# PPower4 Manual

preliminary version 0.4

Klaus Guntermann
Christian Spannagel
TU Darmstadt
Computer Science Department
Systems Programming Group
`guntermann@iti.informatik.tu-darmstadt.de`
`chrisp@iti.informatik.tu-darmstadt.de`

preliminary September 12, 2002

## Contents

Updated sections with respect to previous version:
Several enhancements and extensions. New: use with vtex and dvipdfm

## 1 Introduction

PPower4 can be used to postprocess documents for presentations created with `pdflatex`, `vlatex` or `dvipdfm` where parts of a page are to be uncovered step by step during a presentation with Acrobat Reader. Additional features include displaying parts of a page in any order, removing items from the page, and inserting fancy transitions or backgrounds.
PPower4 ist not a complete presentation package, though. It is meant as an enhancement for your favourite presentation package (be it seminar, pdfslide or foiltex). PPower4 should work with each of them. The main responsibility of PPower4 is to add some dynamic features to your presentation. Because PPower4 needs to insert special tags into the resulting `PDF` document it cannot be used with normal LaTeX followed by dvips and distiller (or GhostScript) to create a `PDF` presentation. But it is no longer tied to `pdflatex` only.
PPower4 consists of some style files to be included during processing with the applications mentioned above and a post processor written in Java.
The current setup provides and describes mostly basic features, which can be used to develop more comfortable packages and environments. Any contributions in this area are welcome.

## 2 Requirements

To make use of all features described in this document it is necessary to use Java2 for running the post processor. Currently `kaffe` (checked with version 1.0.6) cannot be used.
You can use PPower4 together with `pdflatex`, `vlatex` and `dvipdfm`. If you are using `dvipdfm`, you must also have `latex` to prepare the dvi file.

## 3 Installation and Usage

– to be extended –
You need to put the style files in a place, where `pdflatex`, `vlatex` or `latex` can find them. According to the TDS conventions this may be a subdirectory named `tex/latex/ppower4/` or `tex/latex/misc/` in your (site specific) installation tree (insert your appropriate directory delimiter instead of `/`, if needed).
Furthermore you must be able to run Java programs. See the PPower4 home page for links to Java environments. PPower4 is normally called with a script or batch file, which is operating system dependent. You find such scripts for Unix and Windows on the PPower4 home page. But you have to adapt these scripts to your local file system structure. E.g. it is necessary to edit the script to name the path of the `jar` file, which contains the PPower4 system and libraries.

### 3.1 Additional entries in your LaTeX source

If you are using `pdflatex` or `vlatex`, just can simply include the style files without any option via the `\usepackage` command:

```
\usepackage{pause}
\usepackage{background}
\usepackage{mpmulti}
```

Make sure that you have got `ifpdf.sty` and `ifvtex.sty` from CTAN (these are already included in a `vtex` distribution).
If you are using `dvipdfm`, you have to add the option `dvipdfm` for the inclusion of `pause.sty` and `background.sty`:

```
\usepackage[dvipdfm]{pause}
\usepackage[dvipdfm]{background}
\usepackage{mpmulti}
```

If you want the special PPower4 commands to be ignored without changing all the places where the commands referring to the post processor were used in your source, you can add the option `ignore` to the commands including `pause.sty` and `background.sty`:

```
\usepackage[ignore]{pause}
\usepackage[ignore]{background}
```

### 3.2 Invoking PPower4

To run PPower4 you need to run the script or batch file. These require at least two arguments, namely the input file and the name of an output file.
Use different file names for input and output. Depending on your operating system it may be necessary to supply full path names also for the input and output file names.
The input file is the pdf file which you got running `pdftex`, `vtex` or `dvipdfm`. The output file will be created by the post processor. If your Acrobat Reader locks the pdf file during display, you should close it before reprocessing your document with PPower4.
There are some options, which increase the verbosity level of the post processor or switch off compression of the output file. But these are normally not needed and provided mostly for debugging purposes.

