Beamer v3.0 Guide

Ki-Joo Kim (a.k.a. Daisyweb)

November 4, 2004

Why Beamer?

Pros:

- Both `dvips/ps2pdf`[1] and `pdflatex` supports[2]
- Rich overlay and transition effects
- Navigational bars and symbols
- Outputs: screen, transparency, handouts, and notes
- Emulation of other PDF presentation tools such as *Prosper* and *FoilTEX*

Cons:

- Difficult to design a template

[1] You need this route if you use PSTricks.
[2] No dvipdfm support!

Basic Code I

- Beamer class loading with themes

```
\documentclass[slidestop,compress,mathserif]{beamer}
%\usepackage[bars]{beamerthemetree} % Beamer theme v 2.2
\usetheme{Antibes}                  % Beamer theme v 3.0
\usecolortheme{lily}                % Beamer color theme
```

Basic Code I

- Beamer class loading with themes

```
\documentclass[slidestop,compress,mathserif]{beamer}
%\usepackage[bars]{beamerthemetree} % Beamer theme v 2.2
\usetheme{Antibes}                  % Beamer theme v 3.0
\usecolortheme{lily}                % Beamer color theme
```

- Cover title

```
\title{}
\author{}
\institute{}
\begin{document}
\begin{frame}                       % Cover slide
 \titlepage
\end{frame}
% Instead, you can use \frame{\titlepage}} (Beamer v 2.2 macro)
```

Basic Code II

- Main slide frame

```
\section{Introduction}                % Bookmark information
\subsection{History}                  % Bookmark information
\begin{frame}[options]
 \frametitle{History}
 ... slide contents ...
\end{frame}
```

With v 2.2 macro:

```
\frame[options]{\frametitle{History}%
 ... slide contents ...
}%
```

Five Themes

- The main difference between v 3.0 and v 2.2 is *Beamer themes*.
- Five theme categories:
  - Presentation Themes – *Slide template*
  - Color Themes – *Color scheme for slide template*
  - Font Themes
  - Inner Themes
  - Outer Themes
- Example

```
\documentclass[slidestop,compress,mathserif]{beamer}
%\usepackage[bars]{beamerthemetree} % Beamer theme v 2.2
\usetheme{Antibes}                  % Beamer theme v 3.0
\usecolortheme{lily}                % Beamer color theme
```

- Go to ▸ Themes for more information.

Beamer Options for Layout

- `[slidestop]` puts frame titles on the top left corner (default=`[slidescentered]`).
- `[compress]` makes all navigation bars as small as possible (default=`[uncompressed]`).
- `[red]` changes navigation bars and titles to reddish color.
  - `blue`: Default color scheme
  - `red`: Used in this presentation
  - `brown`
  - `blackandwhite`: Good for transparencies

Beamer Options for Output

- Default: PDF screen (size 128mm $\times$ 96 mm)[3].
- `[handout]` for PDF handouts.
- `[trans]` for PDF transparency.

$\Rightarrow$ For handout and trans, you need some extra work to enlarge the size. Click here to see an example!

- `[notes=hide/show/only]` for notes. Hide notes (default), add notes to the PDF screen, or notes only PDF.

[3] Monitor's 4:3 aspect ratio.

Additional Beamer Options

- `[hyperref={bookmarks=false}]` removes bookmarks.
- `[cjk]` for CJK typesetting. ⇒ For hangul, use hfont package.
- `\usepackage[utf8]{inputenc}` for Unicode.

Frame Options

- `\frame[plain]{\frametitle{}..}` for plane frame style as *used in this slide*!
- `[containsverbatim]` for using `verbatim` environment and `\verb` command.
- `[allowframebreaks]` for automatic split of frames if the contents do not fit in a single slide.
- `[shrink]` for shrinking the contents to fit in a single slide.
- `[squeeze]` for squeezing vertical space.

Transparency Effects

- All overlayed stuffs are covered (default)
- `\beamertemplatetransparentcoveredhigh` makes all covered text highly transparent
- `\beamertemplatetransparentcovereddynamicmedium` makes all covered text quite transparent, but is a dynamic way. The range of dynamics is smaller.

Text and Math Fonts I

- Excellent support for selecting text and math fonts.
- Default text and math fonts: CMSS and CMR Math
  - You should *avoid* CMR Math in presentation
  - Example: `http://faq.ktug.or.kr/wiki/uploads/MathFonts.pdf`
- Beamer option `[sans]` for text font (default)
  - `mathsans` is default.
  - Equivalent to `\usefonttheme{default}`
- Beamer option `[sefif]` for text font
  - `mathserif` is default.
  - Equivalent to `\usefonttheme[options]{serif}`
- Beamer option `[mathsans/mathserif]` for math font

Text and Math Fonts II

- Beamer option [professionalfonts] to turn off Beamer's internal font rewriting (⇒ Equivalent to \usefonttheme{professionalfonts})
- Additional font theme macros
  - \usefonttheme{structurebold} for bold faced structures (titles, headlines, footlines, sidebars, ...)
  - \usefonttheme{structureitalicserif}
  - \usefonttheme{structuresmallcapsserif}
- Font settings in this document:

```
\documentclass[mathserif]{beamer}  % sans (text) + mathserif
\usepackage{lucidaso}              % Lucida Bright (SO Version)
\usepackage[small]{eulervm}        % Euler VM
```

Font Size

- Default font size: 11pt (At the full screen mode this font size corresponds to 22 pt.)
- Available font size options: 8pt, 9pt, 10pt, 11pt, 12pt, 14pt, 17pt, 20pt

Color Definition

- Beamer loads xcolor package by Uwe Kern, which also supports color and pstcol.
- 'xcolor' definition
  - `\xdefinecolor{lavendar}{rgb}{0.8,0.6,1}`
  - `\xdefinecolor{olive}{cmyk}{0.64,0,0.95,0.4}`
  - `\colorlet{structure}{green!60!black}` for color substitution
  - Predefined colors: red, green, blue, cyan, magenta, yellow, black, darkgray, gray, lightgray, orange, violet, purple, and brown
- If you want to use the options of 'color' package, pass `[color=option]` option to Beamer.
- If you want to use 'pstcol', pass `[xcolor=pst,dvips]` option to Beamer. Now you should use '`dvips/ps2pdf`'

More colors in ‘xcolor’ package

- Color mixing is very easy!

color	example	meaning
`green!80!gray`	text	80% green + 20% gray
`green!60!gray`	text	60% green + 40% gray
`green!40!gray`	text	40% green + 60% gray
`-green`	text	remove green from above

- You can use `animate` (Beamer macro) or `multido` (PSTricks macro) for fade-in and fade-out!

Highlighting Colors

- Beamer also has theme-specific highlighting colors:
  - `\alert{text}` ⇒ text
  - `\structure{text}` ⇒ text
- To change these colors:
  - `\usecolortheme[named=yellow]{structure}` to change to yellow.
  - `\setbeamercolor{alerted_text}{fg=cyan}`[4] to change to cyan.

[4] '_' means space.

Background Colors

- To set solid background color, `\beamersetaveragebackground{color}` or `\beamertemplatesolidbackgroundcolor{color}`
- To set gradient background color, `\beamertemplateshadingbackground{color1}{color2}`. ⇒ The colors in this slide is `{blue!5}{yellow!10}`.
- To set grid background, `\beamertemplategridbackground[grid_space]`.

