Active-DVI

Reference manual
Version 1.4

Didier Rémy and Pierre Weis

January 24, 2003

Active-DVI is a viewer for DVI files that also recognizes a new class of `\special`'s targeted to presentations via laptop computers: various special effects can easily be incorporated to the presentation, via a companion `advi` LaTeX package.

Acknowledgements and contributors

Active-DVI is based on Mldvi which constitutes its core rendering engine. Alexandre Miquel wrote Mldvi and distributes it under the LGPL license at URL `http://pauillac.inria.fr/~miquel/`.

Active-DVI has been developed by Jun Furuse, Didier Rémy and Pierre Weis with also contributions by Roberto Di Cosmo, Didier Le Botlan, Xavier Leroy, and Alan Schmitt.

Contents

1 Installation

2 Active-DVI for the impatient

- As a previewer, Active-DVI can preview any correct DVI file.
- As a presenter, Active-DVI provides some LaTeX packages to facilitate animations and interaction with the presenter from within your LaTeX source text. The `advi-slides.sty` package is designed to be a simple way to build a presentation for Active-DVI.

Here is a simplistic talk example, to begin with. As a first simple interaction with the presenter, write `\pause` into your LaTeX code when you want the presenter to stop when it is displaying a slide.

```
\documentclass[landscape]{slides}

% The mandatory definition of the \footer macro
% Footer is empty, if you write \def\footer{}
\def\footer{{\hfill \em {Me.Myself@institut.fr\hfill 24-01-2003}}}

\usepackage{advi-slides}

\begin{document}
\firstslide{My talk}
 \vspace*{5cm}\centerline{{\large \bf \em Me Myself}}
\newslide{Plan}
This talk is divided into $n$ parts: Part 1, Part 2, and Partn.
\newslide{Part 1}
This is my first talk using {\ActiveDVI}!
\newslide{Part 2}
Bla bla.
\newslide{Part n}
Blz blz.
\newslide{Conclusion}
This was my first talk.
\end{document}
```

More involved examples (including Makefiles to compile the whole thing) can be found in the directory `examples` of the distribution. (In particular, this example is given in `examples/slitex/simplistic`.)

3 Safety concerns when using the Active-DVI previewer

Warning! Active-DVI may execute programs and commands embedded into the DVI file. Hence, when playing a DVI file from an untrusted source, you should run `advi` with the

`-safer` option that inhibits the execution of embedded applications. This warning applies in particular if you choose Active-DVI as your default `meta-mail` previewer for the `application/x-dvi` mime-type.

The default safety option is the `-ask` option: it tells Active-DVI to ask the user each time it must launch an application. (Note that in such a case Active-DVI asks only once to launch a given application: it remembers your previous decisions concerning the command and acts accordingly for the rest of the presentation.)

The second safety option is the above mentioned `-safer` option: it completely inhibits the execution of embedded applications.

The last safety option is `-exec`: if you call `advi -exec`, `advi` automatically and silently launches all embedded applications (this is useful to play your own presentations without the burden of answering yes to Active-DVI's questions).

As mentioned, the safe `-ask` option is the default, automatically set when nothing has been explicitly specified by the user. If desired, the default safety option can be set via *initialisation files*, either on a system large scale by the machine administrator (in the file `/etc/advirc`), on a local scale by individual users (setting the default policy for that user), or even on a per directory basis (setting the default policy to show DVI files in this directory)! (This last option is convenient to gracefully run your own talks, while still being cautious when running talks from others.)

4 Initialisation files for Active-DVI

4.1 Syntax of initialisation files

An initialisation file for Active-DVI is simply a text file that contains options exactly similar to those you can give on the command line (with the exception of comments, made of a sharp sign (`#`) followed by some text that is ignored until the end of line). For instance:

```
-exec # I know what I mean!
-bgcolor grey16
-fgcolor grey95
```

is a valid initialisation file that sets the safety policy to `-exec`, then sets the background and foreground colors to obtain a nice reverse video effect.

4.2 Loading initialisation files

Before parsing options on the command line, Active-DVI loads, in the order listed below, the following initialisation files (nothing happens if any of them does not exist):

- system wide initialisation file: `/etc/advirc`,
- user specific initialisation files: `~/.advirc` then `~/.advi/advirc`,
- directory specific initialisation file: `./.advirc`.