## 4 Simple application

To build a page incrementally, it is sufficient to insert `\pause` commands into your document. Take care to place these commands close to some text, because otherwise the command is likely to insert additional whitespace. In the `PDF` file created by `pdflatex`, `vlatex` or `dvipdfm` the spots where you placed the `\pause` command are indicated with a small colored rectangle. This rectangle will vanish in the final version. If you do not want these rectangles to be visible, include the style file `pause.sty` with the option `nomarkers`:

```
\usepackage[nomarkers]{pause}
```

After invoking PPower4 you will have a sequence of pages, which contain the partly built contents of the page and finally the full page contents.
The following example is similar to the text in one of the demo documents. It shows how to build a nested itemized list step by step just inserting `\pause` commands. You may wish to configure `itemize` to display fancier or colored elements.

```
\begin{itemize}
\item First one expects to build itemized lists\pause
  \begin{itemize}
  \item which can be nested\pause
    \begin{itemize}
    \item even this deep\pause
    \end{itemize}
  \item back again\pause
  \end{itemize}
\item and the end of this list
\end{itemize}
```

Please note, that the `\pause` to hold on after the inner `itemize` is located behind the last `\item`'s text, not behind the `\end{itemize}`.
Of course you can also insert a `\pause` into the text in the middle of a line (if that is reasonable).

```
What would you expect posing this question?\pause{} An answer.
```

`\pause` tries not to interfere with the rest of the text, but as with all commands some care must be taken. Make sure, that any whitespace surrounding `\pause` is wanted. Use `\pause{}` to keep whitespace after the command like in the last example.

## 5 Transition effects

Acrobat Reader allows several special effects, when going to a page. This transition is an attribute of the page, not of the step between the pages. You will always see the same transition effect when going *to* that page, be it forward, backward or through any hyperlink.
You will have to decide whether the transition effects will help the audience to follow the presentation or whether it will distract their attention from the contents.

### 5.1 Between original pages

For normal usage the effects can be triggered with functions provided by the hyperref package. The file `pagetrans.tex` (available from the PPower4 home page, kindly provided by Marc van Dongen) supports some abbreviations to facilitate using these effects.
The following list has the details, according to the Manual from Adobe:

`\Replace` This is the default effect: the new page image replaces the old page image instantaneously. Mostly used to reset the transition mode after a fancy transition.

`\Dissolve` The old page image "dissolves" in a piecemeal fashion to the new page image.

`\HBlinds` Multiple lines, evenly distributes accross the page, sweep horizontally to reveal the new page.

`\VBlinds` The same as `\HBlinds` but with vertical lines.

`\HOSplit` Two lines sweep across the screen revealing the new page image. The lines are horizontal and move from the center out.

`\HISplit` The same as `\HOSplit`, but lines move from the edges in.

`\VOSplit` The same as `\HOSplit`, but the lines are vertical.

`\VISplit` The same as `\VOSplit`, but the lines move from the edges in.

`\OBox` A box sweeps from the center out revealing the new page image.

`\IBox` A box sweeps from the edges inward revealing the new page image.

`\Wipe{value}` A single line sweeps across the screen from one edge to the other, revealing the new page image. The argument value is an angle, possible values include 0, 90, 180 and 270.

`\pageTransitionGlitter{value}` Similar `\Dissolve`, except the effect sweeps across the image in a wide band from one side of the screen to the other. The argument value is a direction, supported values are 0, 270 and 315.

### 5.2 Between partial builds of a page

If you want to use transition effects between the partially built pages, the arguments need to be specified in some extra place, because the `PDF` file format can contain only one transition effect per page. But in the initial pdf document created by `pdflatex`, `vlatex` or `dvipdfm` there is only one page for a sequence of pages in the final document.
To allow different transition effects between partial pages the `\pause` command has an optional argument. If there are no other selections made, the transition effect set for the complete page through hyperref supported selections will be inherited.
Specifying the effects literally each time would be error prone. And you would not have an easy way of finding out what went wrong, when an intended transition does not appear. Thus `pause.sty` defines specific commands which supply the user's selection to the argument of `\pause`. Only parts which must be specified explicitly (like some angles for directions etc.) must still be supplied in the call. The selected effect is kept for the current page and will be used for all future `\pause` commands until another selection is made.
All effects mentioned in the subsection above are also available here. The list is repeated for easier reference.

`\pauseReplace` This is the default effect: new page image replaces the old page image instantaneously. Mostly used to reset the transition mode after a fancy transition.

`\pauseDissolve` The old page image "dissolves" in a piecemeal fashion to the new page image.

`\pauseHBlinds` Multiple lines, evenly distributes accross the page, sweep horizontally to reveal the new page.

`\pauseVBlinds` The same as `\pauseHBlinds` but with vertical lines.

