P^4
(PPower4)
Pdf Presentation Post Processor

Presentations with LaTeX and Acrobat Reader

Klaus Guntermann
Darmstadt University of Technology
Department of Computer Science
Systems Programming Group

(English version of report TI-14/99)

September 1999

Contents

1 Overview

More and more lecture halls are equipped with beamers. When preparing a lecture one can get rid of real slides to be presented with an overhead projector. But one has to determine, how one can prepare the contents for the presentation and which software can be choosen for displaying.

This report describes a tool, which helps to prepare contents with a large amount of mathematical formulas in an accustomed way and in good quality. First we show, why one cannot get the job done with the available tools. Then we explain, what areas still need compromises.

2 Requirements and area of application

For presentation with a beamer (what includes lecture presentations) the following software can be used:

- dedicated presentation programs like PowerPoint[1] [Koh94] (for Windows) or MagicPoint (for Linux/Unix),
- the preview program `xdvi`, which is used for proofreading of output from the typesetting program TeX [Knu86] under Unix, and
- Acrobat Reader for `pdf` [Ado99] files.

If a presentation is supposed to not only show colorful pictures and text, but needs a seamless integration of formulas, one does not have much choice. If there are only a very small number of formulas in a text, one may be able to include them as special graphics. But in a mathematically oriented text this strategy will be much too complicated.

The programs under consideration have the following properties:

2.1 PowerPoint etc.

MagicPoint is free software for Unix. The input is prepared with any text editor, but does not allow descriptions of a formula. But graphics can be included easily. It is rather straight forward to prepare an itemized list with various levels. The presentation can be stopped in the middle of a page and can be resumed after an interaction. Colorful backgrounds can be designed and included. Furthermore simple animated texts are one of the features.

PowerPoint is well known as a commercial software product to be used on a PC with the Windows operating systems. It is in widespread use. But it does support inclusion of formulas only by preparing a graphics replacement. The presentations are prepared within the system. Thus portability to other systems is a problem. A wide range of special effects can be selected from menus. It is easy to create backgrounds, prepare animations, to fade in and out parts of the text dynamically or to highlight parts depending on the current state of the presentation.

Nevertheless scientific texts require an easy and well known description of formulas. Both MagicPoint and PowerPoint cannot help with that. This makes

[1] trademarks are not marked in this text. This should not be taken as an indication, that all used names are free of rights of their trademark holders.

using them uncomfortable, although they can support the development of a presentation with dynamic effects visually and offer a wide range of extras.

2.2 xdvi

The program `xdvi` has been developed as a tool for proofreading. The features for presentations are rather limited. For example it is not possible to go through a page uncovering it step by step. Even a full screen mode can only be simulated by explicit configuration of the program and the window manager. Using rastered fonts the adaption to different screen resolutions can not be completely exact. If one has to use an unknown machine for a presentation during a conference, it has to be expected that the display cannot be scaled properly. Furthermore included (PostScript) graphics are rendered in the background by an external program and this may lead to delays when a page is advanced. The impression of an instantaneous switch of pages can only be expected with very fast machines and only without included graphics. Because a document will refer to external fonts and grahics files one has to supply a number of files for a presentation and make sure that all of them are available.

2.3 Acrobat Reader

The displaying program Acrobat Reader is available free of charge for many platforms. Unfortunately there is no source available. If a problem is discovered one has to wait for an update from the supplier. To prepare a document for displaying with Acrobat Reader one does not need to use the licensed program Distiller, which is not free. In the meantime a selection of free programs has evolved, which can create documents in the `pdf` format to be displayed by Acrobat Reader.

2.3.1 Displaying properties

Doing a presentation with Acrobat Reader one cannot dynamically uncover a text. Only moves between pages and hyperlinks are supported. The program supports full screen display and adapts to different screen sizes automatically, scaling fonts to the proper size. Advancing a page in full screen mode can be instantaneously. But other page replacement strategies are also possible, e.g. dissolving or striping. With “hot spots” one can navigate within a document, including dynamic back links, if several pages link to the same destination, which will bring you back to the page visited before. Each document is selfcontained, i.e. it includes any additional fonts and graphics. For a presentation one needs to supply only one file.

