data will be put only once into the `.pdf` file. Furthermore, if you use `latex` instead of `pdflatex`, the PGF package will automatically search for a graphic file with the extension `.eps` instead of the extensions appropriate for `pdflatex`.

The PGF commands are used as follows: To declare an image, you write, somewhere early in your file,

```
\pgfdeclareimage{icon}{9pt}{10pt}{iconfile}
```

The first parameter is a string by which you can refer to the image later on. The second two parameters are the width and height of the image, although one (but not both) can be omitted in which case the missing value is computed automatically such that the image's aspect ratio remains correct. The last parameter is the graphic file name *without* the extension. Depending on whether a PDF or a PostScript file is created, the package will try appropriate extensions automatically (`.eps` for normal \LaTeX , `.png`, `.jpg`, and `.pdf` for `pdflatex`).

To use a previously declared image, just write

```
\pgfuseimage{icon}
```

For more details, consult the PGF User Manual.

5.1.2 Inlining Graphic Commands

A different way of creating graphics is to insert graphic drawing commands directly into your \LaTeX file. There are numerous packages that help you do this. They have various degrees of sophistication. Inlining graphics suffers from none of the disadvantages mentioned above for including external graphic files, but the main disadvantage is that it is often hard to use these packages. In some sense, you “program” your graphics, which requires a bit of practice.

When choosing a graphic package, there are a few things to keep in mind:

- Many packages produce poor quality graphics. This is especially true of the standard `picture` environment of \LaTeX .
- Powerful packages that produce high-quality graphics often do not work together with `pdflatex`.
- The most powerful and easiest-to-use package around, namely `pstricks`, does not work together with `pdflatex` and this is a fundamental problem. Due to the fundamental differences between PDF and PostScript, it is not possible to write a “`pdflatex` backend for `pstricks`.”

A solution to the above problem (though not necessarily the best) is to use the PGF package. It produces high-quality graphics and works together with `pdflatex`, but also with normal `latex`. It is not as powerful as `pstricks` (as pointed out above, this is because of rather fundamental reasons) and not as easy to use, but it should be sufficient in most cases.

5.2 Animations

There are two ways of creating animations. One way is to use the overlay commands of the BEAMER package to create a series of slides that, when shown in rapid succession, present an animation. This is a flexible approach, but such animations will typically be rather static since it will take at least one second to advance

from one slide to the next. This approach is mostly useful for animations where you want to explain each “picture” of the animation.

A second way is to include an animation generated using some external program (like a renderer) and to then use the capabilities of the presentation program (like the Acrobat Reader) to show the animation. Unfortunately, currently there is no portable way of doing this and even the Acrobat Reader does not support this feature on all platforms.

5.3 Slide Transitions

PDF in general, and the Acrobat Reader in particular, offer a standardized way of defining *slide transitions*. Such a transition is a visual effect that is used to show the slide. For example, instead of just showing the slide immediately, whatever was shown before might slowly “dissolve” and be replaced by the slide’s content.

Slide transitions should be used with great care. Most of the time, they only distract. However, they can be useful in a limited number of situations: For example, you might show a young boy on a slide and might wish to dissolve this slide into slide showing a grown man instead. In this case, the dissolving gives the audience visual feedback that the young boy “slowly becomes” the man.

There are a number of commands that can be used to specify what effect should be used when the current slide is presented. Consider the following example:

```
\frame{
  \pgfuseimage{youngboy}
}
\frame{
  \transdissolve
  \pgfuseimage{man}
}
```

The command `\transdissolve` causes the slide of the second frame to be shown in a “dissolved way.” Note that the dissolving is a property of the second frame, not of the first one. We could have placed the command anywhere on the frame.

The transition commands are overlay-specification-aware. We could collapse the two frames into one frame like this:

```
\frame{
  \only<1>{\pgfuseimage{youngboy}}
  \only<2>{\pgfuseimage{man}}
  \transdissolve<2>
}
```

This states that on the first slide the young boy should be shown, on the second slide the old man should be shown, and when the second slide is shown, it should be shown in a “dissolved way.”

In the following, the different commands for creating transitional effects are listed.

Command `transblindshorizontal`

Description: Show the slide as if horizontal blinds were pulled away.

Example: `\transblindshorizontal`

Command `transblindsvertical`

Description: Show the slide as if vertical blinds were pulled away.

Example: `\transblindsvertical<2,3>`

Command `transboxin`

Description: Show the slide by moving to the center from all four sides.

Example: `\transboxin<1>`

Command `transboxout`

Description: Show the slide by showing more and more of a rectangular area that is centered on the slide center.

Example: `\transboxout`

Command `transdissolve`

Description: Show the slide by slowly dissolving what was shown before.

Example: `\transdissolve`

Command `transglitter`

Parameters:

1. a degree (must be a multiple of 90).

Description: Show the slide with a glitter effect that sweeps in the specified direction.