`\pauseHOSplit` Two lines sweep across the screen revealing the new page image. The lines are horizontal and move from the center out.

`\pauseHISplit` The same as `\pauseHOSplit`, but lines move from the edges in.

`\pauseVOSplit` The same as `\pauseHOSplit`, but the lines are vertical.

`\pauseVISplit` The same as `\pauseVOSplit`, but the lines move from the edges in.

`\pauseOBox` A box sweeps from the center out revealing the new page image.

`\pauseIBox` A box sweeps from the edges inward revealing the new page image.

`\pauseWipe{value}` A single line sweeps across the screen from one edge to the other, revealing the new page image. The argument value is an angle, possible values include 0, 90, 180 and 270.

`\pauseGlitter{value}` Similar `\pauseDissolve`, except the effect sweeps across the image in a wide band from one side of the screen to the other. The argument value is a direction, supported values are 0, 270 and 315.

## 6 Principles of operation

We have seen, that the `\pause` commands split the page into sections, which are shown step by step until the page is completed. We will refer to such a text section as a *chunk*. But this method does not yet allow to display the chunks in a different sequence.
Introducing a different build sequence for the chunks adds another dimension to the display process. Normally processing a document describes the text, which fills a two dimensional page. If we build

the page in the same sequence, no additional information is needed. Marking the breaks is sufficient. Because there are relations between the elements of a page (e. g. the width or height of the entries in a table) we cannot ignore these preconditions. We still base our page descriptions on the features provided by TeX. After all we want to take advantage of the excellent formatting capabilities of TeX. If we tag each chunk with a number, we can build the page in the sequence given by these tags. To keep compatibility with the initial setup we define that by default the first chunks gets number 1 and that each `\pause` command increments the current number by 1, which is assigned to the next chunk.
This means inserting $n$ `\pause` commands will assign level number 1 to the text before the first `\pause`, level number 2 to the text between the first and the second `\pause` etc. Finally the text behind the $n$th `\pause` will be assigned level number $n + 1$. When the output pages are created, all chunks which are tagged with a level number up to the current level will be included for display until pages for all levels are created.
Placement in any order can now be achieved by assigning a level number explicitly to a chunk. We explain this in more detail now.

## 6.1 Assigning levels

For the assignment of a level number to a chunk we introduce the command `\pauselevel`. The argument of `\pauselevel` can *set* the level to number $n$ with `\pauselevel{=`$n$`}`.
Absolute level numbers can make it difficult to insert additional building steps. To overcome this problem we allow also to *increase* the level number by $n$ using `\pauselevel{=+`$n$`}`. Similarly `\pauselevel{=-`$n$`}` will *decrease* the level number by $n$. If you use the relative setting, you must take into account, that the preceding `\pause` will have updated the current level already.
Of course several chunks can be assigned the same level number. They will appear together.
Because it is tedious to set the level explicitly, e. g. for writing from right to left, `\pauselevel` also supports to set the increment or decrement value for all subsequent `\pause` commands. At the beginning of each page the incrementing value is +1.
To set the level number to 11 and count down with subsequent `\pause` commands you would use the command `\pauselevel{=11 -1}`. Note that while the whitespace between `=11` and `-1` is required, there must be *no* whitespace between the `=` and the subsequent `+` and `-`.
It will not be possible to decrement the level number below 1. If you try, the corresponding chunk will show up already in the first page part.
To have material, which is added by the output routine of TeX for the bottom of a page, appear already on the first version of a page use the sequence `\pause\pauselevel{=1}` at the end of your text for the page.
Currently see `leveldemo.tex` for some examples.
The maximum level number can be quite large, but it is recommended to make the range not too wide, because otherwise building the pages may take some time.

## 6.2 Removing items

Setting the level number $n$ for a chunk will include the chunk in all pages displayed with highest level number larger or equal to $n$. To make a chunk vanish we need to specify a maximum level number $m$. Then the chunk will only be included in all pages starting from level 1 or any specified level $n$ until maximum level number $m$.
The maximum level number is specified after a colon in the `\pauselevel` command. You can specify the level *absolutely* as number $m$ with `\pauselevel{:`$m$`}`. If you want to specify it *relative* to the current level number with an distance of $m$ use `\pauselevel{:+`$m$`}`, similarly use `\pauselevel{:-`$m$`}` for negative distance. The latter will only be reasonable, if the the current level reduced by $n$ given in `=-`$n$ is not smaller than $m$; otherwise the text will never appear.

