Further development

- Add more navigation buttons (and a more sensible naming scheme).
- More flexible labelling of standard buttons.
- Add means for **inline bookmarks** (parts of the table of contents displayed in a panel as a 'jump table').
- Sensible handling of **thumbnails**.
- Provide **progress indicators** which can also be used to jump to certain parts of the presentation.

8.1 Basic display effects

8.1.1 The \pause command

\pause ships out the current page, starts a new page and copies whatever was on the current page onto the new page, where typesetting is resumed.

This will create the effect of a **pause** in the presentation.

Things to pay attention to

1. **\pause** should appear in **vertical mode** only, i. e. between paragraphs or at places where ending the current paragraph doesn't hurt.
2. This means **\pause** is forbidden in all **boxed** material (including **tabular**), **headers/footers**, and **floats**.
3. **\pause** shouldn't appear either in environments which have to be *closed* to work properly, like **picture**, **tabbing**, and (unfortunately) environments for **aligned math formulas**.

4. **\pause** does work in all environments which mainly influence paragraph formatting, like **center**, **quote** or all **list** environments.

8.1.2 The \stepwise command

\stepwise{⟨contents⟩} is a command for displaying some part of a LaTeX document (which is contained in ⟨**contents**⟩) 'step by step'.

If ⟨**contents**⟩ contains one or more constructs of the form **\step{⟨stepcontents⟩}**, the following happens:

1. The current contents of the page are saved (as with **\pause**).

2. As many pages as there are **\step** commands in ⟨**contents**⟩ are produced.

   Every page starts with what was on the current page when **\stepwise** started.

   The first page also contains everything in ⟨**contents**⟩ which is *not* in ⟨**stepcontents**⟩ for any **\step** command.

   The second page additionally contains the ⟨**stepcontents**⟩ for the *first* **\step** command, and so on, until all ⟨**stepcontents**⟩ are displayed.

3. When all ⟨**stepcontents**⟩ are displayed, **\stepwise** ends and typesetting is resumed (still on the current page).

This will create the effect that the **\step** commands are executed 'step by step'.

Things to pay attention to

1. **\stepwise** should appear in **vertical mode** only, i. e. between paragraphs, just like **\pause**.
2. Don't put **\pause** or nested occurrences of **\stepwise** into ⟨**contents**⟩.
3. Structures where **\pause** does not work (like **tabular** or aligned equations) can go *completely* into ⟨**contents**⟩, where **\step** can be used freely.

4. `\step` can go in ⟨`stepcontents`⟩. The order of execution of `\step` commands is the order in which they appear in ⟨`contents`⟩, independent of nesting.


```
\begin{itemize}
\item one\pause
\item two\pause
\item three
\end{itemize}
```

gives

- one
- two
- three


```
\stepwise
{%
  one,
  \step{two, }%
  \step{three.}%
}
```

gives
one, two, three.

8.2 Customising display effects

8.2.1 `\boxedsteps` and `\nonboxedsteps`

By default, ⟨`stepcontents`⟩ belonging to a `\step` which is not yet 'active' are ignored altogether. This makes it possible to include e. g. tabulators `&` or line breaks into ⟨`stepcontents`⟩ without breaking anything.

Sometimes, the desired behaviour of a `\step` which is not yet 'active' is to create an appropriate amount of *blank space* where ⟨`stepcontents`⟩ can go as soon as it is activated.

The simplest and most robust way of doing this is to create an empty box (aka `\phantom`) with the same dimensions as the text to be hidden.

This behaviour is toggled by the following commands.

`\boxedsteps` makes `\step` create a blank box the size of ⟨`stepcontents`⟩ when inactive and put ⟨`stepcontents`⟩ into a box when active.

`\nonboxedsteps` activates the default behaviour.


```
\stepwise
{\begin{tabular}{ll}
    \hline 1 & one%
    \step{\\ 2 & two}%
    \step{\\ 3 & three}\\
    \hline
  \end{tabular}
  \par\boxedsteps
  \begin{tabular}{ll}
    \hline 1 & one\\
    \step{2}&\step{two}\\
    \step{3}&\step{three}\\
    \hline
  \end{tabular}}
```

gives

1	one
2	two
3	three

1	one
2	two
3	three

8.2.2 Custom versions of `\stepwise`

Sometimes, it might happen that vertical spacing is different on every page of a sequence generated by `\stepwise`, making lines 'wobble'.