Color Example

- Color changes in
  - Navigational bars
  - Background
  - `structure{..}` color

Color Example

- Color changes in
  - Navigational bars
  - Background
  - `structure{..}` color
- Code:

```
\colorlet{mystruct}{structure}                    % Save current structure
\colorlet{structure}{magenta}                      % New structure
\usestructuretemplate{\color{structure}}{}         % \structure{..}
\beamertemplateshadingbackground{yellow!50}{magenta!50} % New background
\frame{%
 ...
}%
% Back to the original "structure" and bg color schemes
\colorlet{structure}{mystruct}
\beamertemplateshadingbackground{blue!10}{yellow!10}
```

Verbatim w/o Overlays

- '\verb' or 'verbatim' cannot be *directly* used in Beamer!
- If there is no overlay, use \frame[containsverbatim]

```
\frame[containsverbatim]{\frametitle{..}%
  \begin{verbatim}
   .. verbatim contents ..
  \end{verbatim}
}%
```

- Now in-line verbatim is possible with '\verb'.
- Color and size can be easily changed.

Inline Verbatim with Overlays

- My solution: `\path{..}` instead of `\verb`.
  - Color: `Hello`, `Hello`
  - Size: `Hello`, `Hello`, `Hello`

Inline Verbatim with Overlays

- My solution: `\path{..}` instead of `\verb`.
  - Color: `Hello`, `Hello`
  - Size: `Hello`, `Hello`, `Hello`
- Beamer solution: `\defverb\command|contents|` outside the frame.
  - Define `\defverb\myverb|Hello,World!|`
  - Then use `\myverb` ⇒ `Hello, World!`

Verbatim with Overlays

- Use *lstlisting* environment instead of *verbatim* environment.
- Define `\defverbatim[colored]\command{contents}` outside frame.
- '`contents`' are the `listing` environment.

Verbatim with Overlays

- Use *lstlisting* environment instead of *verbatim* environment.
- Define `\defverbatim[colored]\command{contents}` outside frame.
- 'contents' are the `listing` environment.
- Example:

```
\defverbatim[colored]\testcode{%
  \begin{lstlisting}[frame=single,emph={ga},emphstyle=\color{olive}]
    ....
  \end{lstlisting}}%
\frame{%
  \testcode
}%
```

Figures Intro

- Standard LaTeX `figure` environment can be used.
- Beamer also loads *pgf* package. So PGF command, `\pgfimage[]{file}`, is also possible.
- `\includegraphics`, `\pgfimage`, and `\pdfuseimage` understand overlays.

Figure: Tiger

PGF Macros

- PSTricks and PGF can be used for locating figures *exactly* .
- Grid size of slide: (LL $\times$ UR) = (0cm,-7cm) $\times$ (11cm,1cm)
- PGF macro for locating figures:

```
\pgfputat{\pgfxy(0,-6.5)}{\pgfbox[left,base]{\pgfimage[width=1cm]{tiger}}}
```

- If you use the same figure several times, use `\pgfdecalreimage` and `\pgfuseimage`. Or just use `\includegraphics`.

Figures inside Columns

- Figures inside ‘columns’ environment need exact position.

Figures inside Columns

- Figures inside ‘columns’ environment need exact position.
- PGF macros (PDF, PNG, and JPEG with `pdflatex`)

```
\begin{columns}
\begin{column}{0.65\textwidth}
  A\\B
\end{column}
\begin{column}{0.35\textwidth}
  \pgfputat{\pgfxy(0,0)}{\pgfbox[left,top]{\includegraphics[width=\textwidth]{tig
\end{column}
\end{columns}
```

Figures inside Columns

- Figures inside ‘columns’ environment need exact position.
- PGF macros (PDF, PNG, and JPEG with `pdflatex`)

```
\begin{columns}
 \begin{column}{0.65\textwidth}
   A\\B
 \end{column}
 \begin{column}{0.35\textwidth}
   \pgfputat{\pgfxy(0,0)}{\pgfbox[left,top]{\includegraphics[width=\textwidth]{tig
 \end{column}
\end{columns}
```

- PSTricks macros (EPS with `dvips`)

```
\begin{columns}
 \begin{column}{0.65\textwidth}
  A\\B
 \end{column}
 \begin{column}{0.35\textwidth}
  \rput[lt](0,0){\includegraphics[clip=true,width=\textwidth]{tiger}}
 \end{column}
\end{columns}
```

Zooming Figures

- Figures can be zoomed[5] using
  `\framezoom<button overlay><zoomed overlay>(x,y)(w,h)`.
- `(x,y)`: Upper left coordinate point
  `(w,h)`: Width and height for zooming
- Example:

```
\frame{\frametitle{Zooming Figures -- Example}
 \framezoom<1><2>[border](0.5cm,0.5cm)(2cm,1.5cm)
 \framezoom<1><3>[border](1cm,3cm)(2cm,1.5cm)
 \framezoom<1><4>[border](3cm,2cm)(2cm,2cm)

 \pgfimage[height=6cm]{tiger}
 %\includegraphics[height=6cm]{tiger} is working, too!
}%
```

[5] New in Version 2.2

Zooming Figures – Example

Click the border to zoom-in.

Zooming Figures – Example

Zooming Figures – Example

Zooming Figures – Example

Drawing Figures

- The most powerful and easiest-to-use package, *PSTricks*, does not work with `pdflatex` due to fundamental differences in PS and PDF.

[6] Note that Beamer does not support `dvipdfm`.

Drawing Figures

- The most powerful and easiest-to-use package, *PSTricks*, does not work with `pdflatex` due to fundamental differences in PS and PDF.
- *PGF* (portable graphics format) by the Beamer author.
  - Less powerful than PSTricks, but works fine.
  - Supports `dvips`, `dvipdfm`[6], and `pdflatex`.

[6] Note that Beamer does not support `dvipdfm`.

Drawing Figures

- The most powerful and easiest-to-use package, *PSTricks*, does not work with `pdflatex` due to fundamental differences in PS and PDF.
- *PGF* (portable graphics format) by the Beamer author.
  - Less powerful than PSTricks, but works fine.
  - Supports `dvips`, `dvipdfm`[6], and `pdflatex`.
- *MetaPost*
  - Works with `dvips/ps2pdf`, `dvipdfm`, and `pdflatex`

[6] Note that Beamer does not support `dvipdfm`.

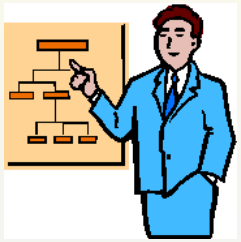
Drawing Figures

- The most powerful and easiest-to-use package, *PSTricks*, does not work with `pdflatex` due to fundamental differences in PS and PDF.
- *PGF* (portable graphics format) by the Beamer author.
  - Less powerful than PSTricks, but works fine.
  - Supports `dvips`, `dvipdfm`[6], and `pdflatex`.
- *MetaPost*
  - Works with `dvips/ps2pdf`, `dvipdfm`, and `pdflatex`
- I prefer *Beamer + PSTricks*.
  ⇒ See beamer_pstricks.pdf [1]

[6] Note that Beamer does not support `dvipdfm`.

Masking Figures

- Want to mask *white background* of your images?

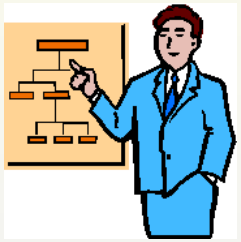


+ =

[7] I do not know the exact requirement.

Masking Figures

- Want to mask *white background* of your images?


+

=

- Make a mask image in 256 Colors and JPEG Compression[7]

[7] I do not know the exact requirement.

Masking Figures

- Want to mask *white background* of your images?

- Make a mask image in 256 Colors and JPEG Compression[7]
- Use `\pgfdeclaremask` in pdf package. But only works with pdflatex!

[7] I do not know the exact requirement.

Masking Figures

- Want to mask *white background* of your images?

- Make a mask image in 256 Colors and JPEG Compression[7]
- Use `\pgfdeclaremask` in pdf package. But only works with pdflatex!
- Source code:

```
\pgfdeclaremask{mymask}{ppt.mask}          % Mask image: ppt.mask.jpg
\pgfimage[mask=mymask,interpolate=true]{ppt}      % Masking ppt.png
```

[7] I do not know the exact requirement.