2.3.2 Creating `pdf` files

Any user who needs a large number of formulas and wants to process these on several different computing platforms will find that TeX and LaTeX [Lam95] are the best choice. Because TeX outputs a device independent format (`dvi`), one can convert this `dvi` output to `pdf`. Another option is to convert `dvi` to PostScript first and finally process this with the (commercial) Distiller. But there is another shorter way.

Newer versions of the TEX distribution teTeX for Unix (and other distributions as well) include `pdftex`, a variant of TEX which creates a `pdf` file directly for immediate presentation. A drawback is, that graphic inclusion works only for files in `JPEG` [Wal91], MetaPost [Hob92] or `pdf` format. But all graphics prepared with `xfig` [SS] can be easily converted to MetaPost with an appropriate backend. If a presentation is newly created and does not require to include a large number of PostScript graphics, this is the easiest way. And on this path the post processor PPower4 can be used.

3 Objectives of PPower4

The text formatter is designed to create printed documents. But the printed output is static and only of limited use for a live presentation. One missing feature is to be able to uncover a page step by step. It may be better in some cases, when a reader can read ahead and catch the overall view in advance. But on the other hand one may have an unexpected or surprising development. If this is within the range of the current slide, the remarkable item should neither go on the next slide nor be visible from the beginning.

Doing a presentation with Acrobat Reader one can give the effect of dynamically building a page, because pages are updated instantaneously. If one wishes to uncover a page in several steps, one can make a sequence of pages and add some more text on each of them. The only item to keep in mind that one has to avoid updating the page or slide number between the intermediate pages, if one has numbers for general orientation or reference.

3.1 Implementation using only the text formatter

A first solution for this task can be prepared using only the features of the text formatter. If there is enough material on a page to be displayed one can fill the current slide with (white) space and show it. Furthermore the contents of this page omitting the trailing space are carried over to the next slide without incrementing the counter.

As easy as this description is also an implementation of this strategy. The preparation of the intermediate page can be triggered independently of the current output routine. Of course one has to mark the action within the text. The macro for this is named `\pause`. The implementation can be seen in figure 1.

Within lists this macro can be inserted easily, as the following example shows.

```
\begin{enumerate}
\item this describes the first case,\pause
\item this the second one,\pause
\item and this the last.
\end{enumerate}
```

The result is shown as three minipages, which should give the impression of a page built dynamically, if the switching of pages is not noticeable.

```
%% texpause.sty                                        25 May 99
%%-------------------------------------------------------------
%% This is a quick hack to enable repeated pages with incremental
%% contents e.g. for displaying slides uncovering step by step.
%%
%% It depends on pdftex to be activated, i.e. for normal (La)TeX we
%% would ignore the command.
%%
%% The initial version (numbered 1.0) was written 07 May 99.
%% Version 1.1 was created 25 May 99 and fixed a naming problem.
%%
%% Possible extensions: make the page numbering optional (maybe using
%% subnumbering)
%% Make action optional (for printed versions via pdf).
%%
%% Plan of attack (should work with TeX and LaTeX):
%% - Get some ressources, i.e. one counter, one token register and one
%%   box.
%% - When activated save the current page count in the counter and the
%%   output routine in the token register.
%% - Setup a new output routine, which saves away a copy of the
%%   current page.
%% - Trigger this output routine to save the cumulated page contents.
%% - Restore the former output routine and run it with the restored
%%   saved contents.
%% - Reset the page count and reinsert the contents once again,
%%   removing the last glue item on the page.
%%
\ifx\pdfoutput\undefined
\let\pause=\relax\expandafter\endinput
\fi
\newbox\p@uses@vebox
\newtoks\p@uses@veoutput
\newcount\p@uses@vepage
\def\pause{\global\p@uses@vepage=\count0\relax %save pagenumber
  \p@uses@veoutput=\output % make backup copy of output routine
  \output={\global\setbox\p@uses@vebox=\box255}% copy current contents
         % only, when triggered
  \vfill\eject %trigger now
  \output=\p@uses@veoutput % restore output routine
  \unvcopy\p@uses@vebox % insert contents
  \eject % now really show the output
  \global\count0=\p@uses@vepage\relax %restore page number
  \unvbox\p@uses@vebox\vskip-\lastskip % and insert again for next turn
}%
```

Figure 1: Implementation of incremental page builds using only TEX

1. this describes the first case,	1. this describes the first case, 2. this the second one,	1. this describes the first case, 2. this the second one, 3. and this the last.