This is caused by interactions between different ways vertical spacing is added to the page. Hopefully, problems caused this way can be reduced until the first *beta* release.

There are two custom versions of `\stepwise` which should produce better vertical spacing.

`\liststepwise{`⟨`contents`⟩`}` works exactly like `\stepwise`, but adds an 'invisible rule' before ⟨`contents`⟩. Use for list environments and aligned equations.

`\parstepwise{`⟨`contents`⟩`}` works like `\liststepwise`, but `\boxedsteps` is turned on by default. Use for texts where `\step`s are to be filled into blank spaces.

8.2.3 Starred versions of \stepwise commands

Usually, the first page of a sequence produced contains *only* material which is *not* part of any ⟨**stepcontents**⟩. The first ⟨**stepcontents**⟩ are displayed on the second page of the sequence.

For special effects, it might be desirable to have the first ⟨**stepcontents**⟩ active even on the first page of the sequence.

All variants of **\stepwise** have a starred version (e. g. **\stepwise***) which does exactly that.

8.2.4 The optional argument of \stepwise

Every variant of **\stepwise** takes an optional argument, like this: **\stepwise[⟨settings⟩]{⟨contents⟩}**.

⟨**settings**⟩ will be placed right before the internal loop which produces the sequence of pages. It can contain settings of parameters which modify the behaviour of **\stepwise** or **\step**. ⟨**settings**⟩ is placed inside a group so all changes are local to this call of **\stepwise**.

Some internal macros and counters which can be adjusted are explained in the following.

8.2.5 Customizing the way ⟨stepcontents⟩ is diplayed

Internally, there are three macros (taking one argument each) which control how ⟨**stepcontents**⟩ is displayed: **\displaystepcontents**, **\hidestepcontents**, and **\activatestep**. Virtually, every **\step{⟨stepcontents⟩}** is replaced by

\hidestepcontents{⟨stepcontents⟩}
when this step is not yet active.

\displaystepcontents{\activatestep{⟨stepcontents

when this step is activated *for the first time*.

\displaystepcontents{⟨stepcontents⟩}
when this step has been activated before.

By redefining these macros, the behaviour of **\step** is changed accordingly. You can redefine them inside ⟨**contents**⟩ to provide a change affecting one **\step** only, or in the optional argument of **\stepwise** to provide a change for all **\step**s inside ⟨**contents**⟩.

\activatestep is set to **\displayidentical** by default, the default settings of **\hidestepcontents** and

\displaystepcontents depend on whether **\boxedsteps** or **\nonboxedsteps** (default) is used.

texpower offers nine standard definitions.

For interpreting **\displaystepcontents**:

\displayidentical Simply expands to its argument. The same as LaTeXs **\@ident**. Used by **\nonboxedsteps** (default).

\displayboxed Expands to an **\mbox** containing its argument. Used by **\boxedsteps**.

For interpreting **\hidestepcontents**:

\hideignore Expands to nothing. Used by **\nonboxedsteps** (default).

\hidephantom Expands to a **\phantom** containing its argument. Used by **\boxedsteps**.

\hidevanish In a colored document, makes its argument 'vanish' by setting all colors to **\vanishcolor** (defaults to **pagecolor**). This will give weird results with structured backgrounds.

s **\hidetext** Produces blank space of the same dimensions as the space that would be occupied if its argument would be typeset in the current paragraph. Respects automatic hyphenation and line breaks.

This command needs the **soul** package to work, which is not loaded by **texpower** itself. Consult the documentation of **soul** concerning restrictions on commands implemented using **soul**.

`\hidedimmed` In a colored document, displays its argument with dimmed colors. Note that this doesn't make the argument completely invisible.

For monochrome documents, there is no useful interpretation for this command, so it is disabled.

For interpreting `\activatestep`:

`\highlightboxed` If the `colorhighlight` option is set, expands to a box with colored background containing its argument. Otherwise, expands to an `\fbox` containing its argument. The resulting box has the same dimensions as the argument (background may overlap surrounding text).

There is a new length register `\highlightboxsep` which acts like `\fboxsep` for the resulting box and defaults to `0.5\fboxsep`.

s `\highlighttext` If the `colorhighlight` option is set, puts its argument on colored background. Otherwise, underlines its argument. The resulting text has the same dimensions as the argument (background may overlap surrounding text).