Masking Figures

- Want to mask *white background* of your images?

- Make a mask image in 256 Colors and JPEG Compression[7]
- Use `\pgfdeclaremask` in pdf package. But only works with pdflatex!
- Source code:

```
\pgfdeclaremask{mymask}{ppt.mask}          % Mask image: ppt.mask.jpg
\pgfimage[mask=mymask,interpolate=true]{ppt}      % Masking ppt.png
```

- But the mask image masks the whole slide! See the font outlines.

[7] I do not know the exact requirement.

Fancy Bullets

1. `\beamertemplateballitem` in the preamble
2. `itemize` environment ⇒ Fancy ball
3. `enumerate` environment ⇒ Fancy numbered ball (used here).

Fancy Bullets

1. `\beamertemplateballitem` in the preamble
2. `itemize` environment ⇒ Fancy ball
3. `enumerate` environment ⇒ Fancy numbered ball (used here).

To use different enumerate templates,

```
\begin{enumerate}[minitemplate]
  \item ...
\end{enumerate}
```

where mini template can be 'A', 'a', 'i', 'I', '(A)', ... But the indentation may be changed (bug?)

i Item 1

ii Item 2

Framed Text – Predefined

- Beamer supports predefined framed texts:
  - `theorem`, `corollary`, `definition` in structure color frame
  - `examples` in green color frame
  - `block` in structure color frame with your own title
  - `alertblock` in alert color frame with your own title

Framed Text – Predefined

- Beamer supports predefined framed texts:
  - `theorem`, `corollary`, `definition` in structure color frame
  - `examples` in green color frame
  - `block` in structure color frame with your own title
  - `alertblock` in alert color frame with your own title
- They are aware of overlay
- But their color schemes are *theme dependent*

Framed Text – Predefined

- Beamer supports predefined framed texts:
  - `theorem`, `corollary`, `definition` in structure color frame
  - `examples` in green color frame
  - `block` in structure color frame with your own title
  - `alertblock` in alert color frame with your own title
- They are aware of overlay
- But their color schemes are *theme dependent*
- Example:

Summary

Beamer is excellent!

Framed Text – Predefined

- Beamer supports predefined framed texts:
  - `theorem`, `corollary`, `definition` in structure color frame
  - `examples` in green color frame
  - `block` in structure color frame with your own title
  - `alertblock` in alert color frame with your own title
- They are aware of overlay
- But their color schemes are *theme dependent*
- Example:

Summary

Beamer is excellent!

- Sample code:

```
\begin{block}<+->{Summary}
 Beamer is exllent
\end{block}
```

Framed Text – User-defined

- `beamerboxesrounded` environment
- Example

Theorem

$$A = B$$

$$B = C$$

$\Rightarrow$

Theorem

$$A = C?$$

Framed Text – User-defined

- `beamerboxesrounded` environment
- Example

Theorem

$A = B$

$B = C$

$\Rightarrow$

Theorem

$A = C$?

- Source Code:

```
\setbeamercolor{uppercol}{fg=white,bg=ugreen}%
\setbeamercolor{lowercol}{fg=black,bg=lgreen}%
\begin{beamerboxesrounded}[upper=uppercol,lower=lowercol,shadow=true]{Theorem}
  $A = B$.
\end{beamerboxesrounded}}
```

Columns

- Use LaTeX `minipage` environment or
- Use Beamer `columns` environment

```
\begin{columns}
 \begin{column}[pos]{width}
   ... contents ...
 \end{column}
 \begin{column}[pos]{width}
  ... contents ...
 \end{column}
\end{columns}
```

Tables

- Standard LaTeX `table` environment can be used.
- `\onslide` inside '`overprint`' environment for showing overlays in the right example.

Table Overlays:

Tables

- Standard LaTeX `table` environment can be used.
- `\onslide` inside '`overprint`' environment for showing overlays in the right example.

Table Overlays:

Cells	are	growing

Tables

- Standard LaTeX `table` environment can be used.
- `\onslide` inside '`overprint`' environment for showing overlays in the right example.

Table Overlays:

Cells	are	growing
step	by	

Tables

- Standard LaTeX `table` environment can be used.
- `\onslide` inside '`overprint`' environment for showing overlays in the right example.

Table Overlays:

Cells	are	growing
step	by	
step.		

Tables

- Standard LaTeX `table` environment can be used.
- `\onslide` inside '`overprint`' environment for showing overlays in the right example.

Table Overlays:

Cells	are	growing
step	by	
step.		Finished!

Transitions

- PDF supports seven transitions: Blinds, Box, Dissolve, Glitter, Replace, Split, Wipe.
- Transition commands are inside `frame` environment.
- Beamer transition commands understand overlays. Without overlays the transition is *global* to the current frame.

Transitions

- PDF supports seven transitions: Blinds, Box, Dissolve, Glitter, Replace, Split, Wipe.
- Transition commands are inside `frame` environment.
- Beamer transition commands understand overlays. Without overlays the transition is *global* to the current frame.
- Overlayed transition examples:
  - Glitter at /Di 315 (default on this slide): `\transglitter[direction=315]`

Transitions

- PDF supports seven transitions: Blinds, Box, Dissolve, Glitter, Replace, Split, Wipe.
- Transition commands are inside `frame` environment.
- Beamer transition commands understand overlays. Without overlays the transition is *global* to the current frame.
- Overlayed transition examples:
  - Glitter at /Di 315 (default on this slide): `\transglitter[direction=315]`
  - Boxout ..... `\transboxout<3>`

Transitions

- PDF supports seven transitions: Blinds, Box, Dissolve, Glitter, Replace, Split, Wipe.
- Transition commands are inside `frame` environment.
- Beamer transition commands understand overlays. Without overlays the transition is *global* to the current frame.
- Overlayed transition examples:
  - Glitter at /Di 315 (default on this slide): `\transglitter[direction=315]`
  - Boxout .....`\transboxout<3>`
  - .....Boxin: `\transboxin<4>` .....

Transitions

- PDF supports seven transitions: Blinds, Box, Dissolve, Glitter, Replace, Split, Wipe.
- Transition commands are inside `frame` environment.
- Beamer transition commands understand overlays. Without overlays the transition is *global* to the current frame.
- Overlayed transition examples:
  - Glitter at /Di 315 (default on this slide): `\transglitter[direction=315]`
  - Boxout .....`\transboxout<3>`
  - .....Boxin: `\transboxin<4>` .....
  - Dissolve transition: `\transdissolve<5>`

Transitions

- PDF supports seven transitions: Blinds, Box, Dissolve, Glitter, Replace, Split, Wipe.
- Transition commands are inside `frame` environment.
- Beamer transition commands understand overlays. Without overlays the transition is *global* to the current frame.
- Overlayed transition examples:
  - Glitter at /Di 315 (default on this slide): `\transglitter[direction=315]`
  - Boxout .....`\transboxout<3>`
  - .....Boxin: `\transboxin<4>` .....
  - Dissolve transition: `\transdissolve<5>`
  - Split vertical out: `\transsplitverticalout<6>`

Overlays - Overview

- Overlays is the heart of dynamic PDF presentation.
- Beamer provides plenty of overlay commands.

Overlays - Overview

- Overlays is the heart of dynamic PDF presentation.
- Beamer provides plenty of overlay commands.
- Key overlay functions are:
  - Stepwise viewing
  - Replace
  - Highlighting

Overlays - Overview

- Overlays is the heart of dynamic PDF presentation.
- Beamer provides plenty of overlay commands.
- Key overlay functions are:
  - Stepwise viewing
  - Replace
  - Highlighting
- Various overlay counters: ‘n’, ‘n-’, ‘-n’, ‘n1-n2’, ‘+-’.