But this strategy has some drawbacks:

1. As soon as itemized lists fill a page completely, TeX will try to shrink the space between the items, when the bottom is reached, to improve the overall page. But this will lead to slight moves of the first items, when the last is added. However this annoying effect can be overcome by removing the shrinkability between the items.

2. It is impossible to interrupt within a line. Because the slide is filled vertically, the current line will be terminated. Space is limited on a slide due to the fact that one has to use large fonts for better readability given the limited resolution of beamer displays (currently XGA is an affordable maximum, that is 1024×768 pixels). And one can not always overcome the problem by starting a new line. Furthermore the final layout will be less readable with more short lines in different lengths.

3. One cannot step through an aligned multiline formula display line by line.

4. There are more structured text elements, which cannot be interrupted and resumed with this strategy.

After all this strategy is only helpful for a very limited amount of interaction. But it has the advantage that it can be used independently, that it can cooperate with many macro packages of the text formatter, that it does not need any post processing, and that it is not limited to `pdf`.

3.2 Post processing

Interrupting the displayed page at any place can only be achieved by post processing the output of TeX. If TeX creates device independent output, such a step is needed anyway for preparing the pages for the intended output device. The `dvi` format includes additional commands (“`\special`”) for communication between the formatter and the device driver. Creating `pdf` directly from TeX this step is omitted. There is no `dvi` file. And `pdf` does not have “special” commands. But `pdf` allows to insert comments into a document, which are normally ignored by the processing programs. These comments can be used to transport commands to the post processor, which has to split each stepwise developing page into a sequence of pages.
For the approach to postprocess the files there are the following requirements:

1. It should be avoided to have to run the post processor each time during development of a presentation, just to see where the steps are in a page.

2. Any bottom material (including page numbers etc.) should already be visible on the first partial page, not only when everything is shown (this

makes a difference compared to covering parts of a slide with a piece of paper on an overhead). The use of footnotes on slides should be avoided because of the reduced resolution and the lack of space.

3. Hyperlinks to a page should go to the complete page.

4. "hot spots" should be active already on partial pages.

These requirements are fulfilled as follows:

1. The text formatter can insert a small mark to visualize the point of interruption, which will not influence the layout of the page. This mark can be removed by the post processor and allows to judge the flow of the presentation without post processing.

2. The formatter must write the bottom matters together with the top matters before the body of a slide is written. While this is possible it requires modifications of the output routine of the macro package in use. One would wish that this sequence of creating the page output would become default or at least an option of the macro packages.

3. During post processing the complete pages must keep their places in the document or all references to them must be updated. Given the structure of a `pdf` document this can be achieved by keeping the sequence of pages and inserting the new intermediate pages.

4. Activating the "hot spots" already on partial pages can be done by copying all hot spot info from the complete page. But some care is required, if the hot spots are marked with frames. These would also show up on the partial pages. But frames for the hot spots are less helpful in a presentation, which is usually given by the author or one of the authors. In contrast to a new reader of a document the author does not need the highlighting of hot spots in a page. On the contrary. The highlighting may distract the audience from the really important parts of a page. Therefore it may be helpful to have hot spots not marked with frames and even leave them unmarked. The presenter will have a chance to know where a hot spot is during the presentation because Acrobat Reader changes the appearance of the mouse pointer, when it is over a hot spot. This can help to locate even unmarked hot spots.

3.3 Implementation

To process the `pdf` format we need a program, which can read, modify and write that format. Since 1999 there are some libraries available for this task. But some of them cover only parts of the requirements (e.g. can only read to display or can only write to convert to `pdf`) or they are not available as open source. One exception is a library written by Etymon Systems, Inc. [Nas98]. This library has been used to prepare a first prototype of PPower4. Using Java as the implementation language gives the additional advantage that the resulting program can be used on a wide range of platforms without requiring any implementation or porting effort. Because the post processing will not be needed that often during the development of a presentation any doubts concerning the throughput of this solution because of interpretation by a virtual machine will have only minor impact.