Example: `\transglitter<2-3>{90}`

Command `transsplitverticalin`

Description: Show the slide by sweeping two vertical lines from the sides inward.

Example: `\transsplitverticalin`

Command `transsplitverticalout`

Description: Show the slide by sweeping two vertical lines from the center outward.

Example: `\transsplitverticalout`

Command `transsplithorizontalin`

Description: Show the slide by sweeping two horizontal lines from the sides inward.

Example: `\transsplithorizontalin`

Command `transsplithorizontalout`

Description: Show the slide by sweeping two horizontal lines from the center outward.

Example: `\transsplithorizontalout`

Command `transwipe`

Parameters:

1. a degree (must be a multiple of 90).

Description: Show the slide by sweeping a single line in the specified direction, thereby “wiping out” the previous contents.

Example: `\transwipe{90}`

6 Creating Handouts, Transparencies, and Notes

The BEAMER package offers different ways of creating special versions of your talk that can be used in different contexts. You can easily create a *handout* version of the presentation that can be distributed to the audience. You can also create a version that is more suitable for a presentation using an overhead projector. Finally, you can add notes for yourself that help you remember what to say for specific slides. All of these versions coexist in your main file. They are created by specifying different class options and rerunning \TeX on the main file.

6.1 Creating Handouts

A *handout* is a version of a presentation that is printed on paper and handed out to the audience before or after the talk. (See Section 2.4.2 for how to place numerous frames on one pages, which is very useful for handouts.) For the handout you typically want to produce as few slides as possible per frame. In particular, you do not want to print a new slide for each slide of a frame. Rather, only the “last” slide should be printed.

In order to create a handout, specify the class option `handout`. If you do not specify anything else, this will cause all overlay specifications to be suppressed. For most cases this will create exactly the desired result.

In some cases, you may want a more complex behaviour. For example, if you use many `\only` commands to draw an animation. In this case, suppressing all overlay specifications is not such a good idea, since this will cause all steps of the animation to be shown at the same time. In some cases this is not desirable. Also, it might be desirable to suppress some `\alert` commands that apply only to specific slides in the handout.

For a fine-grained control of what is shown on a handout, you can use *alternate overlay specifications*. They specify which slides of a frame should be shown for a special version, for example for the handout version. An alternate overlay specification is written alongside the normal overlay specification inside the pointed brackets. It is separated from the normal specification by a vertical bar and a space. The version to which the alternate specification applies is written first, followed by a colon. Here is an example:

```
\only<1-3,5-9| handout:2-3,5>{Text}
```

This specification says: “Normally, insert the text on slides 1–3 and 5–9. For the handout version, insert the text only on slides 2, 3, and 5.” If no alternate overlay specification is given for handouts, the default is “always.” This causes the desirable effect that if you do not specify anything, the overlay specification is effectively suppressed for the handout.

An especially useful specification is the following:

```
\only<3| handout:0>{Not shown on handout.}
```

Since there is no zeroth slide, the text is not shown. Likewise, `\alert<3| handout:0>{Text}` will not alert the text on a handout.

You can also use an alternate overlay specification for the optional argument of the `frame` command as in the following example.

```
\frame[1-| handout:0]{Text...}
```

This causes the frame to be suppressed in the handout version. Also, you can restrict the presentation such that only specific slides of the frame are shown on the handout:

```
\frame[1-| handout:4-5]{Text...}
```

It is also possible to give only an alternate overlay specification. For example, `\alert<handout:0>{...}` causes the text to be always highlighted during the presentation, but never on the handout version. Likewise, `\frame[handout:0]{...}` causes the frame to be suppressed for the handout.

Finally, note that it is possible to give more than one alternate overlay specification and in any order. For example, the following specification states that the text should be inserted on the first three slides in the presentation, in the first two slides of the transparency version, and not at all in the handout.

```
\only<trans:1-2| 1-3| handout:0>{Text}
```

If you wish to give the same specification in all versions, you can do so by specifying `all:` as the version. For example,

```
\frame[all:1-2]
{
  blah...
}
```

ensures that the frame has two slides in all versions.

6.2 Creating Transparencies

The main aim of the BEAMER class is to create presentations for beamers. However, it is often useful to print transparencies as backup, in case the hardware fails. A transparencies version of a talk often has less slides than the main version, since it takes more time to switch slides, but it may have more slides than the handout version. For example, while in a handout an animation might be condensed to a single slide, you might wish to print several slides for the transparency version.

You can use the same mechanism as for creating handouts: Specify `trans` as a class option and add alternate transparency specifications for the `trans` version as needed. An elaborated example of different overlay specifications for the presentation, the handout, and the transparencies can be found in the file `beamereexample.tex`.