Pause for Stepwise Viewing

`pause` command[8] for easy and simple overlays.

```
\begin{itemize}
\pause \item Every thing
\pause \item that has
\pause \item beginning
\pause \item has end.
\end{itemize}
```

[8] There is also `\unpause` command.

Pause for Stepwise Viewing

`pause` command[8] for easy and simple overlays.

```
\begin{itemize}
\pause \item Every thing
\pause \item that has
\pause \item beginning
\pause \item has end.
\end{itemize}
```

- Every thing

[8] There is also `\unpause` command.

Pause for Stepwise Viewing

`pause` command[8] for easy and simple overlays.

```
\begin{itemize}
\pause \item Every thing
\pause \item that has
\pause \item beginning
\pause \item has end.
\end{itemize}
```

- Every thing
- that has

[8] There is also `\unpause` command.

Pause for Stepwise Viewing

`pause` command[8] for easy and simple overlays.

```
\begin{itemize}
\pause \item Every thing
\pause \item that has
\pause \item beginning
\pause \item has end.
\end{itemize}
```

- Every thing
- that has
- beginning

[8] There is also `\unpause` command.

Pause for Stepwise Viewing

`pause` command[8] for easy and simple overlays.

```
\begin{itemize}
\pause \item Every thing
\pause \item that has
\pause \item beginning
\pause \item has end.
\end{itemize}
```

- Every thing
- that has
- beginning
- has end.

[8] There is also `\unpause` command.

Pause for Stepwise Viewing

`pause` command[8] for easy and simple overlays.

```
\begin{itemize}
\pause \item Every thing
\pause \item that has
\pause \item beginning
\pause \item has end.
\end{itemize}
```

- Every thing
- that has
- beginning
- has end.

Note that `pause` does not know overlay counters.

[8] There is also `\unpause` command.

Pause: Table Example

- Row increment in a table:

Pause: Table Example

- Row increment in a table:

Class	A	B	C	D
X	1	2	3	4

Pause: Table Example

- Row increment in a table:

Class	A	B	C	D
X	1	2	3	4
Y	3	4	5	6

Pause: Table Example

- Row increment in a table:

Class	A	B	C	D
X	1	2	3	4
Y	3	4	5	6
Z	5	6	7	8

Pause: Table Example

- Row increment in a table:

Class	A	B	C	D
X	1	2	3	4
Y	3	4	5	6
Z	5	6	7	8

- Source code:

```
\rowcolors[]{1}{blue!20}{blue!10}
\begin{tabular}{l!{\vrule}cccc}
 Class & A & B & C & D \\\hline
 X & 1 & 2 & 3 & 4 \pause \\
 Y & 3 & 4 & 5 & 6 \pause \\
 Z & 5 & 6 & 7 & 8
\end{tabular}
```

Onslide for Stepwise Viewing

- `\onslide<n->stuff` shows stuff on the given slides.

Onslide for Stepwise Viewing

- `\onslide<n->stuff` shows stuff on the given slides.
- Example: Column increment in a table:

Class	A
X	1
Y	3
Z	5

Onslide for Stepwise Viewing

- `\onslide<n->stuff` shows stuff on the given slides.
- Example: Column increment in a table:

Class	A	B
X	1	2
Y	3	4
Z	5	6

Onslide for Stepwise Viewing

- `\onslide<n->stuff` shows stuff on the given slides.
- Example: Column increment in a table:

Class	A	B	C
X	1	2	3
Y	3	4	5
Z	5	6	7

Onslide for Stepwise Viewing

- `\onslide<n->stuff` shows stuff on the given slides.
- Example: Column increment in a table:

Class	A	B	C	D
X	1	2	3	4
Y	3	4	5	6
Z	5	6	7	8

Onslide for Stepwise Viewing

- `\onslide<n->stuff` shows stuff on the given slides.
- Example: Column increment in a table:

Class	A	B	C	D
X	1	2	3	4
Y	3	4	5	6
Z	5	6	7	8

- Source code:

```
\rowcolors[]{1}{blue!20}{blue!10}
\begin{tabular}{l!{\vrule}c<{\onslide<2->}c<{\onslide<3->} %
                c<{\onslide<4->}c<{\onslide}c}
  Class & A & B & C & D \\
  X & 1 & 2 & 3 & 4 \\
  Y & 3 & 4 & 5 & 6 \\
  Z & 5 & 6 & 7 & 8
\end{tabular}
```

Item I for Stepwise Viewing

`\item<n->` for incremental overlays with overlay counters.

```
\begin{itemize}
\item<2-> Every thing
\item<3-> that has
\item<4-> beginning
\item<5-> has end.
\end{itemize}
```

Item I for Stepwise Viewing

`\item<n->` for incremental overlays with overlay counters.

```
\begin{itemize}
\item<2-> Every thing
\item<3-> that has
\item<4-> beginning
\item<5-> has end.
\end{itemize}
```

- Everything

Item I for Stepwise Viewing

`\item<n->` for incremental overlays with overlay counters.

```
\begin{itemize}
\item<2-> Every thing
\item<3-> that has
\item<4-> beginning
\item<5-> has end.
\end{itemize}
```

- Everything
- that has

Item I for Stepwise Viewing

`\item<n->` for incremental overlays with overlay counters.

```
\begin{itemize}
\item<2-> Every thing
\item<3-> that has
\item<4-> beginning
\item<5-> has end.
\end{itemize}
```

- Everything
- that has
- beginning

Item I for Stepwise Viewing

`\item<n->` for incremental overlays with overlay counters.

```
\begin{itemize}
\item<2-> Every thing
\item<3-> that has
\item<4-> beginning
\item<5-> has end.
\end{itemize}
```

- Everything
- that has
- beginning
- has end.

Item I for Stepwise Viewing

`\item<n->` for incremental overlays with overlay counters.

```
\begin{itemize}
\item<2-> Every thing
\item<3-> that has
\item<4-> beginning
\item<5-> has end.
\end{itemize}
```

- Everything
- that has
- beginning
- has end.

What if more items are *inserted*?

Item II for Stepwise Viewing

<+-> for incremental overlays w/o overlay counters.

```
\begin{itemize}[<+->]
\item Every thing
\item that has
\item beginning
\item has end.
\end{itemize}
```

Item II for Stepwise Viewing

<+-> for incremental overlays w/o overlay counters.

```
\begin{itemize}[<+->]
\item Every thing
\item that has
\item beginning
\item has end.
\end{itemize}
```

- Everything

Item II for Stepwise Viewing

<+-> for incremental overlays w/o overlay counters.

```
\begin{itemize}[<+->]
\item Every thing
\item that has
\item beginning
\item has end.
\end{itemize}
```

- Everything
- that has

Item II for Stepwise Viewing

<+-> for incremental overlays w/o overlay counters.

```
\begin{itemize}[<+->]
\item Every thing
\item that has
\item beginning
\item has end.
\end{itemize}
```

- Everything
- that has
- beginning

Item II for Stepwise Viewing

<+-> for incremental overlays w/o overlay counters.

```
\begin{itemize}[<+->]
\item Every thing
\item that has
\item beginning
\item has end.
\end{itemize}
```

- Everything
- that has
- beginning
- has end.

Item II for Stepwise Viewing

<+-> for incremental overlays w/o overlay counters.

```
\begin{itemize}[<+->]
\item Every thing
\item that has
\item beginning
\item has end.
\end{itemize}
```

- Everything
- that has
- beginning
- has end.

Note that \item<+-> can be used instead of global setting of \begin{itemize}[<+->].