Using TEX the marker `\pause` inserts a small colored block (with zero width and height for the formatter) into the document. In the `pdf` file this block is surrounded by two comments, which will lead to a page split removing the block. Processing the document without creating `pdf` the command will have no effect.

It is not reasonable to include the whole program of the post processor here. Instead the following description should suffice:

- The post processor consists of
  - a control part, which parses the command line arguments and activates the other parts.
  - the file processor, which scans the input file page by page, looking for the comments to process. It has methods to modify the page contents and to create new pages.
  - the specific comment processors, which implement the associated actions.
- Any operations with the `pdf` file are kept in the file processor. If another library for `pdf` processing becomes available all other parts of the system should be unaffected.
- It is easy to add further comment processing to the system. Examples are discussed below.
- The system (not including the library to parse and create `pdf`) has about 1000 lines of Java code.

The required style files, libraries, and the program can be found at

`http://www-sp.iti.informatik.tu-darmstadt.de/software/ppower4/`

4 Extensions

A full blown presentation needs a lot more functionality, which is not completely implementable with the given environment.

4.1 Fading in and out

It may be helpful to fade in or out complete passages on a slide. But currently there is no user interface for such actions defined. Creating a document textually with a text editor and LATEX one might use minipages as the entity of manipulation. This approach would be rather expensive because `pdftex` places the items on a page using relative distances. This makes the sequence of items on a page fixed and one would have to keep track of all positions by complete analysis of the contents of a page.

4.2 Animations

Animated effects are currently not supported by Acrobat Reader. It would be necessary to supply them by external programs which can be called. I have not investigated, whether the `JavaScript` in Acrobat Reader can help in preparing a solution for this.

4.3 Backgrounds

The overall effect from a presentation can be enhanced by using color. This is in contrast to printed material, which is normally created by TEX. First the definition of a colored background can help. When this was initially taken into consideration for PPower4, background coloring was not supported for `pdftex`. But there has been a suggestion for an implementation of monochrome backgrounds in the Internet discussion list for `pdftex`. This can be done without post processing. But also for PPower4 the task is easily done.

But delegating the background insertion to a post processor may lead to unreadable documents before application of the post processor, if the foreground color of the text is indistinguishable from the default background.

5 User manual

This section will discuss installation and use of the post processor.

5.1 Installation of the post processor

To run the post processor the following items are needed: the `Java` libraries `pj` (tested with version 0.22) and `gnu.getopt`. Both are protected by the GNU general public license and can be obtained and used without charge.

These libraries can be combined with the implementation of PPower4 (if neither of them is required for other projects) in one `Java` archive file. Running the program under Unix the following short script may be helpful, which sets the environment for calling PPower4. The script can be as follows:

```
#!/bin/sh
CLASSPATH=/common/Java/lib/ppower4/pp4.jar \
          java de.tu_darmstadt.de.sp.pp4.PPower4 "$@"
```

5.2 Using PPower4 for a document

To use the macro `\pause` the definition file `pause.sty` must be processed. It has to be placed in an area which is searched by LATEX. To use the background processing additional definitions from `background.sty` are required.

The following background selections are available:

`\pagecolor{color}` Monochrome background in the selected color.

`\hpagecolor[color1]{color2}` Background color changing horizontally from `color1` to `color2` or, if the optional argument is missing, background color fading to brighter variant starting with `color2`.

`\vpagecolor[color1]{color2}` Color of background changing vertically from `color1` to `color2` or, if the optional argument is missing, background fading to brighter variant starting with `color2`.

The selected colors must have been defined for LaTeX. One can refer to the colors predefined by `color.sty` or add new colors with `\definecolor`. For changing colors both color definitions must be in the same color model (rgb or cmyk). Simple gray scale values are not supported.

If there is a background definition for a page to be split into partial pages, the background must be defined before the first `\pause` is triggered. Otherwise the selection will be ignored for this page. Backgrounds are kept for subsequent pages until redefined.