6.3 Adding Notes

You can add notes to your slides using the command `\note`. A note is a reminder to yourself of what you should say or should keep in mind when presenting a frame. The `\note` command should be given after the frame to which the note applies. Here is a typical example.

```
\frame{
  \begin{itemize}
    \item<1-> Eggs
    \item<2-> Plants
    \item<3-> Animals
  \end{itemize}
}
\note{Tell joke about eggs.}
```

The note command will create a new page that contains your text plus some information that should make it easier to match the note to the frame while talking.

Since you normally do not wish the notes to be part of your presentation, you must explicitly specify the class option `notes` to include notes. If this option is not specified, notes are suppressed. If you specify `notesonly` instead of `notes`, only notes will be included and all normal frames are parsed, but not displayed. This is useful for printing the notes.

Command `note`

Parameters:

1. a note text.

Description: Creates a note page. Should be given right after a frame.

Example: `\note{Talk no more than 1 minute.}`

Command `noteitems`

Parameters:

1. a list of `item` commands.

Description: Just like the `note` command, except that an `itemize` environment is setup inside the note.

Example:

```
\frame{Bla bla...}
\noteitems{
\item Stress the importance.
\item Use no more than 2 minutes.
}
```

7 Customization

7.1 Fonts

By default, the beamer class uses the Computer Modern sans-serif fonts for typesetting a presentation. The Computer Modern font family is the original font family designed by Donald Knuth himself for the T_EX program. A sans-serif font is a font in which the letters do not have serifs (from French *sans*, which means “without”). Serifs are the little hooks at the ending of the strokes that make up a letter. The font you are currently reading is a serif font. By comparison, this text is in a sans-serif font.

The choice Computer Modern sans-serif had the following reasons:

- The Computer Modern family has a very large of symbols available that go well together.
- Sans-serif fonts are (generally considered to be) easier to read when used in a presentation. In low resolution rendering, serifs decrease the legibility of a font.

While these reasons are pretty good, you still might wish to change the font:

- The Computer Modern fonts are a bit boring if you have seen them too often. Using another font (but not Times!) can give a fresh look.
- Other fonts, especially Times, are sometime rendered better since they seem to have better internal hinting.
- A presentation typeset in a serif font creates a conservative impression, which might be exactly what you wish to create.

There are two ways of changing the document font: First, you must decide whether the text should be typeset in sans serif or in serif. To choose this, use either the class option `sans` or `serif`. By default, `sans` is selected, so you do not need to specify this. Furthermore, you can specify one of the two options `mathsans` or `mathserif`. These options override the overall sans-serif/serif choice for math text.

Second, you can independently switch the document font. To do so, you should use one of the prepared packages of L^AT_EX’s font mechanism. For example, to change to Times/Helvetica, simply add

```
\usepackage{times}
```

in your preamble. Note that if you do not specify `serif` as a class option, Helvetica (not Times) will be selected as the text font.

There may be many other fonts available on your installation. Typically, at least some of the following packages should be available: `avant`, `bookman`, `chancery`, `charter`, `euler`, `helvet`, `mathtime`, `mathptm`, `newcent`, `palatino`, `pifont`, `times`, `utopia`.

If you use `times` together with the `serif` option, you may wish to include also the package `mathptm`. If you use the `mathtime` package (you have to buy some of the fonts), you also need to specify the `serif` option.

7.2 Margins and Sizes

The “paper size” of a beamer presentation is fixed to 128mm times 96mm. The aspect ratio of this size is 4:3, which is exactly what most beamers offer these days. It is the job of the presentation program (like **acroread**) to display the slides at full screen size. The main advantage of using a small “paper size” is that you can use all your normal fonts at their natural sizes. In particular, inserting a graphic with 11pt labels will result in reasonably sized labels during the presentation.

You should refrain from changing the “paper size.” However, you *can* change the size of the left and right margins, which default to 1cm. To change them, use the command **\geometry**. For example, to set the left margin to 0.5cm and the right margin to 0.25cm, use

```
\geometry{left=0.5cm,right=0.25cm}
```

somewhere in the preamble. You can also specify the margins differently, see the documentation of the **geometry** package for details on the different ways.

7.3 Class Options

Class options are listed right behind the command **\documentclass** in square brackets. Class options, see the following list, govern certain global behaviors of the presentation.

Class Option notes

Description: Include notes in the output file. Normally, notes are not included.

Class Option notesonly

Description: Include only the notes in the output file. Useful for printing them.

Class Option handout

Description: Create a version that uses the **handout** overlay specifications. See subsection 6.1.

Class Option trans

Description: Create a version that uses the **trans** overlay specifications. See subsection 6.2.

Class Option hidesubsections

Description: Suppresses the subsections in the main table of contents. This is useful if you do not wish to show too many details when presenting the talk outline. However, the subsections of the current subsection will still be shown when using the command **tableofcontentscurrent**.

Class Option shadesubsections

Description: Show the subsections in the main table of contents in a shaded way. This is useful if you wish to show, but do not wish to talk about them when presenting the talk outline.