4.3 Automatic setting of options

In addition, the user may load an arbitrary file containing options by specifying the file path via the command line argument `-options-file`. Hence, `-options-file filename` loads `filename` when parsing this option to set up the options contained in `filename` (thus overriding the options set before by the default `/.advirc` or `/.advi/advirc` init files).

5 Using the Active-DVI previewer

5.1 Command line options

Active-DVI is invoked with the following command syntax

```
advi [options] dvifile
```

The `advi` commands recognized the following options:

Help and info options

`-v`	Prints the `advi` current version and exits
`--version`	Prints the full `advi` current version and exits
`-help`	Short command line options help

Window and display specifications

`-geometry geom`	Geometry of Active-DVI's window specification

Geometry `geom` is specified in pixels, using the standard format for XWindow geometry specifications (i.e: `WIDTHxHEIGHT[+XOFFSET+YOFFSET]`).

`-fullwidth`	Adjust the size of the window to the width of the screen

`-nomargins`	Cancel horizontal and vertical margins
`-hmargin dimen`	Horizontal margin specification (default: 1cm)
`-vmargin dimen`	Vertical margin specification (default: 1cm)

Dimensions are specified as numbers optionally followed by two letters representing units. When no units are given, dimensions are treated as numbers of pixels. Currently supported units are the standard TeX units as specified in the TeXbook (D. Knuth, Addison-Wesley, (C) 1986): 'pt' (point), 'pc' (pica), 'in' (inch), 'bp' (big point), 'cm' (centimeter), 'mm' (millimeter), 'dd' (didot point), 'cc' (cicero) and 'sp' (scaled point). Note that dimensions are specified w.r.t the original TeX document, and do not correspond to what is actually shown on the screen, which can be displayed at a different resolution than specified in the original TeX source.

`-crop`	Crop the window to the best size (default)
`-nocrop`	Disable cropping

Color specifications

`-fgcolor <color>`	Specify the color of the foreground color
`-bgcolor <color>`	Specify the color of the background color
`-rv`	Specify that reverse video should be simulated by exchanging the backgound ar

Helpers specification

`-pager`	Specify the name of the pager to launch on a `txt` link
`-browser`	Specify the name of the browser to a `html` link

Debugging options

`--debug`	General debug option
`--debug_pages`	Debug page motion
`--show_ps`	Print a copy of Postscript sent to gs to stdout
`--verbose_image_access`	Change the cursor while loading images

Rendering options

`-A`	Toggle Postscript antialiasing
`-passive`	Inhibit effects that are visible when redrawing the page (Transitions, delays, embedded applications)

Safety options

`-exec`	Set safety policy to “automatic execution”
`-ask`	Set safety policy to “ask user before execution”
`-safer`	Set safety policy to “no execution of embedded aplications”

Option files option

`-option-file <filename>`	Load `filename` as a file containing a list of options as given on the command line to `advi`.

5.2 Cut and paste

Text can be copied from the Active-DVI previewer to an another application. However, this uses the `XBuffer` and not the `XSelection` mechanism.

- Shift middle-click copies the current word.
- Shift right-click and drag copies the specified region.

Moreover, Shift left-click dump an ASCII representation of click in the source file. This expects the DVI to be instrumented with line numbers of the form `line:` ⟨*line*⟩ ⟨*file*⟩

where ⟨*line*⟩ and ⟨*file*⟩ are the current source line and current source file.

The position is exported in ASCII in the form

`#line` ⟨*before*⟩, ⟨*after*⟩ `<<`⟨*prefix*⟩`>><<`⟨*suffix*⟩`>>` ⟨*file*⟩

where ⟨*before*⟩ and ⟨*after*⟩ are the enclosing line numbers, ⟨*prefix*⟩⟨*suffix*⟩ are the word constituent surrounding the position, and file is the current file.

Line numbers default to 0 when not found. Note that line numbers may be inconsistent if they `\special`-line commands have not been inserted close enough to the position.

5.3 Hyperref

Active-DVI supports the LaTeX `hyperref` package with both internal and cross-file references. For cross-file references, it launches a new `advi` process to view the target.