5.3 Calling the post processor

After processing the file with `pdftex/pdflatex` the post processor must be called. It requires at least the names of an input file and an output file. Furthermore the processing can be controlled selecting some of the following options:

`-v` increase verbosity for messages (can be used repeatedly). This is mostly usable for debugging new post processing schemes.

`--verbose=`*n* set verbosity level to *n*

`-n` do not compress the output file (useful for test or demonstration purposes)

`--nocompress` like `-n`

`-h` display a (helpful) usage message

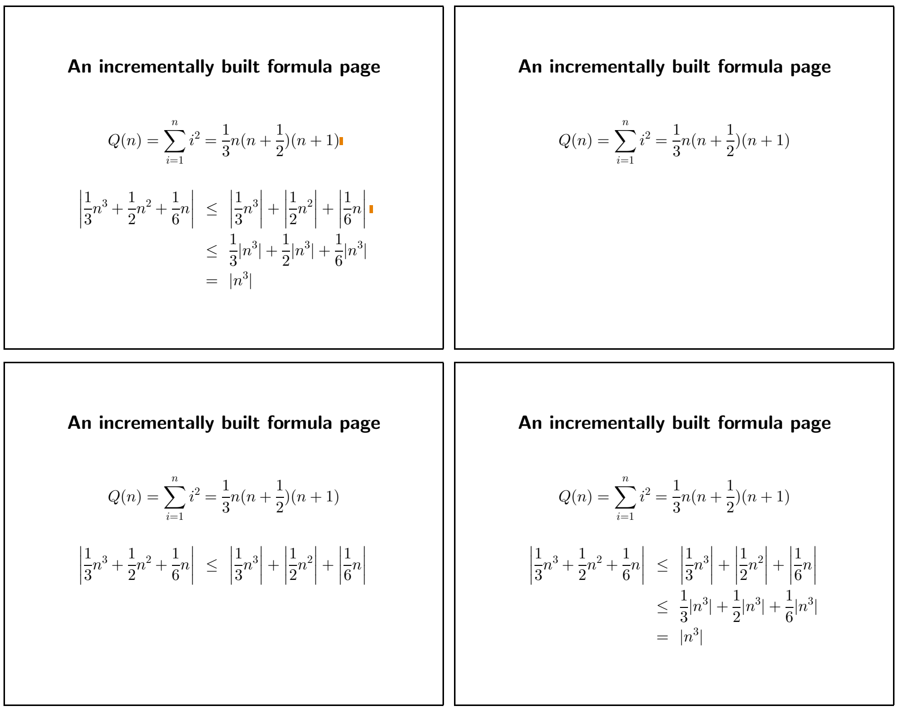
`--help` like `-h`

`-?` like `-h`

5.4 Examples

A first very small example using the class `foils` shows how to build a page inrementally with a multiline formula. To save some space we restrict ourselves to a small number of steps.

```
\documentclass[30pt,landscape]{foils}
\usepackage[pdftex]{geometry}
\geometry{headsep=3ex,hscale=0.9}
\usepackage[pdftex]{color} % for the colored block
\usepackage{pause}
\rightfooter{} % no more page numbers bottom right
\MyLogo{} % no logo bottom left
\begin{document}
\foilhead{An incrementally built formula page}
\[
   Q(n) = \sum_{i=1}^{n}i^{2} =
         \frac{1}{3}n(n+\frac{1}{2})(n+1)\pause
\]
```



An incrementally built formula page

$$Q(n)=\sum_{i=1}^{n} i^2=\frac{1}{3}n(n+\frac{1}{2})(n+1)$$

$$\begin{aligned}\left|\frac{1}{3}n^3+\frac{1}{2}n^2+\frac{1}{6}n\right| &\leq \left|\frac{1}{3}n^3\right|+\left|\frac{1}{2}n^2\right|+\left|\frac{1}{6}n\right|\\ &\leq \frac{1}{3}|n^3|+\frac{1}{2}|n^3|+\frac{1}{6}|n^3|\\ &= |n^3|\end{aligned}$$