Item III for Stepwise Viewing

`\item<n1-n2>` for fine control of overlays.

```
\begin{itemize}
\item<1-> Every thing
\item<3-4> that has
\item<4> beginning
\item<2-5> has end.
\end{itemize}
```

Item III for Stepwise Viewing

`\item<n1-n2>` for fine control of overlays.

```
\begin{itemize}
\item<1-> Every thing
\item<3-4> that has
\item<4> beginning
\item<2-5> has end.
\end{itemize}
```

- Everything

Item III for Stepwise Viewing

`\item<n1-n2>` for fine control of overlays.

```
\begin{itemize}
\item<1-> Every thing
\item<3-4> that has
\item<4> beginning
\item<2-5> has end.
\end{itemize}
```

- Everything
- has end.

Item III for Stepwise Viewing

`\item<n1-n2>` for fine control of overlays.

```
\begin{itemize}
\item<1-> Every thing
\item<3-4> that has
\item<4> beginning
\item<2-5> has end.
\end{itemize}
```

- Everything
- that has
- has end.

Item III for Stepwise Viewing

`\item<n1-n2>` for fine control of overlays.

```
\begin{itemize}
\item<1-> Every thing
\item<3-4> that has
\item<4> beginning
\item<2-5> has end.
\end{itemize}
```

- Everything
- that has
- beginning
- has end.

Item III for Stepwise Viewing

`\item<n1-n2>` for fine control of overlays.

```
\begin{itemize}
\item<1-> Every thing
\item<3-4> that has
\item<4> beginning
\item<2-5> has end.
\end{itemize}
```

- Everything
- has end.

Replace

- Successive `\only<n>{..}`.

  (Ex) `\only<1>{GA}\only<2>{MOGA}\only<3>{pMOGA}` ⇒ GA

Slide 1

[9] See also highlighting section.

Replace

- Successive `\only<n>{..}`.

  (Ex) `\only<1>{GA}\only<2>{MOGA}\only<3>{pMOGA}` ⇒ MOGA

[9] See also highlighting section.

Replace

- Successive `\only<n>{..}`.

  (Ex) `\only<1>{GA}\only<2>{MOGA}\only<3>{pMOGA}` ⇒ pMOGA

[9] See also highlighting section.

Replace

- Successive `\only<n>{..}`.

  (Ex) `\only<1>{GA}\only<2>{MOGA}\only<3>{pMOGA}` ⇒ pMOGA

- `\uncover<n>{..}` shows at given n.

  (Ex) `\uncover<5>{I am 5}` ⇒

[9] See also highlighting section.

Replace

- Successive \only<n>{..}.

  (Ex) \only<1>{GA}\only<2>{MOGA}\only<3>{pMOGA} ⇒ pMOGA

- \uncover<n>{..} shows at given n.

  (Ex) \uncover<5>{I am 5} ⇒ I am 5

[9] See also highlighting section.

Replace

- Successive `\only<n>{..}`.

  (Ex) `\only<1>{GA}\only<2>{MOGA}\only<3>{pMOGA}` ⇒ pMOGA

- `\uncover<n>{..}` shows at given n.

  (Ex) `\uncover<5>{I am 5}` ⇒

[9] See also highlighting section.

Replace

- Successive `\only<n>{..}`.

  (Ex) `\only<1>{GA}\only<2>{MOGA}\only<3>{pMOGA}` ⇒ pMOGA
- `\uncover<n>{..}` shows at given n.

  (Ex) `\uncover<5>{I am 5}` ⇒
- `\invisible<n>{..}` hides at given n.

  (Ex) `\invisible<8>{Invisible at 8}` ⇒ Invisible at 8

[9] See also highlighting section.

Replace

- Successive `\only<n>{..}`.

  (Ex) `\only<1>{GA}\only<2>{MOGA}\only<3>{pMOGA}` ⇒ pMOGA
- `\uncover<n>{..}` shows at given n.

  (Ex) `\uncover<5>{I am 5}` ⇒
- `\invisible<n>{..}` hides at given n.

  (Ex) `\invisible<8>{Invisible at 8}` ⇒

[9] See also highlighting section.

Replace

- Successive `\only<n>{..}`.
  (Ex) `\only<1>{GA}\only<2>{MOGA}\only<3>{pMOGA}` ⇒ pMOGA
- `\uncover<n>{..}` shows at given n.
  (Ex) `\uncover<5>{I am 5}` ⇒
- `\invisible<n>{..}` hides at given n.
  (Ex) `\invisible<8>{Invisible at 8}` ⇒ Invisible at 8

[9] See also highlighting section.

Replace

- Successive `\only<n>{..}`.
  (Ex) `\only<1>{GA}\only<2>{MOGA}\only<3>{pMOGA}` ⇒ pMOGA
- `\uncover<n>{..}` shows at given n.
  (Ex) `\uncover<5>{I am 5}` ⇒
- `\invisible<n>{..}` hides at given n.
  (Ex) `\invisible<8>{Invisible at 8}` ⇒ Invisible at 8
- `\alt<n>{at n}{not at n}` for two alternatives.
  (Ex) `\alt<11>{I am 11}{I am not 11}` ⇒ I am not 11

[9] See also highlighting section.

Replace

- Successive `\only<n>{..}`.
  (Ex) `\only<1>{GA}\only<2>{MOGA}\only<3>{pMOGA}` ⇒ pMOGA
- `\uncover<n>{..}` shows at given n.
  (Ex) `\uncover<5>{I am 5}` ⇒
- `\invisible<n>{..}` hides at given n.
  (Ex) `\invisible<8>{Invisible at 8}` ⇒ Invisible at 8
- `\alt<n>{at n}{not at n}` for two alternatives.
  (Ex) `\alt<11>{I am 11}{I am not 11}` ⇒ I am 11

[9] See also highlighting section.

Replace

- Successive `\only<n>{..}`.
  (Ex) `\only<1>{GA}\only<2>{MOGA}\only<3>{pMOGA}` ⇒ pMOGA
- `\uncover<n>{..}` shows at given n.
  (Ex) `\uncover<5>{I am 5}` ⇒
- `\invisible<n>{..}` hides at given n.
  (Ex) `\invisible<8>{Invisible at 8}` ⇒ Invisible at 8
- `\alt<n>{at n}{not at n}` for two alternatives.
  (Ex) `\alt<11>{I am 11}{I am not 11}` ⇒ I am not 11

[9] See also highlighting section.

Replace

- Successive `\only<n>{..}`.
  (Ex) `\only<1>{GA}\only<2>{MOGA}\only<3>{pMOGA}` ⇒ pMOGA
- `\uncover<n>{..}` shows at given n.
  (Ex) `\uncover<5>{I am 5}` ⇒
- `\invisible<n>{..}` hides at given n.
  (Ex) `\invisible<8>{Invisible at 8}` ⇒ Invisible at 8
- `\alt<n>{at n}{not at n}` for two alternatives.
  (Ex) `\alt<11>{I am 11}{I am not 11}` ⇒ I am not 11
- `\temporal<n>{before}{at n}{after}` for three alternatives.[9]
  (Ex) `\temporal<14>{I am 13}{I am 14}{I am 15}` ⇒ I am 13

[9] See also highlighting section.

Replace

- Successive `\only<n>{..}`.
  (Ex) `\only<1>{GA}\only<2>{MOGA}\only<3>{pMOGA}` ⇒ pMOGA
- `\uncover<n>{..}` shows at given n.
  (Ex) `\uncover<5>{I am 5}` ⇒
- `\invisible<n>{..}` hides at given n.
  (Ex) `\invisible<8>{Invisible at 8}` ⇒ Invisible at 8
- `\alt<n>{at n}{not at n}` for two alternatives.
  (Ex) `\alt<11>{I am 11}{I am not 11}` ⇒ I am not 11
- `\temporal<n>{before}{at n}{after}` for three alternatives.[9]
  (Ex) `\temporal<14>{I am 13}{I am 14}{I am 15}` ⇒ I am 14

[9] See also highlighting section.