Active-DVI improves the treatment of hyperrefs over convential previewers, by emphasizing the hypertarget text of an hyperlink. Thus, an hypertarget definition

`\hypertarget{`⟨*tag*⟩`}{`⟨*text*⟩`}`

should make the activation of the link ⟨*tag*⟩ not only move to the page where ⟨*text*⟩ occurs, but also emphasize the target ⟨*text*⟩. However, since `\hypertarget` does include its second argument withing the target, we use the following command instead:

```
\newcommand{\softtarget}[2]%
  {\special{html:<A name=\hyper@quote #1\hyperquote>}#2\special{html:</A>}}
```

(If you are viewing this document with Active-DVI, you may move over or click on this area to see the effect.)

5.4 Floating table of contents and thumbnails

There are two ways to include a floating table of contents while previewing.

- Active-DVI recognizes the reserved hypertargets `/toc.first` and `/toc.last` as markers for the first and last pages of the table of contents. These pages then become floating, *i.e.* accessible from anywhere in the document with the default keybinding `t`. The first stroke on `t` shows the first page of the table of contents. Successive stokes will show the following pages. (As usual, prefix integer argument may be used to directly access a specific page of the table of contents.)

  The package `advi` below redefines the macro `\tableofcontents` so that it automatically inserts the reserved hypertargets markers around the table of contents. It also provides two new macros, `\advitoc` and `\endadvitoc`, that serve to insert these markers when the table of contents is hand-made.

- If no table of contents markers are found, then Active-DVI will compute thumbnails, *i.e.* will show the whole set of pages of the presentation, each page drawn at a smaller scale and packed with the others on a single page. Active-DVI computes the scale so that all the thumbnails fit on one page only, provided that the scale is less or equal to a maximal value; otherwise, the maximal value scale is selected and the thumbnail pages spread on several pages. The default maximal scale value is 5, so that 25 thumbnails can fit on the same page. This value can be changed using the option `-thumbnail_scale`.

  Normally, thumbnails are drawn for all the pages. However, thumbnail pages can also be defined manually, with an hypertarget whose anchor is of the form `/page.`⟨*suffix*⟩. In this case, all the desired thumbnails must be explicitly marked.

By default, the binding `T` processes thumbnails and the binding `t` displays thumbnails if already processed, or shows the table of contents if available. Otherwise pressing `t` has no action. Thumbnails computation is explicit, so that incidentally hitting the `t` key does not lead to an unexpected computation, hence an unexpected delay.

5.5 Moving around

See the key bindings in the appendix.

5.6 Using and making special effects

Presentation examples can be found in the `examples` directory. Don't miss to play them! Then, feel free to read their source code and copy the effects they provide.

Active-DVI can be used as is, but will shine when driven by a user with a bias toward programming: special effects can easily be realized by using the LaTeX packages provided with the distribution.

Creative advanced users may program the presenter at various levels, either using or defining simple LaTeX macros, writing new LaTeX package files, or by implementing extensions to the previewer itself.

6 The `advi.sty` LaTeX package

Active-DVI provides some LaTeX packages to facilitate animations and interaction with the presenter from within your LaTeX source text.

The `advi.sty` package is the main package to include when writing a presentation for Active-DVI. It defines the main set of interactive commands for Active-DVI to animate the show. However, there is no need to load the package if no Active-DVI special effects are required for the presentation.

6.1 Printing the presentation

The `advi` package recognizes the special option `ignore`, which is devoted to help the production of a printable version of the presentation: the `ignore` option makes the package not to produce `advi` specials, so that the show can be previewed by other previewers or turned into Postscript with `dvips`. Of course, this option disables most effects that cannot be printed, although some of them are still approximated.

If the `ignore` option is not set globally, it can be set locally with the commands `\adviignore`. However, this will not prevent all effects, since some decisions are taken when the package is loaded.

The package also defines the conditional `\ifadvi` which evaluates its first argument if advi is not in ignore mode and its second argument otherwise.

6.2 Pause, Record and Play

`\adviwait[`⟨*seconds*⟩`]`

> Wait for ⟨*seconds*⟩. If no argument is provided, waits until the user requests to continue (hitting a key to move to next pause or to change page).

`\advirecord[play]{`⟨*tag*⟩`}{`⟨*latex code*⟩`}`
`\begin{advirecording}[play]{`⟨*tag*⟩`}{`⟨*latex code*⟩`}`⟨*text*⟩`\end{advirecording}`

> Processes ⟨*latex code*⟩ and records the corresponding DVI output, bound to the tag ⟨*tag*⟩. When recording, the DVI output is not displayed unless the option `play` is set.
>
> Records can be nested. If so, the inner record is always recorded with its own tag; if the inner record is played when recording, it is also recorded as part of the outer tag.
>
> If the environment form is used, the ⟨*latex code*⟩ may contain fragile commands.

