CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

F3 Faculty of Electrical Engineering
Department of Mathematics

CTUslides — simple slides in CTUstyle design

Petr Olšák
petr@olsak.net

http://petr.olsak.net/ctustyle.html

Basics

- **The document is included in a file (say `file.tex`) and it can be processed by `pdfcsplain file` command.**
- **The header of the document should be:**

```
\input ctuslides2 % slides macro (in version 2)
\worktype[B/EN]   % type of the work (B,M,D,O) and language (CZ,SK,EN)
\faculty{F3}      % the faculty in short
\department {Department of Mathematics} % depart
ment

\slideshow        % begin of the document
... document ...
\pg.
```

- **The document must be finished by `\pg` followed by period.**
- **You need OPmac in the version May 2015 or newer. Available at `http://petr.olsak.net/opmac-e.html`.**
- **The work type should be set similarly as in CTUstyle.**
- **Only `\worktype`, `\faculty` and `\department` work here. No more declaration sequences from CTUstyle.**

The structural commands

- You can type * for starting of the item.
- Nested items lists (second and more level) are created in the `\begitems...\enditems` environments.
- The slide titles are created by `\sec Text` followed by empty line. You can use `\secc Text` similarly.
- The title page (first slide) can be special if `\tit Title` (followed by empty line) is used here.
- The `\subtit Author name etc.` (followed by empty line) can be used after `\tit` at the first slide.
- The paragraph texts are ragged right.
- You can use `\nl` for new line in the paragraph.
- You can use `\pg` followed by + or , or . for new slide.
- The page-bar in the right corner is clickable at it will be created correctly after second pass of the TEX run.

Next page (next slide)

- The control sequence \pg must be followed by:
  - the character + – next page keeps the same text and a next text is added (usable for partially uncovering of ideas),

Next page (next slide)

- The control sequence \pg must be followed by:
  - the character + – next page keeps the same text and a next text is added (usable for partially uncovering of ideas),
  - the character ; – normal next page,

Next page (next slide)

- The control sequence \pg must be followed by:
  - the character + – next page keeps the same text and a next text is added (usable for partially uncovering of ideas),
  - the character ; – normal next page,
  - the character . – the end of the document.

Next page (next slide)

- The control sequence `\pg` must be followed by:
  - the character `+` – next page keeps the same text and a next text is added (usable for partially uncovering of ideas),
  - the character `;` – normal next page,
  - the character `.` – the end of the document.
- Summary:

```
\pg+    ... uncover next text
\pg;    ... next page
\pg.    ... the end of the document
```

Next page (next slide)

- The control sequence \pg must be followed by:
  - the character + – next page keeps the same text and a next text is added (usable for partially uncovering of ideas),
  - the character ; – normal next page,
  - the character . – the end of the document.
- Summary:

```
\pg+    ... uncover next text
\pg;    ... next page
\pg.    ... the end of the document
```

- If the control sequence \slideshow is removed (or commented out) from the beginning of the document then \pg+ sequences are deactivated. This is usable for printing version of the document.
- Another variant is \pg= (i. e. \pg followed by =). It does not create new page, but it is used for verbatim environment (see next slide...).

Verbatim

Verbatim in paragraph

- In-line verbatim doesn't work with declared `\activettchar` when `\slideshow` is used.
- You can use the `\code` sequence described in OPmac trick 0102, see http://petr.olsak.net/opmac-tricks-e.html#code.
- The argument of the `\code` sequence is printed verbatim, but special TEX characters must be preceded by backslash. I. e. backslash is printed when it is doubled.

Verbatim

Verbatim in paragraph

- In-line verbatim doesn't work with declared `\activettchar` when `\slideshow` is used.
- You can use the `\code` sequence described in OPmac trick 0102, see http://petr.olsak.net/opmac-tricks-e.html#code.
- The argument of the `\code` sequence is printed verbatim, but special TEX characters must be preceded by backslash. I. e. backslash is printed when it is doubled.