Class Option inrow

Description: All small frame representation in the navigation bars for a single section are shown alongside each other. Normally, the representation for different subsections are shown in different lines.

Class Option blue, red, grey, brown

Description: These options change the main color of the navigation and title bars to the given colors. Other colors can be setup using the command **themecolor**, which takes three parameters: a red, a green, and a blue value as floating point numbers between 0 (black) and 1 (highest intensity).

Class Option bigger

Description: Makes all fonts a little bigger, which makes the text more readable. The downside is that less fits onto each frame.

Class Option `smaller`

Description: Makes all fonts a little smaller, which allows you to fit more onto frames. Normally, this is not a good idea.

Class Option `sans`

Description: Use a sans-serif font during the presentation. (Default.)

Class Option `serif`

Description: Use a serif font during the presentation.

Class Option `mathsans`

Description: Override the math font to be a sans-serif font.

Class Option `mathserif`

Description: Override the math font to be a serif font.

7.4 Themes

Just like \LaTeX in general, the BEAMER class tries to separate the contents of a text from the way it is typeset (displayed). There are two ways in which you can change how a presentation is typeset: you can specify a different theme and you can specify different templates. A theme is a predefined collection of templates.

There exist a number of different predefined themes that can be used together with the BEAMER class. Feel free to add further themes. Themes are used by including an appropriate \LaTeX style file, using the standard `\usepackage` command.

Theme `beamerthemebars`

Example:

Outline	Our Model	Power of the Model	Limitations of the Model	Summary
oo	oooo	oo	ooo	
		oooooo	ooo	
		ooo	oo	
<div> <div> Computation with Absolutely No Space Overhead </div> <div> Lane Hemaspaandra¹ Proshanto Mukherji¹ Till Tantau² </div> <div> ¹Department of Computer Science University of Rochester </div> <div> ²Fakultät für Elektrotechnik und Informatik Technical University of Berlin </div> <div> Developments in Language Theory Conference, 2003 </div> </div> <div> <div> The Model of Overhead-Free Computation </div> <div> The Standard Model of Linear Space Our Model of Absolutely No Space Overhead </div> <div> The Power of Overhead-Free Computation </div> <div> Palindromes Linear Languages Context-Free Languages with a Forbidden Subword Languages Complete for Polynomial Space </div> <div> Limitations of Overhead-Free Computation </div> <div> Linear Space is Strictly More Powerful </div> </div>				
<div> Lane Hemaspaandra, Proshanto Mukherji, Till Tantau </div> <div> Computation with Absolutely No Space Overhead </div>				

Theme beamerthemeclassic

Example:

Outline	Our Model	Power of the Model	Limitations of the Model	Summary
oo	oo	oo	oo	oo
oooo	oooo	oooo	oooo	oooo
oo	oo	oo	oo	oo

Computation with Absolutely No Space Overhead

Lane Hemaspaandra¹ Proshanto Mukherji¹ Till Tantau²

¹Department of Computer Science
University of Rochester

²Fakultät für Elektrotechnik und Informatik
Technical University of Berlin

Developments in Language Theory Conference, 2003

The Model of Overhead-Free Computation

The Standard Model of Linear Space
Our Model of Absolutely No Space Overhead

The Power of Overhead-Free Computation

Palindromes
Linear Languages
Context-Free Languages with a Forbidden Subword
Languages Complete for Polynomial Space

Limitations of Overhead-Free Computation

Linear Space is Strictly More Powerful

Theme beamerthemelined

Example:

Outline	Our Model	Power of the Model	Limitations of the Model	Summary
oo	oo	oo	oo	oo
oooo	oooo	oooo	oooo	oooo
oo	oo	oo	oo	oo

Computation with Absolutely No Space Overhead

Lane Hemaspaandra¹ Proshanto Mukherji¹ Till Tantau²

¹Department of Computer Science
University of Rochester

²Fakultät für Elektrotechnik und Informatik
Technical University of Berlin

Developments in Language Theory Conference, 2003

The Model of Overhead-Free Computation

The Standard Model of Linear Space
Our Model of Absolutely No Space Overhead

The Power of Overhead-Free Computation

Palindromes
Linear Languages
Context-Free Languages with a Forbidden Subword
Languages Complete for Polynomial Space

Limitations of Overhead-Free Computation

Linear Space is Strictly More Powerful

Lane Hemaspaandra, Proshanto Mukherji, Till Tantau: Computation with Absolutely No Space Overhead

Lane Hemaspaandra, Proshanto Mukherji, Till Tantau: Computation with Absolutely No Space Overhead

Theme beamerthemeplain

Example:

Outline	Our Model	Power of the Model	Limitations of the Model	Summary
oo	oo	oo	oo	oo
oooo	oooo	oooo	oooo	oooo
oo	oo	oo	oo	oo