`\highlightboxsep` is used to determine the extent of the coloured box(es) used as background.

This command needs the `soul` package to work (compare the description of `\hidetext`).

`\highlightenhanced` In a colored document, displays its argument with enhanced colors.

For monochrome documents, there is no useful interpretation for this command, so it is disabled.

8.2.6 Variants of `\step`

There are a couple of custom versions of `\step` which make it easier to achieve special effects needed frequently.

`\bstep` Like `\step`, but is *always* boxed.

`\bstep{⟨stepcontents⟩}` is implemented in principle as `{\boxedsteps\step{⟨stepcontents⟩}}`.

`\switch{⟨ifinactive⟩}{⟨ifactive⟩}` is a variant of `\step` which, instead of making its argument appear, switches between ⟨`ifinactive`⟩ and ⟨`ifactive`⟩ when activated.

In fact, `\step{⟨stepcontents⟩}` is in principle implemented by

```
\switch{\hidestepcontents{⟨stepcontents⟩}}
       {\displaystepcontents{⟨stepcontents⟩}}
```

Beware of problems when ⟨`ifinactive`⟩ and ⟨`ifactive`⟩ have different dimensions.

`\dstep` A variant of `\step` which takes **no** argument, but simply switches colors to 'dimmed' if not active.

`\vstep` A variant of `\step` which takes **no** argument, but simply switches all colors to `\vanishcolor` (defaults to `pagecolor`) if not active.

`\steponce` Like `\step`, but goes inactive again in the subsequent step.

`\multistep` is a shorthand macro for executing several steps successively. The syntax is

`\multistep*[⟨activatefirst⟩]{⟨n⟩}{⟨stepcontents⟩}`

where ⟨`n`⟩ is the number of steps.

Only one instance of ⟨`stepcontents`⟩ is displayed at a time. Inside ⟨`stepcontents`⟩, a counter `substep` can be evaluated which tells the number of the current instance.

In the starred form the last instance of ⟨`stepcontents`⟩ stays visible.

`\movie` works like `\multistep`, but between `\steps`, pages are advanced automatically every ⟨`dur`⟩ seconds. The syntax is

`\movie{⟨n⟩}{⟨dur⟩}[⟨stop⟩]{⟨stepcontents⟩}`

where ⟨`n`⟩ is the number of steps. The additional optional argument ⟨`stop`⟩ gives the code (default: `\stopAdvancing`) which stops the animation. (`\movie` accepts the same first optional argument as `\multistep` but it was left out above.)

`\overlays` is another shorthand macro for executing

several steps successively. In contrast to `\multistep`, it doesn't print things *after* each other, but *over* each other. The syntax is

`\overlays[⟨activatefirst⟩]{⟨n⟩}{⟨stepcontents⟩}`

where ⟨`n`⟩ is the number of steps. Inside ⟨`stepcontents`⟩, a counter `substep` can be evaluated which tells the number of the current instance.

`\restep`, `\rebstep`, `\reswitch`, `\redstep`, `\revstep`. Frequently, it is desirable for two or more steps to appear at the same time, for instance to fill in arguments at several places in a formula at once. `\restep{⟨stepcontents⟩}` is identical with `\step{⟨stepcontents⟩}`, but is activated at the same time as the previous occurrence of `\step`. `\rebstep`, `\reswitch`, `\redstep`, and `\revstep` do the same for `\bstep`, `\switch`, `\dstep`, and `\vstep`.

8.3 Controlling the order of display

8.3.1 Optional arguments of `\step`

The variants of `\step` take two optional arguments for influencing the mode of activation, like this:

`\step[⟨activatefirst⟩][⟨whenactive⟩]{⟨stepcontents⟩}`.

Both ⟨`activatefirst`⟩ and ⟨`whenactive`⟩ should be conditions in the syntax of the `\ifthenelse` command.

⟨`activatefirst`⟩ checks whether this `\step` is to be activated *for the first time*. The default value is `\value{step}=\value{stepcommand}`. By using `\value{step}=`⟨*n*⟩, this `\step` can be forced to appear as the *n*th one.

⟨`whenactive`⟩ checks whether this `\step` is to be considered active *at all*. The default behaviour is to check whether this `\step` has been activated before (this is saved internally for every step).