Multi-line verbatim

- Multi-line verbatim is printed by `\begtt...\endtt` but `\pg=` must preceded:

```
\pg=\begtt
... verbatim text ...
\endtt
```

Example of multi-line verbatim

The source code includes:

```
\pg=\Red\typosize[13/15]\begtt
#include <stdio.h>
int main();
{
  printf("Hello world!\n");
}
\endtt
```

Example of multi-line verbatim

The source code includes:

```
\pg=\Red\typosize[13/15]\begtt
#include <stdio.h>
int main();
{
  printf("Hello world!\n");
}
\endtt
```

and the result is:

```
#include <stdio.h>
int main();
{
  printf("Hello world!\n");
}
```

Note that local declarations can be inserted between `\pg=` **and** `\begtt`.

Limits of the \pg+ sequence

- The \pg+ sequence cannot be used inside a group.
- The exception is the nested environment \begitems...\enditems.

Limits of the \pg+ sequence

- The \pg+ sequence cannot be used inside a group.
- The exception is the nested environment \begitems...\enditems.

What to do?

- If you need to set a different font size by \typosize or \typoscale then you this size globally and you can use \pg+ inside different size of the font. Finally, you have to return back to normal size by the \normalsize sequence.
- If you need to partially uncover the multi-line verbatim then you can use:

```
\pg=\begtt
... first line of the code ...
\endtt \pg+ \pg=\begtt
... second line of the code ...
\endtt \pg+
```

- If you need to uncover the texts more ingenious then you can use macros \use or \pshow (see next slide...)

Uncovering by \use and \pshow

- The macro \use{condition}\action runs \action only if the number of the slide layer passes the given condition.
- The macro \pshow X (means partially show) prints the following text to the end of the current group:
  - invisible, if the number of the slide layer is less than X,
  - red, if the number of slide the layer is equal to X,
  - normal (black), if the number of slide layers is greater than X.
- The number of the slide layer is reset to one after each \pg; and it is incremented by one after each \pg+.
- The \pshow macro is defined by the \use macro as follows:

```
\def\pshow#1{\use{=#1}\Red \use{<#1}\White \ignorespaces}
```

  You can redefine it as you wish.

An example of \pshow usage

```
\secc Ideas in special order

* {\pshow1 First idea}
* {\pshow3 Second idea}
* {\pshow2 Third idea}

\pg+\pg+\pg+

\secc A formula

Consider
$$
  E = {\pshow5 m}{\pshow6 c^2}
$$

\pg+\pg+\pg+

And that is all.

\pg;
```

Ideas in special order

- First idea
-
-

An example of \pshow usage

```
\secc Ideas in special order

* {\pshow1 First idea}
* {\pshow3 Second idea}
* {\pshow2 Third idea}

\pg+\pg+\pg+

\secc A formula

Consider
$$
  E = {\pshow5 m}{\pshow6 c^2}
$$

\pg+\pg+\pg+

And that is all.

\pg;
```

Ideas in special order

- First idea
-
- Third idea

An example of \pshow usage

```
\secc Ideas in special order

* {\pshow1 First idea}
* {\pshow3 Second idea}
* {\pshow2 Third idea}

\pg+\pg+\pg+

\secc A formula

Consider
$$
  E = {\pshow5 m}{\pshow6 c^2}
$$

\pg+\pg+\pg+

And that is all.

\pg;
```

Ideas in special order

- First idea
- Second idea
- Third idea

An example of \pshow usage

```
\secc Ideas in special order

* {\pshow1 First idea}
* {\pshow3 Second idea}
* {\pshow2 Third idea}

\pg+\pg+\pg+

\secc A formula

Consider
$$
  E = {\pshow5 m}{\pshow6 c^2}
$$

\pg+\pg+\pg+

And that is all.

\pg;
```