Computation with Absolutely No Space Overhead

Lane Hemaspaandra¹ Proshanto Mukherji¹ Till Tantau²

¹Department of Computer Science
University of Rochester

²Fakultät für Elektrotechnik und Informatik
Technical University of Berlin

Developments in Language Theory Conference, 2003

The Model of Overhead-Free Computation

The Standard Model of Linear Space
Our Model of Absolutely No Space Overhead

The Power of Overhead-Free Computation

Palindromes
Linear Languages
Context-Free Languages with a Forbidden Subword
Languages Complete for Polynomial Space

Limitations of Overhead-Free Computation

Linear Space is Strictly More Powerful

Example:

<div> <div>Outline</div> <div>The Model of Overhead-Free Computation</div> <div>The Power of Overhead-Free Computation</div> <div>Limitations of Overhead-Free Computation</div> <div>Summary</div> </div>	<div> <div>Outline</div> <div>The Model of Overhead-Free Computation</div> <div>The Power of Overhead-Free Computation</div> <div>Limitations of Overhead-Free Computation</div> <div>Summary</div> </div>
<div> <div>Computation with Absolutely No Space Overhead</div> <div> <div>Lane Hemaspaandra¹</div> <div>Proshanto Mukherji¹</div> <div>Till Tantau²</div> <div> <div>¹Department of Computer Science</div> <div>University of Rochester</div> <div>²Fakultät für Elektrotechnik und Informatik</div> <div>Technical University of Berlin</div> </div> <div>Developments in Language Theory Conference, 2003</div> </div> </div>	<div> <div>The Model of Overhead-Free Computation</div> <div> <div>The Standard Model of Linear Space</div> <div>Our Model of Absolutely No Space Overhead</div> </div> <div>The Power of Overhead-Free Computation</div> <div> <div>Palindromes</div> <div>Linear Languages</div> <div>Context-Free Languages with a Forbidden Subword</div> <div>Languages Complete for Polynomial Space</div> </div> <div>Limitations of Overhead-Free Computation</div> <div>Linear Space is Strictly More Powerful</div> </div>
<div>Lane Hemaspaandra, Proshanto Mukherji, Till Tantau</div> <div>Computation with Absolutely No Space Overhead</div>	<div>Lane Hemaspaandra, Proshanto Mukherji, Till Tantau</div> <div>Computation with Absolutely No Space Overhead</div>

Theme beamerthemetree

Example:

Computation with Absolutely No Space Overhead	Computation with Absolutely No Space Overhead ↳ Outline
<p>Computation with Absolutely No Space Overhead</p> <p>Lane Hemaspaandra¹ Proshanto Mukherji¹ Till Tantau²</p> <p>¹Department of Computer Science University of Rochester</p> <p>²Fakultät für Elektrotechnik und Informatik Technical University of Berlin</p> <p>Developments in Language Theory Conference, 2003</p>	<p>The Model of Overhead-Free Computation</p> <p>The Standard Model of Linear Space</p> <p>Our Model of Absolutely No Space Overhead</p> <p>The Power of Overhead-Free Computation</p> <p>Palindromes</p> <p>Linear Languages</p> <p>Context-Free Languages with a Forbidden Subword</p> <p>Languages Complete for Polynomial Space</p> <p>Limitations of Overhead-Free Computation</p> <p>Linear Space is Strictly More Powerful</p>

Theme beamerthemetreear

Example:

Computation with Absolutely No Space Overhead	Computation with Absolutely No Space Overhead
	Outline
<p>Computation with Absolutely No Space Overhead</p> <p>Lane Hemaspaandra¹ Proshanto Mukherji¹ Till Tantau²</p> <p>¹Department of Computer Science University of Rochester</p> <p>²Fakultät für Elektrotechnik und Informatik Technical University of Berlin</p> <p>Developments in Language Theory Conference, 2003</p>	<p>The Model of Overhead-Free Computation</p> <p>The Standard Model of Linear Space</p> <p>Our Model of Absolutely No Space Overhead</p> <p>The Power of Overhead-Free Computation</p> <p>Palindromes</p> <p>Linear Languages</p> <p>Context-Free Languages with a Forbidden Subword</p> <p>Languages Complete for Polynomial Space</p> <p>Limitations of Overhead-Free Computation</p> <p>Linear Space is Strictly More Powerful</p>

7.5 Predefined Templates

If you only wish to modify a small part of how your presentation is rendered, you do not need to create a whole new theme. Instead, you can modify an appropriate template.

A template specifies how a part of a presentation is typeset. For example, the frame title template dictates where the frame title is put, which font is used, and so on.

As the name suggests, you specify a template by writing the exact L^AT_EX code you would also use

when typesetting a single frame title by hand. Only, instead of the actual title, you use the command `\insertframetitle`.