Replace

- Successive `\only<n>{..}`.
  (Ex) `\only<1>{GA}\only<2>{MOGA}\only<3>{pMOGA}` ⇒ pMOGA
- `\uncover<n>{..}` shows at given n.
  (Ex) `\uncover<5>{I am 5}` ⇒
- `\invisible<n>{..}` hides at given n.
  (Ex) `\invisible<8>{Invisible at 8}` ⇒ Invisible at 8
- `\alt<n>{at n}{not at n}` for two alternatives.
  (Ex) `\alt<11>{I am 11}{I am not 11}` ⇒ I am not 11
- `\temporal<n>{before}{at n}{after}` for three alternatives.[9]
  (Ex) `\temporal<14>{I am 13}{I am 14}{I am 15}` ⇒ I am 15

[9] See also highlighting section.

More Replaces

In case of subtle differences in the heights of replacements, `overlayarea` and `overprint` environments can be used.

More Replaces

In case of subtle differences in the heights of replacements, `overlayarea` and `overprint` environments can be used.

- `\only<n>` in overlayarea environment:
  The development of pMSGA is based on NSGA-II and PGAPack.

More Replaces

In case of subtle differences in the heights of replacements, `overlayarea` and `overprint` environments can be used.

- `\only<n>` in overlayarea environment:
  The main difference is sharing again and new density function.

More Replaces

In case of subtle differences in the heights of replacements, `overlayarea` and `overprint` environments can be used.

- `\only<n>` in overlayarea environment:
  The main difference is sharing again and new density function.

- `\onslide<n>` in overprint environment:
  This is a first line.
  This is a second, long line.

More Replaces

In case of subtle differences in the heights of replacements, `overlayarea` and `overprint` environments can be used.

- `\only<n>` in overlayarea environment:
  The main difference is sharing again and new density function.

- `\onslide<n>` in overprint environment:
  The previous two lines are replaced by this one.

Simple Highlighting

`\item <+-| alert@+>` for automatic highlighting.

```
\begin{itemize}
\item <+-| alert@+> Every thing
\item <+-| alert@+> that has
\item <+-| alert@+> beginning
\item <+-| alert@+> has end.
\end{itemize}
```

Simple Highlighting

`\item <+-| alert@+>` for automatic highlighting.

```
\begin{itemize}
\item <+-| alert@+> Every thing
\item <+-| alert@+> that has
\item <+-| alert@+> beginning
\item <+-| alert@+> has end.
\end{itemize}
```

- Everything

Simple Highlighting

`\item <+-| alert@+>` for automatic highlighting.

```
\begin{itemize}
\item <+-| alert@+> Every thing
\item <+-| alert@+> that has
\item <+-| alert@+> beginning
\item <+-| alert@+> has end.
\end{itemize}
```

- Everything
- that has

Simple Highlighting

`\item <+-| alert@+>` for automatic highlighting.

```
\begin{itemize}
\item <+-| alert@+> Every thing
\item <+-| alert@+> that has
\item <+-| alert@+> beginning
\item <+-| alert@+> has end.
\end{itemize}
```

- Everything
- that has
- beginning

Simple Highlighting

`\item <+-| alert@+>` for automatic highlighting.

```
\begin{itemize}
\item <+-| alert@+> Every thing
\item <+-| alert@+> that has
\item <+-| alert@+> beginning
\item <+-| alert@+> has end.
\end{itemize}
```

- Everything
- that has
- beginning
- has end.

Simple Highlighting

`\item <+-| alert@+>` for automatic highlighting.

```
\begin{itemize}
\item <+-| alert@+> Every thing
\item <+-| alert@+> that has
\item <+-| alert@+> beginning
\item <+-| alert@+> has end.
\end{itemize}
```

- Everything
- that has
- beginning
- has end.

Simple Highlighting

`\item <+-| alert@+>` for automatic highlighting.

```
\begin{itemize}
\item <+-| alert@+> Every thing
\item <+-| alert@+> that has
\item <+-| alert@+> beginning
\item <+-| alert@+> has end.
\end{itemize}
```

- Everything
- that has
- beginning
- has end.

- You can also use `\begin{itemize}[<+-|alert@+>]` instead of individual '`\item <+-| alert@+>`'.

Simple Highlighting

`\item <+-| alert@+>` for automatic highlighting.

```
\begin{itemize}
\item <+-| alert@+> Every thing
\item <+-| alert@+> that has
\item <+-| alert@+> beginning
\item <+-| alert@+> has end.
\end{itemize}
```

- Everything
- that has
- beginning
- has end.

- You can also use `\begin{itemize}[<+-|alert@+>]` instead of individual '`\item <+-| alert@+>`'.
- You can also use `structure` instead of `alert`.

Alert for Highlighting

`\item<n->\alert<n>{stuff}` is better than the previous automatic one.

```
\begin{itemize}
\item<2->\alert<2> Every thing
\item<2->\alert<3> that has
\item<2->\alert<4> beginning
\item<2->\alert<5> has end.
\end{itemize}
```

Alert for Highlighting

`\item<n->\alert<n>{stuff}` is better than the previous automatic one.

```
\begin{itemize}
\item<2->\alert<2> Every thing
\item<2->\alert<3> that has
\item<2->\alert<4> beginning
\item<2->\alert<5> has end.
\end{itemize}
```

- Everything
- that has
- beginning
- has end.

Alert for Highlighting

`\item<n->\alert<n>{stuff}` is better than the previous automatic one.

```
\begin{itemize}
\item<2->\alert<2> Every thing
\item<2->\alert<3> that has
\item<2->\alert<4> beginning
\item<2->\alert<5> has end.
\end{itemize}
```

- Everything
- that has
- beginning
- has end.

Alert for Highlighting

`\item<n->\alert<n>{stuff}` is better than the previous automatic one.

```
\begin{itemize}
\item<2->\alert<2> Every thing
\item<2->\alert<3> that has
\item<2->\alert<4> beginning
\item<2->\alert<5> has end.
\end{itemize}
```

- Everything
- that has
- beginning
- has end.

Alert for Highlighting

`\item<n->\alert<n>{stuff}` is better than the previous automatic one.

```
\begin{itemize}
\item<2->\alert<2> Every thing
\item<2->\alert<3> that has
\item<2->\alert<4> beginning
\item<2->\alert<5> has end.
\end{itemize}
```

- Everything
- that has
- beginning
- has end.

Alert for Highlighting

`\item<n->\alert<n>{stuff}` is better than the previous automatic one.

```
\begin{itemize}
\item<2->\alert<2> Every thing
\item<2->\alert<3> that has
\item<2->\alert<4> beginning
\item<2->\alert<5> has end.
\end{itemize}
```

- Everything
- that has
- beginning
- has end.

Note that `\item<2->\alert<2>` is same to `\item<2-|alert@2>`.

Alternative for Highlighting

- `\alt<n>{\color{col1}..}{\color{col2}..}` for active/inactive highlighting

Alternative for Highlighting

- `\alt<n>{\color{col1}..}{\color{col2}..}` for active/inactive highlighting
- Example:
  - Everything
  - that has
  - beginning
  - has end.

Alternative for Highlighting

- `\alt<n>{\color{col1}..}{\color{col2}..}` for active/inactive highlighting
- Example:
  - Everything
  - that has
  - beginning
  - has end.

Alternative for Highlighting

- `\alt<n>{\color{col1}..}{\color{col2}..}` for active/inactive highlighting
- Example:
  - Everything
  - that has
  - beginning
  - has end.