Caveat: the relative numbers will always refer to the current level, which is possibly incremented from a previous `\pause` command.
As with the `=` argument there must be no white space between `:` and a following `+` or `-`. But between different specifications at least one space is required.
Example:

```
...  \pause\pauselevel{=3 :5}\textcolor{red}{at}\pause  ...
```

will show the text "at" only in levels 3 to 5.

## 7 Multiple levels

It may be necessary, to have a chunk appear at level $x$, vanish at a level $y$ ($y > x$) and have it reappar at level $z$ ($z > y$). To achieve this, the level specification for a chunk can include multiple ranges. These ranges are separated by a comma.
Example:

```
...  \pause\pauselevel{=3 :5, =8 :10}\textcolor{red}{at}%
\pause  ...
```

will show the text at in levels 3 to 5 and in levels 8 to 10.

## 8 Highlighting

Sometimes it is helpful not to build a page incrementally, but to highlight the chunks and move the highlighted area through the page. Changing the color on a page can be done by the post processor, if it has instructions what and how to change.
To achieve large flexibility highlighting is handled as follows: First you mark all text to be highlighted with a special color. If you do not need the selected color for other purposes, you may pick one of the standard colors. Otherwise you should define a new color. Then you have to declare, in what color items of the selected color should be presented in normal mode and in highlighted mode. This declaration has the form
`\pausecolors`{textcolor}{normalcolor}{highlightcolor}
It is possible to replace more than one color this way. All replacements are cumulated and are done in one step. Processing in one step avoids unexpected results if one color is created from one highlight color replacement and modified by another. It is also possible to replace a color by itself in either normal or highlight mode. Obviously selecting all three colors the same is not reasonable.
Examples:

```
...  \pausecolors{red}{gray}{magenta}%
We highlight only \textcolor{red}{one} word  ...
```

The word "one" will show up in color gray on all normal pages and in color magenta on the highlighted page.

```
...  \pausecolors{red}{gray}{magenta}%
\pausecolors{blue}{gray}{cyan}%
We \textcolor{blue}{can} highlight also \textcolor{red}{more}
words.  ...
```

The word "can" will show up in color gray on all normal pages and in color cyan on the highlighted page, while the word "more" will be highlighted using magenta on the same highlighted page and be in the same gray color on all other pages.

The color mapping specifications take effect for the full page, no matter in which chunk they are specified. They are kept for all following pages. To reset all color mappings you can use the command `\pausecolorreset`.
Please note that highlighting affects not only text material, but also some graphics. Namely everyting on the page which is created with `PDF` commands will be subject to color modifications if the same color model and the same color values are used. An exception are e.g. any embedded jpeg pictures. We have not covered how to specify when highlighting should be used yet.
Highlighting parts of a page temporarily is similar to building a page incrementally. This means that we can use the chunk as the basic unit for highlighting. If a color replacement definition is available, when pages are built incrementally, the newly displayed chunks are subject to highlighting. That means, that the color replacements are activated for them.

### 8.1 Switching to highlight mode

But we introduce also a special highlight mode. In this mode the level number which is assigned to a chunk does not name the level, when a chunk appears, but the level when a chunk is highlighted. In addition the ending level, when a chunk disappears during the incremental build for a page, gives the level, when highlighting ends. If no ending level is specified, the chunk will be highlighted only in one level.
Use the command `\pausehighlight` to switch from incremental build mode to highlight mode for a page. This mode is kept for all following pages until you switch back to build mode explicitly.

### 8.2 Switching to build mode

If you switched to highlight mode for your pages, you may want to come back to incremental build mode. This is done with the command `\pausebuild`. This mode is kept for all following pages until you switch back to highlight mode explicitly. Build mode is selected, when PPower4 starts.
If a color replacement definition is available in this mode, the newly displayed chunks are subject to highlighting and the color replacements are activated for them.

## 9 Mixed modes

It may be necessary to have single chunks in highlight mode also on pages which are processed in build mode and vice versa.