`\adviplay[`⟨*color*⟩`]{`⟨*tag*⟩`}`

> Replay the DVI previously recorded and bound to ⟨*tag*⟩. The optional argument changes the color to ⟨*color*⟩ during replay.

`\advianchor[`⟨*activation*⟩`]{`⟨*tag*⟩`}{`⟨*text*⟩`}`
`\begin{advianchoring}[`⟨*activation*⟩`]{`⟨*tag*⟩`}`⟨*text*⟩`\end{advianchoring}`

> Plays the record bound to ⟨*tag*⟩ when the anchor is activated.
>
> The argument ⟨*activation*⟩ may either be `over` or `click`. The page is reset to its original appearance when the anchor is no more activated (the mouse leaves the anchor area or the button is released).
>
> If the environment form is used, ⟨*text*⟩ may contain fragile commands.

6.3 Images

Images can be encapsulated into the presentation using the Caml library CamlImages provided with the distribution of Active-DVI (see section 7.4).

6.4 Colors

The `color` package

In Active-DVI, colors can be specified with the conventions of the LaTeX package `color.sty`, that is, it should either be a previously defined color or a specification of the form `[`⟨*model*⟩`]` `{`⟨*model color specification*⟩`}`.

Named colors

Colors can be named using the keyword ⟨*named*⟩. If you use named colors, the color names are case sensitive and should generally be capitalized; for instance: `\color[named]{White}` specifies the white color. Hence, `\color[named]{Red}{`*some text*`}` writes *some text* in red.

The names of available colors can be found in the `dvipsnam.def` file, generally at location `/usr/share/texmf/tex/latex/graphics/dvipsnam.def`.

To give an idea, the names and colors available on a standard installation of LaTeX are:

GreenYellow Yellow Goldenrod Dandelion Apricot
Peach Melon YellowOrange Orange BurntOrange
Bittersweet RedOrange Mahogany Maroon BrickRed
Red OrangeRed RubineRed WildStrawberry Salmon
CarnationPink Magenta VioletRed Rhodamine Mulberry
RedViolet Fuchsia Lavender Thistle Orchid DarkOrchid
Purple Plum Violet RoyalPurple BlueViolet
Periwinkle CadetBlue CornflowerBlue MidnightBlue
NavyBlue RoyalBlue Blue Cerulean Cyan ProcessBlue
SkyBlue Turquoise TealBlue Aquamarine BlueGreen
Emerald JungleGreen SeaGreen Green ForestGreen
PineGreen LimeGreen YellowGreen SpringGreen
OliveGreen RawSienna Sepia Brown Tan
Gray Black White

The CMYK specifications of colors

You may also explicitely use a CMYK (Cyan, Magenta, Yellow, Black) specification. In this case the cyan, magenta, yellow and black values follow the ⟨*cmyk*⟩ keyword, and are given as a list of four integers in the range 0 .. 255. For instance, `\color[cmyk]{0,1,0,0}` is a valid specification for magenta.

The RGB specifications of colors

RGB (Red, Green, Blue) specifications are similar to the CMYK specifications: following the ⟨*rbg*⟩ keyword, the red, green, or blue color values, are given as floating point numbers in the range 0.0 .. 1.0. Hence, `\color[rgb]{1.0,0.0,0.0}` is a valid specification for red.

X colors

Active-DVI provides the package `xcolor`, an extension to the `color` package, that defines a large set of X colors names, as found in the file `rgb.txt` of a typical X installation (this file is generally located on `/usr/X11R6/lib/X11/rgb.txt`). To know which colors are available look at the source file of package `xcolor.sty` in the directory `tex` of the distribution.

6.5 Background

Background of pages can be set in the LaTeX source of the page, either as a plain color or as an image (or both!). You can specify a global option to the background setting, so that this background will be used for the remaining pages of the presentation (otherwise the presenter resets background options at each new page).

Background images can be lighten by specifying an alpha value (a floating point number between 0 and 1) that mesures the mixing between the color of the background and the image.

Background images can also be blended, meaning that you can choose the algorithm that surimposes the image to the background.