An incrementally built formula page

$$Q(n)=\sum_{i=1}^{n} i^2=\frac{1}{3}n(n+\frac{1}{2})(n+1)$$

An incrementally built formula page

$$Q(n)=\sum_{i=1}^{n} i^2=\frac{1}{3}n(n+\frac{1}{2})(n+1)$$

$$\left|\frac{1}{3}n^3+\frac{1}{2}n^2+\frac{1}{6}n\right| \leq \left|\frac{1}{3}n^3\right|+\left|\frac{1}{2}n^2\right|+\left|\frac{1}{6}n\right|$$

An incrementally built formula page

$$Q(n)=\sum_{i=1}^{n} i^2=\frac{1}{3}n(n+\frac{1}{2})(n+1)$$

$$\begin{aligned}\left|\frac{1}{3}n^3+\frac{1}{2}n^2+\frac{1}{6}n\right| &\leq \left|\frac{1}{3}n^3\right|+\left|\frac{1}{2}n^2\right|+\left|\frac{1}{6}n\right|\\ &\leq \frac{1}{3}|n^3|+\frac{1}{2}|n^3|+\frac{1}{6}|n^3|\\ &= |n^3|\end{aligned}$$

Figure 2: Intermediate file and final result of first example

```
\begin{eqnarray*}
\left|\frac{1}{3}n^3+\frac{1}{2}n^{2}+\frac{1}{6}n\right|
 &\leq&\left|\frac{1}{3}n^3\right|+\left|\frac{1}{2}n^{2}\right|
         +\left|\frac{1}{6}n\right|\pause\\
 &\leq&\frac{1}{3}|n^{3}|+\frac{1}{2}|n^{3}|+\frac{1}{6}|n^{3}|\\
 &=&|n^{3}|
\end{eqnarray*}
\end{document}
```

Running `pdflatex` on this file will create a document with only one page. Using the post processor this will be extended to three pages. The intermediate document with the marking blocks and the final result are shown scaled down in figure 2.

In the following extended example the details can be examined in more depth.

```
\documentclass[30pt,landscape]{foils}
\usepackage[english,german]{babel}  % language support
\usepackage[pdftex]{color}
\usepackage[pdftex]{geometry}
\geometry{headsep=3ex,hscale=0.9}
\usepackage{hyperref}
\hypersetup{pdftitle={PPower4Example},
  pdfsubject={An example to demonstrate PPower4},
  pdfauthor={Klaus Guntermann, Systems Programming Group,
10     Darmstadt University of Technology,
11     <guntermann@iti.informatik.tu-darmstadt.de>},
12     pdfkeywords={pdftex, acrobat},
13     pdfpagemode={FullScreen}
14     }
15   \usepackage{pause}
16   \usepackage{background}
17   \usepackage{pdfslide}
18   \begin{document}
19   \definecolor{bgblue}{rgb}{0.04,0.39,0.53}
20   \vpagecolor{bgblue}
21
22   \foilhead{An example with colored background}
23   \begin{itemize}
24   \item Itemizations\pause
25     \begin{itemize}
26     \item are nested\pause
27     \item and labelled
28       \hypertarget{start}{} %mark any place on the page
29       \begin{itemize}
30       \item even nested deeply\pause
31       \end{itemize}
32     \item on different levels\pause
33     \end{itemize}
34   \item with formulas like $\sum_{i=0}^\infty a_i\cdot x^i$
35
36   \end{itemize}
37
38   \foilhead{Different page transitions}
39   \hypersetup{pdfpagetransition=Dissolve}
40   \begin{itemize}
41   \item Switch it on for a page\pause
42   \item But must switch off explicitely
43   \end{itemize}
44
45   \foilhead[2ex]{That's it}
46   \hypersetup{pdfpagetransition=R}
47   \begin{center}
48   Just a small example.
49
50   You may go \hyperlink{start}{back} to start.
51   \end{center}
52   \end{document}
```