Ideas in special order

- First idea
- Second idea
- Third idea

A formula

Consider

$$E =$$

An example of \pshow usage

```
\secc Ideas in special order

* {\pshow1 First idea}
* {\pshow3 Second idea}
* {\pshow2 Third idea}

\pg+\pg+\pg+

\secc A formula

Consider
$$
  E = {\pshow5 m}{\pshow6 c^2}
$$

\pg+\pg+\pg+

And that is all.

\pg;
```

Ideas in special order

- First idea
- Second idea
- Third idea

A formula

Consider

$$E = m$$

An example of \pshow usage

```
\secc Ideas in special order

* {\pshow1 First idea}
* {\pshow3 Second idea}
* {\pshow2 Third idea}

\pg+\pg+\pg+

\secc A formula

Consider
$$
  E = {\pshow5 m}{\pshow6 c^2}
$$

\pg+\pg+\pg+

And that is all.

\pg;
```

Ideas in special order

- First idea
- Second idea
- Third idea

A formula

Consider

$$E = mc^2$$

An example of \pshow usage

```
\secc Ideas in special order

* {\pshow1 First idea}
* {\pshow3 Second idea}
* {\pshow2 Third idea}

\pg+\pg+\pg+

\secc A formula

Consider
$$
  E = {\pshow5 m}{\pshow6 c^2}
$$

\pg+\pg+\pg+

And that is all.

\pg;
```

Ideas in special order

- First idea
- Second idea
- Third idea

A formula

Consider

$$E = mc^2$$

And that is all.

Tables, pictures

- Tables can be created by `\table` or `\ctable` macro.
- Pictures can be included by `\inspic` macro.
- See OPmac documentation for more details.
- The centering would be done by the `\centerline{}` macro:
- Example:

Tables, pictures

- Tables can be created by `\table` or `\ctable` macro.
- Pictures can be included by `\inspic` macro.
- See OPmac documentation for more details.
- The centering would be done by the `\centerline{}` macro:
- Example:

```
\centerline{\picw=5cm \inspic cmelak1.jpg }
```

Comparison CTUslides with Beamer*

The LaTeX package Beamer gives much more features and many themes are prepared for Beamer, **but**

- the user of Beamer is forced to *program* his/her document using dozens of `\begin{foo}` and `\end{foo}` and many another programming constructions,
- on the other hand, plain TeX gives you a possibility to simply *write* your document with minimal markup. The result is more compact.

* http://www.ctan.org/pkg/beamer

Comparison CTUslides with Beamer*

The LaTeX package Beamer gives much more features and many themes are prepared for Beamer, **but**

- the user of Beamer is forced to *program* his/her document using dozens of `\begin{foo}` and `\end{foo}` and many another programming constructions,
- on the other hand, plain TeX gives you a possibility to simply *write* your document with minimal markup. The result is more compact.
- You need to read 250 pages of doc for understanding Beamer,
- on the other hand, you need to read only ten slides** and you are ready to use **CTUslides**.

* http://www.ctan.org/pkg/beamer

** this eleventh slide isn't counted

Comparison CTUslides with Beamer*

The LaTeX package Beamer gives much more features and many themes are prepared for Beamer, **but**

- the user of Beamer is forced to *program* his/her document using dozens of `\begin{foo}` and `\end{foo}` and many another programming constructions,
- on the other hand, plain TeX gives you a possibility to simply *write* your document with minimal markup. The result is more compact.
- You need to read 250 pages of doc for understanding Beamer,
- on the other hand, you need to read only ten slides** and you are ready to use **CTUslides**.
- A notice for programmers: to create another individual typographical design for LaTeX is much more complicated than to do the same in plain TeX. And you need to seriously understand plain TeX if you want to do something more complicated in LaTeX.

* http://www.ctan.org/pkg/beamer

** this eleventh slide isn't counted

Thanks for your attention

Thanks for your attention

Questions?