For example, suppose we would like to have the frame title typeset in red, centered, and boldface. If we were to typeset a single frame title by hand, it might be done like this:

```
\frame
{
  \begin{centering}
    \color{red}
    \textbf{The Title of This Frame.}
  \par
  \end{centering}

  Blah, blah.
}
```

In order to typeset the frame title in this way on all slides, we can change the frame title template as follows:

```
\useframetitletemplate{
  \begin{centering}
    \color{red}
    \textbf{\insertframetitle}
  \par
  \end{centering}
}
```

We can then use the following code to get the desired effect:

```
\frame
{
  \frametitle{The Title of This Frame.}

  Blah, blah.
}
```

When rendering the frame, the BEAMER class will use the code of the frame title template to typeset the frame title and it will replace every occurrence of `\insertframetitle` by the current frame title.

In the next subsection predefined commands are listed that change one or more templates. In the two subsequent subsection, complete listings of all templates are given and the different `\insertxxxx` commands are explained.

7.5.1 Backgrounds

Just as there exist predefined themes, there exist predefined templates. They are defined in the package `beamertemplates`. Calling one of the following commands will change a template in a predefined way. Using them, you can use, for example, your favorite theme together with a predefined background.

Command `beamertemplateshadingbackground`

Parameters:

1. A comma-separated rgb triple specifying the color at the page bottom.
2. A comma-separated rgb triple specifying the color at the top of the page.

Description: Installs a vertically shaded background such that the specified bottom color changes smoothly to the specified top color. **Use with care: Background shadings are often distracting!** However, a very light shading with warm colors can make a presentation more lively.

Example:

```
\beamertemplateshadingbackground{1,0.9,0.9}{0.9,0.9,1}  
% Bottom is light red, top is light blue
```

Command `beamertemplategridbackground`

Description: Installs a light grid as background.

7.5.2 Head and Foot Lines

Command `beamertemplateheadempty`

Description: Makes the head line empty.

Command `beamertemplatefootempty`

Description: Makes the foot line empty.

Command `beamertemplatefootpagenumber`

Description: Shows only the page number in the foot line.

7.5.3 Itemizing

Command `beamertemplatedotitem`

Description: Changes the symbols shown in an `itemize` environment to dots.

Command `beamertemplateballitem`

Description: Changes the symbols shown in an `itemize` environment to small plastic balls.

7.5.4 Bibliography

Command `beamertemplatetextbibitems`

Description: Shows the citation text in front of references in a bibliography instead of a small symbol.

Command `beamertemplatearrowbibitems`

Description: Changes the symbol before references in a bibliography to a small arrow.

Command `beamertemplatebookbibitems`

Description: Changes the symbol before references in a bibliography to a small book icon.

Command `beamertemplatearticlebibitems`

Description: Changes the symbol before references in a bibliography to a small article icon. (Default)

7.5.5 Navigation Bars

Command `beamertemplateboxminiframe`

Description: Changes the symbols in a navigation bar used to represent a frame to a small box.

Command `beamertemplateticksminiframe`

Description: Changes the symbols in a navigation bar used to represent a frame to a small vertical bar of varying length.

7.6 Changing Templates

In the following, all commands that change a template are listed. Inside these commands, you should use the `\insertxxxx` commands listed in the next subsection.

7.6.1 Title Page

Command `usetitlepagetemplate`

Parameters:

1. a template for the title page

Example:

```
\usetitlepagetemplate{
  \vbox{}
  \vfill
  \begin{centering}
    \Large\structure{\inserttitle}
    \vskip1em\par
    \normalsize\insertauthortitle\vskip1em\par
    {\scriptsize\insertinstitute\par}\par\vskip1em
    \insertdate\par\vskip1.5em
    \inserttitlegraphic
  \end{centering}
  \vfill
}
```

7.6.2 Background

Command `usebackgroundtemplate`

Parameters:

1. a template for the page background

Example:

```
\usebackgroundtemplate{%
  \color{red}%
  \vrule height\paperheight width\paperwidth%
}
```

7.6.3 Table of Contents

Command `usetemplatetocsection`

Parameters:

1. a template for a section name in the table of contents.
2. a template for a grayed section name in the table of contents.

Example:

```
\usetemplatetocsection
{\color{structure}\inserttocsection}
{\color{structureshaded}\inserttocsection}
```

Command `usetemplatetocsubsection`

Parameters:

1. a template for a subsection name in the table of contents.
2. a template for a grayed subsection name in the table of contents.