This document uses rather large font sizes and an orientation suitable for presentation with a beamer. This configuration is made through the `geometry` package (lines 4 and 5). To color the pages we need the package `color` (lines 3 and 19). Additional information and navigational support is provided through the `hyperref` package (lines 6, 7, 28, 39, 46, and 50). Specifically for PPower4 we need `pause` and `background` (lines 15 and 16). Finally `pdfslide` will configure other packages and set up defaults (line 17). Thus it should be loaded last. Actually this package removes all bottom material from the `foils` page layout and sets default colors for the different parts. The details are as follows:

```
%% pdfslide.sty                                         30 Aug 99
%%-------------------------------------------------------------
%% Adapt foiltex to be used to prepare slides in pdf format
%% using backgrounds and partial builds.
%%
%% Disable some pdf commands, when not used in pdf(la)tex.
%%
%% Special (PDF) effects:
%% - make background blue, write normally in white and
%%   headings in yellow.
%% - use colorful labels in itemized lists.
%% - avoid footline usage, because that would interfere with
%%   partial builds of a page. Move the page number to the
%%   upper right corner. This means also suppression of Logo
%%   etc.
%% - make sure that links are displayed in text color, not
%%   with frames.
%% - Do not use paragraph indentation and justified text on
%%   slides (shouldn't that be the default?)
\rightfooter{} % no more page numbers bottom right
\MyLogo{} % no logo bottom left
\rightheader{\rlap{\quad\textsf{\tiny\thepage}}} % page number
\parindent 0pt % do not indent paragraphs
\rightskip 0pt plus 1fil % allow ragged right
%% select colors
\RequirePackage{color}
%%   for the frames/page numbers etc.
\renewcommand\Black{\color{white}}
%%   for the headline etc.
\renewcommand\normalcolor{\color{yellow}}
%%   for the background
\pagecolor{blue}
%%   for the text
\color{white}
%% choose some colored item labels
\renewcommand{\labelitemi}{\textcolor{red}{$\bullet$}}
\renewcommand{\labelitemii}{\textcolor{yellow}{$\star$}}
\renewcommand{\labelitemiii}{\textcolor{magenta}{$\ast$}}
\renewcommand{\labelitemiv}{\textcolor{cyan}{$\circ$}}
% Make hyperlinks colored, not framed, if hypersetup is used
\ifx\hypersetup\undefined\relax\else
\hypersetup{colorlinks=true}
\fi
```

In the body of the document the contents of the slides are written down. In all places, where the page build should stop, the command `\pause` must be inserted (lines 24, 26, 30, 32, and 41 of the example file).

The final result is shown in figure 3. All slides showing the partial builds are scaled down.

6 Summary

The implemented processor helps to present also scientific material with a large amount of formulas in a decent form. Further development can add functionality to increase dynamic aspects of a presentation. But careful selection is advised, not to distract the audience from the contents of the presentation by overwhelming them with fancy effects.

References

[Ado99] Adobe Systems Incorporated. *Portable Document Format Reference Manual.* Addison-Wesley, Reading, MA, USA, version 1.3 (march 11, 1999) edition, 1999.

[Hob92] J. D. Hobby. A user's manual for MetaPost. Technical Report 162, AT&T Bell Laboratories, 1992.

[Knu86] Donald E. Knuth. *The TEXbook*, volume A of *Computers and Typesetting.* Addison-Wesley, Reading, MA, USA, 1986.