### 9.1 Highlighted chunks during build mode

If a chunk should be visible from the very beginning on a page, which is presented in build mode, it can be tagged with the word `highlight` in the level assignment. Then it will be highlighted in the level(s) indicated by the command `\pauselevel`. Note that automatic highlighting of the chunk on first appearance, which is normally done in build mode, is suppressed for these chunks.
If you just want the chunk to be highlighted on the level, which is assigned to it by default, it is sufficient to specify `\pauselevel{highlight}`.
Example:

```
...  \pause\pauselevel{highlight =2 :5}%
Text highlighted in levels 2 to 5.\pause  ...
```

The highlight tag is ignored in highlight mode.

### 9.2 Build chunks during highlight mode

If a chunk should appear only in selected levels, although highlight mode is active, you must tag the chunk with the word `build`. Note that automatic highlighting of the chunk on first appearance, which is normally done in build mode, is suppressed for these chunks.
Example:

```
...  \pause\pauselevel{build =2 :5}%
Text visible only in levels 2 to 5, not highlighted at all.%
\pause  ...
```

```
...  \pause\pauselevel{build =2 :5, =3}%
Text visible only in levels 2 to 5, highlighted in level 3
only.\pause  ...
```

The build tag is ignored in build mode.

## 10 Backgrounds

There are several background effects supported by PPower4. Some of these effects will at first occur after PPower4 has been invoked, some of them can already be viewed in the file created by `pdflatex`, `vlatex` or `dvipdfm`. The first ones refer to colored backgrounds, the latter ones to adding background elements.

### 10.1 Colored backgrounds

The overall impression of a presentation can be enhanced by using color. 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`. In the meantime recent versions of `pdftex.def` (versions 0.02t and up) support also the `\pagecolor` command to apply a monochrome background.
Unfortunately this definition interferes with the background processing of PPower4. The background applied by `\pagecolor` from `pdftex.def` cannot be removed or replaced by the post processor. It is part of the page. It may even be hard to make a chunk from it without any additional page material. But if you want just one monochrome background for all your pages, using it is absolutely easy, because you do not need to load any additional style files.
On the other hand one has to take into consideration, that delaying the background insertion to post processing time may lead to unreadable documents before application of the post processor, if the foreground color of the text is indistinguishable from the default background color.
The following background selections are available through the style file `background.sty`. Use this file preferably, if you want multi color backgrounds.

`\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, cmyk, or gray).

Backgrounds are kept for subsequent pages until redefined.
A one color vertically fading background is created with

```
\definecolor{bgblue}{rgb}{0.04,0.39,0.53}
\vpagecolor{bgblue}
```

Similarly the following definition creates a horizontally fading background with explicit selection of colors.

```
\definecolor{bgmag}{rgb}{0.7,0.39,0.7}
\definecolor{bgmaglight}{rgb}{0.95,0.83,0.95}
\hpagecolor[bgmag]{bgmaglight}
```

### 10.2 Adding background elements

If you want to add elements like pictures to the background, you have to install the `eso-pic` package first. This package will be automatically included by the `background.sty` style file.
You must tell the `background.sty` style file that you want to add elements to the background with the option `bgadd`:

```
\usepackage[bgadd]{background}
```

After that you can add whatever you want to the background; the elements will be added to the top left corner of the slides:

```
\bgadd{\includegraphics[width=2cm]{myimage.jpg}}
```

You can add an element with a displacement from the top left corner like in the following example:

```
\bgadd{\vspace{1cm}\hspace{2cm}May 2002}
```

To put an element to the center of a slide, use the `\bgaddcenter` command:

```
\bgaddcenter{\includegraphics[width=2cm]{myimage.jpg}}
```

A slide will contain all elements added with the `\bgadd` command together. You can remove the background elements with the command `\bgclear`.

## 11 Links to first builds of a page

When page references are inserted into a document, these will lead you to the complete page. This strategy is fine, if one wants to go back in a presentation and show again a previously shown page. But in some cases one would also need a link to the first partial build of the destination page.
Because the partial builds are inserted by the post processor it must take care of these references. Of course these references cannot be defined completely by `pdftex`.
Managing links to initial page frames is supported by `pp4link.sty`. This style file relies on `hyperref.sty` and introduces the following commands:

`\toptarget{`*name*`}`

defines a name for a target to hyperlink to the first partial page. Then using

`\toplink{`*name*`}{`*text*`}`

makes *text* the active link to the target. To support links forward and backward it is necessary to run `pdflatex`, `vlatex` or `latex` twice. *name* must consist entirely of letters. Digits or any other characters are not allowed in a name.
If you have several pages with the same page number in your document, referring to the a first link of such a page is not defined. The link may take you to any of the pages with this number after postprocessing.