Example:

```
\usetemplatetocsubsection
{\leavevmode\leftskip=1.5em\color{black}\inserttocsubsection\par}
{\leavevmode\leftskip=1.5em\color{shaded}\inserttocsubsection\par}
```

7.6.4 Frame Titles

Command `useframetitletemplate`

Parameters:

1. a template for the frame title

Example:

```
\useframetitletemplate{%
  \begin{centering}
    \structure{\textbf{\insertframetitle}}
  \par
  \end{centering}
}
```

7.6.5 Headlines, Footlines, and Footnotes

Command `usefoottemplate`

Parameters:

1. a template for the foot line

Example:

```
\usefoottemplate{\hfil\tiny{\color{shaded}\insertpagenumber}}
```

or

```
\usefoottemplate{%
  \vbox{%
    \tinycolouredline{structuremedium}%
    {\color{white}\textbf{\insertshortauthor\hfill\insertshortinstitute}}}%
  \tinycolouredline{structure}%
  {\color{white}\textbf{\insertshorttitle}\hfill}%
  }}%
```

Command useheadtemplate

Parameters:

1. a template for the head line

Example:

```
\useheadtemplate{%  
  \vbox{%  
    \vskip3pt%  
    \line{\insertnavigation{\paperwidth}}%  
    \vskip1.5pt%  
    \insertvrule{0.4pt}{structureshaded}}%  
}
```

Command usefootnotetemplate

Parameters:

1. A template for formatting a footnote.

Example:

```
\usefootnotetemplate{  
  \parindent 1em  
  \noindent  
  \hbox to 1.8em{\hfil\insertfootnotemark}\insertfootnotetext}
```

7.6.6 Navigation Bars

Command usesectionheadtemplate

Parameters:

1. a template for the current section name in a navigation bar.
2. a template for a different section name in a navigation bar.

Example:

```
\usesectionheadtemplate  
{\color{structure}\tiny\insertsectionhead}  
{\color{structureshaded}\tiny\insertsectionhead}
```

Command usesubsectionheadtemplate

Parameters:

1. a template for the current subsection name in a navigation bar.
2. a template for a different subsection name in a navigation bar.

Example:

```
\usesubsectionheadtemplate  
{\color{structure}\tiny\insertsubsectionhead}  
{\color{structureshaded}\tiny\insertsubsectionhead}
```


Command `useminislidetemplate`

Parameters:

1. a template for the mini frame of the current frame in a navigation bar.
2. a template for the mini frame of a frame of the current subsection in a navigation bar.
3. a template for the mini frame of other frames in a navigation bar.
4. horizontal offset between mini frames.
5. vertical offset between mini frames.

Example:

```
\useminislidetemplate
{
  \color{structure}%
  \hskip-0.4pt\vrule height\boxsize width1.2pt%
}
{%
  \color{structure}%
  \vrule height\boxsize width0.4pt%
}
{%
  \color{structureshaded}%
  \vrule height\boxsize width0.4pt%
}
{.1cm}
{.05cm}
```

7.6.7 Hilighting Commands

Command `usealerttemplate`

Parameters:

1. a template for the `alert` command

Example: `\usealerttemplate{{\color{red}\insertalert}}`

Command `usestructuretemplate`

Parameters:

1. a template for the `structure` command

Example: `\usestructuretemplate{{\color{structure}\insertstructure}}`

7.6.8 Block Environments

Command `useblocktemplate`

Parameters:

1. a template for the beginning of the block.
2. a template for the end of the block.

Example:

```

\useblocktemplate
{
  \%
  \medskip
  {\color{blockstructure}\textbf{\insertblockname}}\%
  \par
}
{\medskip}

```

Command usealertblocktemplate

Parameters:

1. a template for the beginning of the block.
2. a template for the end of the block.

Example:

```

\usealertblocktemplate
{
  \%
  \medskip
  {\alert{\textbf{\insertblockname}}}\%
  \par
}
{\medskip}

```

Command useexampleblocktemplate

Parameters:

1. a template for the beginning of the block.
2. a template for the end of the block.

Example:

```

\useexampleblocktemplate
{
  \%
  \medskip
  \begin{group}\color{darkgreen}{\textbf{\insertblockname}}
  \par
}
{
  \%
  \end{group}
  \medskip
}

```

7.6.9 Lists (Itemizations, Enumerations, Descriptions)

Command useenumerateitemtemplate

Parameters:

1. a template for the default item in the top level of an enumeration.

Example: `\useenumerateitemtemplate{\insertenumlabel}`

Command useitemizeitemtemplate

Parameters:

1. a template for the default item in the top level of an itemize list.

Example: `\useitemizeitemtemplate{\pgfuseimage{mybullet}}`

Command `usesubitemizeitemtemplate`

Parameters:

1. a template for the default item in the second level of an itemize list.

Example: `\usesubitemizeitemtemplate{\pgfuseimage{mysubbullet}}`

Command `useitemizetemplate`

Parameters:

1. a template for the beginning a top-level itemize list.
2. a template for the end of a top-level itemize list.

Example: `\useitemizetemplate{}{}`

Command `usesubitemizetemplate`

Parameters:

1. a template for the beginning a second-level itemize list.
2. a template for the end of a second-level itemize list.