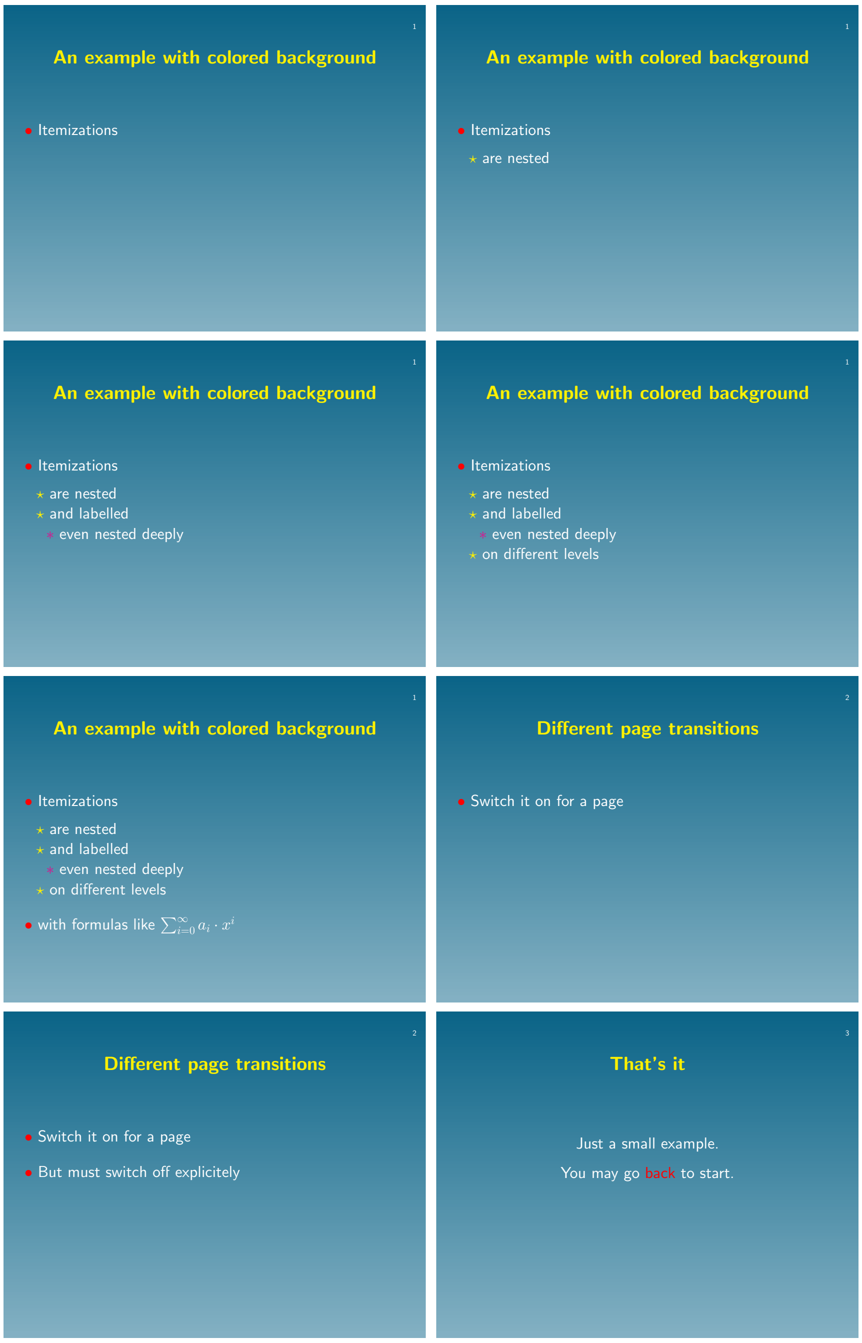
[Koh94] Eva Kohlberg. *Microsoft powerpoint für Windows.* Microsoft Press Deutschland, Unterschleissheim, 1994.

[Lam95] Leslie Lamport. *Das LATEX Handbuch.* Addison-Wesley, Reading, MA, USA, 1995.

[Nas98] Nassib Nassar. Automating pdf objects for interactive publishing. *Web Techniques Magazine*, 3(10), October 1998.

[SS] Brian W. Smith et al. Supoj Sutanthavibul. *Xfig user manual.* online at http://www.xfig.org.

[Wal91] Gregory Wallace. The JPEG still picture compression standard. *Comm. ACM*, 34(4):30–44, 1991.



1

An example with colored background

- Itemizations

1

An example with colored background

- Itemizations
  - ⋆ are nested

1

An example with colored background

- Itemizations
  - ⋆ are nested
  - ⋆ and labelled
    - ∗ even nested deeply

1

An example with colored background

- Itemizations
  - ⋆ are nested
  - ⋆ and labelled
    - ∗ even nested deeply
  - ⋆ on different levels

1

An example with colored background

- Itemizations
  - ⋆ are nested
  - ⋆ and labelled
    - ∗ even nested deeply
  - ⋆ on different levels
- with formulas like $\sum_{i=0}^{\infty} a_i \cdot x^i$

2

Different page transitions

- Switch it on for a page

2

Different page transitions

- Switch it on for a page
- But must switch off explicitely

3

That's it

Just a small example.

You may go back to start.

Figure 3: The complete example 2