8.3.2 Finding out what's going on

Inside ⟨`settings`⟩ and ⟨`contents`⟩, you can refer to the following internal state variables which provide information about the current state of the process executed by `\stepwise`:

counter: `firststep` The number from which to start counting steps (see counter `step` below). Is **0** by default and **1** for starred versions of `\stepwise`. You can set this in ⟨`settings`⟩ for special effects.

counter: `totalsteps` The total number of `\step` commands occurring in ⟨`contents`⟩.

counter: `step` The number of the current iteration, i. e. the number of the current page in the sequence of pages produced by `\stepwise`. Runs from `\value{firststep}` to `\value{totalsteps}`.

counter: `stepcommand` The number of the `\step` command currently being executed.

boolean: `firstactivation` `true` if this `\step` is active for the first time, `false` otherwise.

boolean: `active` `true` if this `\step` is currently active, `false` otherwise.

`stepcommand`, `firstactivation`, and `active` are useful only inside ⟨`stepcontents`⟩.

8.3.3 \afterstep

It might be neccessary to set some parameters which affect the appearance of the *page* (like page transitions) inside ⟨stepcontents⟩. However, the \step commands are usually placed deeply inside some structure, so that all *local* settings are likely to be undone by groups closing before the page is completed.

\afterstep{⟨settings⟩} puts ⟨settings⟩ right before the end of the page, after the current step is performed.

Things to pay attention to

1. There can be only one effective value for ⟨settings⟩. Every occurrence of \afterstep overwrites this value globally.
2. \afterstep will *not* be executed in ⟨stepcontents⟩ if the corresponding \step is not active, even if ⟨stepcontents⟩ is displayed owing to a redefinition of \hidestepcontents.

8.4 Page transitions and automatic advancing

8.4.1 Page transitions

These commands work only if the hyperref package is loaded.

The following page transition commands are defined:

\pageTransitionSplitHO Split Horizontally to the outside.

\pageTransitionSplitHI Split Horizontally to the inside.

\pageTransitionSplitVO Split Vertically to the outside.

\pageTransitionSplitVI Split Vertically to the inside.

\pageTransitionBlindsH Horizontal Blinds.

\pageTransitionBlindsV Vertical Blinds.

\pageTransitionBoxO Growing Box.

\pageTransitionBoxI Shrinking Box.

\pageTransitionWipe{⟨angle⟩}
Wipe from one edge of the page to the facing edge.
⟨angle⟩ is a number between **0** and **360** which specifies the direction (in degrees) in which to wipe.
Apparently, only the values **0**, **90**, **180**, **270** are supported.

\pageTransitionDissolve Dissolve.

\pageTransitionGlitter{⟨angle⟩}
Glitter from one edge of the page to the facing edge.
⟨angle⟩ is a number between **0** and **360** giving the direction (in degrees) in which to glitter.
Apparently, only the values **0**, **270**, **315** are supported.

\pageTransitionReplace Simple Replace (the default).

Things to pay attention to

1. The setting of the page transition is a property of the *page*, i. e. whatever page transition is in effect when a page break occurs, will be assigned to the corresponding pdf page.
2. The page transition setting is local to groups.
   Make sure no LATEX environment is ended between a \pageTransition setting and the next page break. In particular, in ⟨stepcontents⟩, \afterstep should be used.

3. Setting page transitions works well with \pause. Here, \pause acts as a page break, i. e. a different page transition can be set before every occurrence of \pause.

8.4.2 Automatic advancing of pages

If you have loaded the `hyperref` package, then the following command is defined which enables automatic advancing of `pdf` pages.

h `\pageDuration{⟨dur⟩}` causes pages to be advanced automatically every ⟨`dur`⟩ seconds. ⟨`dur`⟩ should be a non-negative fixed-point number.

Depending on the `pdf` viewer, this will happen only in full-screen mode.

The same restrictions as for **page transitions** apply. In particular, the page duration setting is undone by the end of a group, i. e. it is useless to set the page duration if a LATEX environment ends before the next page break.

There is no 'neutral' value for ⟨`dur`⟩ (**0** means advance as fast as possible). You can make automatic advancing stop by calling `\pageDuration{}`. `texpower` offers the custom command

h `\stopAdvancing`

to do this.