Example: `\usesubitemizetemplate{\begin{small}}{\end{small}}`

Command `useenumeratetemplate`

Parameters:

1. a template for the beginning a top-level enumeration.
2. a template for the end of a top-level enumeration.

Example: `\useenumeratetemplate{}{}`

Command `usesubenumerateitemtemplate`

Parameters:

1. a template for the default item in the second level of an enumeration.

Example: `\usesubenumerateitemtemplate{\insertenumlabel-\insertsubenumlabel}`

Command `usesubenumeratetemplate`

Parameters:

1. a template for the beginning a second-level enumeration.
2. a template for the end of a second-level enumeration.

Example: `\usesubenumeratetemplate{\begin{small}}{\end{small}}`

Command `usedescriptiontemplate`

Parameters:

1. A template for the default item in a description. Use `insertdescriptionitem` to insert the current item text.
2. A default width for the default item, if no other width is specified; the width `labelsep` is automatically added to this parameter.

Example: `\usedescriptionitemtemplate{\color{structure}\insertdescriptionitem}{2cm}`

7.6.10 Bibliography

Command `usebibitemtemplate`

Parameters:

1. a template for the citation text before the entry. (The “label” of the item.)

Description: Use `insertbiblabel` to insert the label text.

Example: `\usebibitemtemplate{\color{structure}}\insertbiblabel}`

Command `usebibliographyblocktemplate`

Parameters:

1. a template to be inserted before the first block of the entry (the first block is all text before the first occurrence of a `newblock` command).
2. a template to be inserted before the second block (the text between the first and second occurrence of `newblock`)
3. a template to be inserted before the third block
4. a template to be inserted before all other blocks

Description: The templates are inserted *before* the blocks and you do not have access to the blocks themselves via insert commands. In the following example, the first `par` commands ensure that the author, the title, and the journal are put on different lines. The color commands cause the author (first block) to be typeset using the theme color, the second block (title of the paper) to be typeset in black, and all other lines to be typeset in a washed-out version of the theme color.

Example:

```
\usebibliographyblocktemplate
{\color{structure}}
{\par\color{black}}
{\par\color{structuremedium}}
{\par\color{structuremedium}}
```

7.7 Template Inserts

In the following, an alphabetical list of the different `\insertxxxx` commands is given. These commands are used inside templates.

Command `insertalert`

Description: Inserts the current alerted text into a template.

Command `insertauthor`

Description: Inserts the one-line version of the author names into a template.

Command `insertauthortitle`

Description: Inserts a version of the author names into a template that is useful for the title page.

Command `insertbiblabel`

Description: Inserts the current citation label into a template.

Command `insertblockname`

Description: Inserts the name of the current block into a template.

Command insertdate

Description: Inserts the date into a template.

Command insertdescriptionitem

Description: Inserts the current item of a description environment into a template.

Command insertenumlabel

Description: Inserts the current number of the top-level enumeration (as an Arabic number) into a template.

Command insertfootnotemark

Description: Inserts the current footnote mark (like a raised number) into a template.

Command insertfootnotetext

Description: Inserts the current footnote text into a template.

Command insertframetitle

Description: Inserts the current frame title into a template.

Command insertinstitute

Description: Inserts the institute into a template.

Command insertlogo

Description: Inserts the logo(s) into a template.

Command insertnavigation

Parameters:

1. a width

Description: Inserts a horizontal navigation bar of the given width into a template. The bar lists the sections and below them mini frames for each frame in that section.

Command insertpagenumber

Description: Inserts the current page number into a template.

Command insertsection

Description: Inserts the current section into a template.

Command insertsectionnavigation

Parameters:

1. a width

Description: Inserts a vertical navigation bar containing all sections, with the current section highlighted.

Command insertshortauthor

Description: Inserts the short version of the author into a template.

Command insertshortdate

Description: Inserts the short version of the date into a template.

Command insertshortinstitute

Description: Inserts the short version of the institute into a template.

Command insertshorttitle

Description: Inserts the short version of the document title into a template.

Command insertstructure

Description: Inserts the current structure text into a template.

Command insertsubenumlabel

Description: Inserts the current number of the second-level enumeration (as an Arabic number) into a template.

Command insertsubsectionnavigation

Parameters:

1. a width

Description: Inserts a vertical navigation bar containing all subsections of the current section, with the current subsection highlighted.

Command insertsubsection

Description: Inserts the current subsection into a template.

Command inserttitle

Description: Inserts a version of the document title into a template that is useful for the title page.

Command inserttitlegraphic

Description: Inserts the title graphic into a template.

Command inserttocsection

Description: Inserts the version of the current section name into a template that is useful for the table of contents.

Command inserttocsubsection

Description: Inserts the version of the current subsection name into a template that is useful for the table of contents.

Command insertvrule

Parameters:

1. a color
2. a thickness

Description: Inserts a rule of the given color and thickness into a template.