Alternative for Highlighting

- `\alt<n>{\color{col1}..}{\color{col2}..}` for active/inactive highlighting
- Example:
  - Everything
  - that has
  - beginning
  - has end.

Alternative for Highlighting

- `\alt<n>{\color{col1}..}{\color{col2}..}` for active/inactive highlighting
- Example:
  - Everything
  - that has
  - beginning
  - has end.
- Source code:

```
\begin{itemize}
  \item<2-> \alt<2>{\color{blue} Everything}{\color{gray} Everything}
  \item<2-> \alt<3>{\color{blue} that has}{\color{gray} that has}
  \item<2-> \alt<4>{\color{blue} beginning}{\color{gray} beginning}
  \item<2-> \alt<5>{\color{blue} has end.}{\color{gray} has end.}
\end{itemize}
```

Temporal for Highlighting

- `\temporal<n>{before}{on}{after}` for incremental highlighting

Temporal for Highlighting

- `\temporal<n>{before}{on}{after}` for incremental highlighting
- Ready?
  - Everything
  - that has
  - beginning
  - has end.

Temporal for Highlighting

- `\temporal<n>{before}{on}{after}` for incremental highlighting
- Ready?
  - Everything
  - that has
  - beginning
  - has end.

Temporal for Highlighting

- `\temporal<n>{before}{on}{after}` for incremental highlighting
- Ready?
  - Everything
  - that has
  - beginning
  - has end.

Temporal for Highlighting

- `\temporal<n>{before}{on}{after}` for incremental highlighting
- Ready?
  - Everything
  - that has
  - beginning
  - has end.

Temporal for Highlighting

- `\temporal<n>{before}{on}{after}` for incremental highlighting
- Ready?
  - Everything
  - that has
  - beginning
  - has end.

Temporal for Highlighting

- `\temporal<n>{before}{on}{after}` for incremental highlighting
- Ready?
  - Everything
  - that has
  - beginning
  - has end.
- Source code:

```
\def\hilite<#1>{%
  \temporal<#1>{\color{gray}}{\color{blue}}%
               {\color{blue!25}}}
...
\begin{itemize}
  \hilite<3> \item Everything
  \hilite<4> \item that has
  \hilite<5> \item beginning
  \hilite<6> \item has end.
\end{itemize}
```

Other Highlightings

- `\textbf`, `\textit`, `\textsl`, `\textrm`, `\textsf`, and `\color` also understand overlays.

Other Highlightings

- `\textbf`, `\textit`, `\textsl`, `\textrm`, `\textsf`, and `\color` also understand overlays.
- Example

Other Highlightings

- `\textbf`, `\textit`, `\textsl`, `\textrm`, `\textsf`, and `\color` also understand overlays.
- Example
  - Everything (`\color<3-4>{olive}{Everything}`)

Other Highlightings

- `\textbf`, `\textit`, `\textsl`, `\textrm`, `\textsf`, and `\color` also understand overlays.
- Example
  - Everything (`\color<3-4>{olive}{Everything}`)
  - that has

Other Highlightings

- `\textbf`, `\textit`, `\textsl`, `\textrm`, `\textsf`, and `\color` also understand overlays.
- Example
  - Everything (`\color<3-4>{olive}{Everything}`)
  - that has
  - beginning (`\color<5>[rgb]{.9,.5,.5}beginning`)

Other Highlightings

- `\textbf`, `\textit`, `\textsl`, `\textrm`, `\textsf`, and `\color` also understand overlays.
- Example
  - Everything (`\color<3-4>{olive}{Everything}`)
  - that has
  - beginning (`\color<5>[rgb]{.9,.5,.5}beginning`)
  - has end.

Animation

- For dynamic presentation Beamer supports *transition*, *overlay*, and *animation*.
- Animation depends on your imagination and LaTeX skill.

Animation

- For dynamic presentation Beamer supports *transition*, *overlay*, and *animation*.
- Animation depends on your imagination and LaTeX skill.
- Supported animation types
  - Animate + Overlay
  - Animatevalue
  - Timed overlays (auto advancing)

Animation

- For dynamic presentation Beamer supports *transition*, *overlay*, and *animation*.
- Animation depends on your imagination and LaTeX skill.
- Supported animation types
  - Animate + Overlay
  - Animatevalue
  - Timed overlays (auto advancing)
- Use with caution as animation needs *lots* of slides

Animate + Overlay

- `\animate<n>`[10] for automatic stepwise viewing

[10] Remember that n can be n1-n2, n1-, or etc.

Animate + Overlay

- `\animate<n>`[10] for automatic stepwise viewing
- Ready?

[10] Remember that n can be n1-n2, n1-, or etc.

Animate + Overlay

- `\animate<n>`[10] for automatic stepwise viewing
- Ready?
  - Everything

[10] Remember that n can be n1-n2, n1-, or etc.

Animate + Overlay

- `\animate<n>`[10] for automatic stepwise viewing
- Ready?
  - Everything
  - that has

[10] Remember that n can be n1-n2, n1-, or etc.

Animate + Overlay

- `\animate<n>`[10] for automatic stepwise viewing
- Ready?
  - Everything
  - that has
  - beginning

[10] Remember that n can be n1-n2, n1-, or etc.

Animate + Overlay

- `\animate<n>`[10] for automatic stepwise viewing
- Ready?
  - Everything
  - that has
  - beginning
  - has end.

[10] Remember that n can be n1-n2, n1-, or etc.

Animate + Overlay

- `\animate<n>`[10] for automatic stepwise viewing
- Ready?
  - Everything
  - that has
  - beginning
  - has end.

[10] Remember that n can be n1-n2, n1-, or etc.

Animate + Overlay

- `\animate<n>`[10] for automatic stepwise viewing
- Ready?
  - Everything
  - that has
  - beginning
  - has end.
- Source code:

```
\frame{\animate<3-6>\frametitle{Animate + Overlay}%
...
\begin{itemize}[<+->]
  \item Everything
  \item that has
  \item beginning
  \item has end.
\end{itemize}
```

[10] Remember that n can be n1-n2, n1-, or etc.

Animatevalue

- `\animate<n>` to animate 'n' slides
- `\animatevalue<n>{name}{start}{end}` for specifying animation effects
  - `name`: counter or dimension
  - `start` and end values of the `value`

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

  Flying in from right!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

  Flying in from right!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

  Flying in from right!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

left!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

m left!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

from left!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

in from left!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

ing in from left!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying in from left!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying in from left!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying in from left!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying in from left!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying in from left!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying in from left!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

Flying in from right!

Flying in from left!

Flying Animation

- `\animate` and `\animatevalue` are used.
- This animation consumes 31 pages!
- Ready to explore?

  Flying in from right!
  Flying in from left!

Flying Animation - Source

```
\newcount\opaqueness
\newdimen\offset
\frame{\frametitle{Flying Animation}%
\animate<2-15,17-30>           % Actual animation values. Try <1-31>
\begin{itemize}
 \item[]
 \animatevalue<1-15>{\opaqueness}{0}{100}%
 \animatevalue<1-15>{\offset}{6cm}{0cm}%
 \begin{colormixin}{\the\opaqueness!averagebackgroundcolor}
  \hspace{\offset} Flying in from {\color{olive} right}!
 \end{colormixin}

 \item[]
 \animatevalue<17-31>{\opaqueness}{0}{100}  % Starts at 17, not 16, to give
 \animatevalue<17-31>{\offset}{-5cm}{0cm}   % one pause!
 \begin{colormixin}{\the\opaqueness!averagebackgroundcolor}
  \hspace{\offset} Flying in from {\color{olive} left}!
 \end{colormixin}
\end{itemize}
```

Timed Overlays

- Adobe Reader supports *timed overlays*, often called *auto advancing*.
- Two approaches
  - `\hypersetup{pdfpageduration=time}` from *hyperref* package + overlay macros
  - `\transduration<n>{time}` from *beamer* package + overlay macros
- See beamer_pstricks.pdf to see a fancy example.
- Try to do the same thing using PGF. Easy or not?