## 12 Incremental/highlighted graphics with MetaPost

With our extension to the Xfig to MetaPost converter you can create easily incremental graphics. The different depths of Xfig's figure will result in separate frames which are overlaid to build the complete figure. The figure will be built stuffing levels into frames decreasing the depth. Because it may be necessary to use overlaying also in one frame, frames will be built from contiguous depth values. If you want to start a new frame, just leave at least one level without any items in Xfig. With the macro `\multiinclude` (from `mpmulti.sty`) all frames are inserted into the document and by default a `\pause` is inserted between the frames.

### 12.1 Options of multiinclude

The optional keyword parameter `pause` of `\multiinclude` can be used to specify another command to be inserted between the frames. Furthermore the optional keyword parameter `graphics` can be used to forward optional parameters to the command `\includegraphics`, e. g. to scale all frames.
Furthermore there are keywords `start` and `end`, which can specify the starting and/or ending number for the includes. The default for `start` is 0 and for `end` it is 1 000 000. But including will stop silently, if after the first frame no more overlays can be found.
To use Xfig's output in `pdflatex` you will have to specify, that the extensions with numbers should be treated as Metapost graphics files using `\DeclareGraphicsRule`. To avoid this you can rename the files `basename.0` ... `basename.<n>` to `basename-0.mps` ... `basename-<n>.mps`. The option `format=mps` will switch to this naming scheme. Make sure that you rename all the files created by `mpost` properly.
If you have other graphics files in a different format, which `\includegraphics` can handle, just rename the files according to the naming scheme shown above and specify the extension using the option `format`.

### 12.2 Special pause effects

If you want some of these frames also to disappear at a certain level, you will need a clever strategy and an option to insert something else than just `\pause`. It is best to assign the highest depths to those frames, which need special level assignments. All other frames, which need to appear just according to their regular depth should follow.
Now assign levels ranges to the first special frames with `\pauselevel`. This is done easily with a command to replace the `\pause` between the frames. We will make use of a counter and insert different `\pauselevel` commands depending on the value of the counter, which is incremented each time. The following example can give you an idea of the operation:

```
\newcount\pausecount      % counter for the iterations
\pausecount=0             % initialize
\def\mypause{\ifcase\pausecount % depending on the counter
  \pauselevel{=-1 :+2}\or % level for first frame (actually 0)
  \pauselevel{=-1 :+3}\or % level for second frame
  \pauselevel{=-1}\else   % level for third frame
  \relax\fi               % all other levels no assignment
  \pause                  % but all will have the \pause
  \advance\pausecount1\relax} % and increment the counter
```

### 12.3 mpost processing

To include the MetaPost graphics into your document, you have to convert the MetaPost files to PostScript first. This is done with `mpost`. Notice that if you are using `vlatex` you must ensure that

the PostScript files created by `mpost` are self-contained PostScript files (EPSF). That means that there must be a font definition part at the beginning of the PostScript files. Otherwise `vlatex` will complain that there are undefined names.
You can tell `mpost` to create self-contained PostScript (EPSF) by adding the line

```
  prologues:=2;
```

at the beginning of the MetaPost file. If you have created the MetaPost file from a fig file with `fig2dev`, you can tell `fig2dev` to add this line by using the option `-p 2` (make sure to use a version of `fig2dev` which is newer than the one from a default Xfig 3.2.3d!).
In contrast if you are using `dvipdfm`, you must ensure that the included PostScript file is *not* self-contained; the MetaPost file must not have the line described above, or invoke `fig2dev` *without* the option `-p 2`. `pdftex` can use the default version and the self-contained one.

## 13 More tricks

If you need to show different texts in the same space at different levels, you must hide the width of one of the texts for TEX. Normally it is advisable to hide the shorter text and have the longer text be used in page breaking.
Example:

```
...  \pause \pauselevel{=3 :8}\rlap{2}%
     \pause \pauselevel{=9}8\pause  ...
```

The number 2 will appear in level 3 and stay until level 8. From level 9 the number 8 will take the same place.
– more to come – (e.g. inserting `\pause` commands into the preamble of a table, see the demo for now)

## 14 Files

The post processor is selfcontained in one jar archive, which needs to be called in a Java Runtime Environment.
Additionally you need the style files to be present during formatting.
– to be extended –