Presentation Themes

- `\usetheme[option]{name}`: Named to `beamertheme<name>.sty`.
- Old themes: bars, boxes, classic, default, lined, plain, shadow, sidebar, sidebardark, sidebardarktab, sidebartab, split, tree, treebars
- New themes (v3.0)
  - W/o navigation bar: default, boxes, Bergen, Madrid, Pittsburgh, Rochester
  - With a tree-like navigation bar: Antibes, JuanLesPins, Montpellier.
  - With a TOC sidebar: Berkeley, PaloAlto, Goettingen, Marburg, Hannover
  - With a mini frame navigation: Berlin, Ilmenau, Dresden, Darmstadt, Frankfurt, Singapore, Szeged
  - With section and subsection titles: Copenhagen, Luebeck, Malmoe, Warsaw

Return to Theme

Color Themes

- `\usecolortheme[option]{name}`: Named to `beamercolortheme<name>.sty`.
- Four basic color themes:
  - Default and special-purpose themes: default, structure (e.g., `\usecolortheme[named=SeaGreen]{structure}`).
  - Complete color themes: albatross, beetle, crane, dove, fly, seagull
  - Inner color themes: lily, orchid
  - Outer color themes: whale, seahorse
- `\setbeamercolor{beamer_element}{color}` for color setup of Beamer elements

  (Ex) `\setbeamercolor{frametitle}{fg=blue,bg=yellow}`

Return to Theme

Font Themes

- `\usecolortheme[option]{name}`: Named to `beamerfonttheme<name>.sty`.
- New themes (v3.0): default, professionalfonts, serif, structurebold, structureitalicserif, structuresmallcapsserif

Return to Theme

Hyperlinks and Buttons

- Beamer provides additional options for hyperlinks and buttons.
- `\hyperlink{targetname}{\beamergotobutton{text}}` to create link.
- `\hypertarget{targetname}{text}` to create target.
- Some useful buttons are `\beamerbutton`, `\beamergotobutton`, and `\beamerreturnbutton`.
- To go to the last slide, click here.

Notes

- To add notes to PDF screen, `\documentclass[notes]{beamer}`.
- To make only notes, `\documentclass[notesonly]{beamer}`.

Notes

- To add notes to PDF screen, `\documentclass[notes]{beamer}`.
- To make only notes, `\documentclass[notesonly]{beamer}`.
- Notes addition by adding `\note[options]{...}` after `\frame{...}`.

Notes

- To add notes to PDF screen, `\documentclass[notes]{beamer}`.
- To make only notes, `\documentclass[notesonly]{beamer}`.
- Notes addition by adding `\note[options]{...}` after `\frame{...}`.
- This slide has notes. Want to see them?

Notes

- To add notes to PDF screen, `\documentclass[notes]{beamer}`.
- To make only notes, `\documentclass[notesonly]{beamer}`.
- Notes addition by adding `\note[options]{...}` after `\frame{...}`.
- This slide has notes. Want to see them?

- Can you see me?
- Two note options for `note` are itemize and enumerate.
- `\beamertemplatenoteplain` for plain note page!

Merge for “trans” Output

- Beamer screen size = 128mm x 96mm
- Merge transparency output on letter paper for printing!

```
\documentclass{article} %
\usepackage{pdfpages}
\begin{document}
\includepdf[nup=2x2,landscape,delta=5mm 5mm,%
            scale=0.95,pages={1-18}]{trans.pdf}
\end{document}
```

- To return, click here.

Movie

- Beamer provides `multimedia` package.[11]
- `\movie[options]{poster}{file_name}`
  - `poster`: Poster for the movie (empty, text, or image).
  - `file_name`: AVI or MPG.
  - Works with pdflatex and dvips/ps2pdf routes.
- Some useful options
  - `autostart`, `loop`, `repeat`, `palindrome`
  - `borderwidth`, `showcontrols`, `externalviewer`

- Example: `clock.avi`

[11] New in Version 2.2. Can be used independently.

Sound

- Beamer provides `multimedia` package.
- `\sound[options]{poster}{file_name}`
  - Cannot be used with `dvips/ps2pdf` route.
  - File types depend on Acrobat Reader versions
- Some useful options
  - `autostart`, `automute`, `loop`, `repeat`.
  - `inlinesound` to embed sound files to PDF.
  - `channels` (1), `samplingrate` (44100), `bitspersample` (16), `encoding` (μlaw) are important!
- Example: `\sound[autostart,samplingrate=705000,bitspersample=16,channels=2]{Example}{notify.wav}`

Footer Design

- To add logo, `\logo{stuff}` in the preamble.
  - The logo will place in the right bottom corner.
  - How to change it? – See below!
- To redesign the footer, apply the following code:

```
\usefoottemplate{\vbox{%
\tinycouredline{structure!25}%
  {\color{white}\textbf{\insertshortauthor\hfill%
   \insertshortinstitute}}%
\tinycouredline{structure}%
  {\color{white}\textbf{\insertshorttitle}\hfill}%
}}
```

Emulations of Other Packages

- You can use FoilTEX, (HA)Prosper, Seminar, or TEXPower slides within Beamer.
- Not perfect, but you can *easily* import your slides written from the above four classes.
- Prosper example:

```
\usepackage{beamerprosper}                % Required
...
\overlays{8}{%
\begin{slide}{Prosper Emulation Example}
 \begin{itemize}
  \item Backward writing is easy and simple:
   \fromSlide{8}{{\color{green} P}}%
   \fromSlide{7}{{\color{blue} R}}%
   \fromSlide{6}{{\color{magenta} O}}%
   \fromSlide{5}{{\color{cyan} S}}%
   \fromSlide{4}{{\color{yellow} P}}%
   \fromSlide{3}{{\color{olive} E}}%
   \fromSlide{2}{{{\color{red} R}}}
 \end{itemize}
\end{slide} }%
```

Prosper Result

- This slide is written with Prosper syntax!
- Backward writing is easy and simple:

Prosper Result

- This slide is written with Prosper syntax!
- Backward writing is easy and simple:

R

Prosper Result

- This slide is written with Prosper syntax!
- Backward writing is easy and simple:

  ER

Prosper Result

- This slide is written with Prosper syntax!
- Backward writing is easy and simple:

  PER

Prosper Result

- This slide is written with Prosper syntax!
- Backward writing is easy and simple:

  SPER

Prosper Result

- This slide is written with Prosper syntax!
- Backward writing is easy and simple:
  OSPER

Prosper Result

- This slide is written with Prosper syntax!
- Backward writing is easy and simple:
  ROSPER

Prosper Result

- This slide is written with Prosper syntax!
- Backward writing is easy and simple:
  PROSPER

Hangul

- If you installed HLaTeX, load `\usepackage{hfont}`.
  - `\textgs{...}` ⇒ 아름다운 한글 그리고 金 杞朱
  - Click here to return.
- Note: Hangul bookmarks and Hangul search in PDF are only supported by `dvipdfm(x)`. But Beamer does not support `dvipdfm(x)`.
- Beamer option `[cjk]` is supported.
- `\usepackage[utf8]{inputenc}` is supported.

Other Macros

- To remove navigation symbols, `\usenavigationsymbolstemplate{}`.

Last Slide

- This page is directed from the button you clicked.
- To go back, click here.

Reference


Ki-Joo Kim, *Ki-Joo's LaTeX Documents* (http://www.geocities.com/kijoo2000/).


Michael Wiedmann, *Screen Presentation Tools* (http://www.miwie.org/presentations/presentations.html).