`\advibg[global]{`⟨*decl*⟩`}`

where ⟨*decl*⟩ is a list of settings of the following from:

`color=`⟨*color*⟩ (default value is `none`)

Set the background color to ⟨*color*⟩. If ⟨*color*⟩ is `none` this unsets the background color. Otherwise, ⟨*color*⟩ must follow the convention of the package `color`, that is, it should either be a previously defined color or of the form `[`⟨*model*⟩`]{`⟨*model color specification*⟩`}`.

For example, the following specifications are all correct:

```
color=blue
color=[named]{Yellow}
color=[rgb]{0.7,0.3,0.8}
```

`image=`⟨*file*⟩ (default value is `none`)

Use the image found in ⟨*file*⟩ as background (`none` means unset).

`fit=`⟨*fit style*⟩ (default value is `auto`)

Fit the background image according to ⟨*style*⟩, which may be one of the following keywords:

		`topleft`	`top`	`topright`
`auto`	or	`left`	`center`	`right`
		`bottomleft`	`bottom`	`bottomright`

The `auto` fit style means scaling the image as desired in both directions so that it fits the entire page. Other styles only force the same scaling factor in both directions:

- Corner-styles means set the image in the corresponding corner and scale it to cover the entire page.
- `center` means set the image in the center of the page and scale it to cover the entire page.
- Segment-styles means adjust the image and the page on the segment (in which case, the image may not completely cover the page on the opposite side).

`alpha=`⟨*float*⟩ (default value is `none`)

Set the alpha channel factor for the background image to ⟨*float*⟩ (`none` means unset). An alpha factor of 0 means that the image is not visible at all; conversely, an alpha factor of 1 means that the image covers the background.

`blend=`⟨*blend mode*⟩ (default value is `none`)

Set the blend mode to ⟨*blend mode*⟩, which are reminiscent of Ghostscript blending options. The blend mode should be one of the following: `normal`, `multiply`, `screen`, `overlay`, `dodge`, `burn`, `darken`, `lighten`, `difference`, `exclusion`, (`none` means unset).

`none`

Unset all background parameters. This key must appear on its own, no arguments or keys are allowed.

The optional parameter `global` indicates that the definition is global and will affect the following pages, as well as the current page.

By default, the background settings only affect the current page.

6.6 Transitions

`\advitransition[global]{`⟨*decl*⟩`}`

where ⟨*decl*⟩ is a list of settings of the following from:

`none` or `slide` or `block` or `wipe`

Set the transition mode to the corresponding key. One of this key is mandatory (if several are provided the last one is selected).

`from=`⟨*direction*⟩

Make the transition come from ⟨*direction*⟩. Directions should be one of the following:

```
topleft         top         topright
left           center          right
bottomleft     bottom    bottomright
```

The default direction, to be used when no local or global direction has been specified, is determined dynamically: `right` when coming from previous page, `left` when coming from next page, and `top` otherwise.

`steps=`⟨*n*⟩

Make the transition in ⟨*n*⟩ steps.

As for `\advibg`, the optional parameter `global` indicates that the definition is global and will affect the following pages, as well as the current page.

By default, the transition definitions only affect the current page.

`\advitransbox`{⟨*key=val list*⟩}{⟨*hbox material*⟩}

where ⟨*key=val list*⟩ is as above and {⟨*hbox material*⟩} is whatever can follow an `\hbox` command. In particular, the material may contain verbatim commands, since as for the `\hbox` it is parsed incrementally.

The optional parameter `global` indicates that the definition is global and will affect the following pages, as well as the current page.

By default, the transition affects only the current page.

6.7 Embedded applications

Active-DVI can launch arbitrary applications you need to animate your show.

6.7.1 Launching embedded applications

The LaTeX command to launch an application during the presentation is

`\adviembed[`⟨*key=value list*⟩`]`{⟨*command*⟩}

where ⟨*key=value list*⟩ is a list of bindings of the following kind:

`name=`⟨*name*⟩

Allows to refer to the embedded application as ⟨*name*⟩. Anonymous applications have actually the default name `anonymous`.

`ephemeral=`⟨*name*⟩

This is the default case: the application is specific to a given page. Launched whenever the page is displayed, the application is killed when the page is turned.

`persistent=`⟨*name*⟩

Launched only once, the application keeps running in the background, but is only visible on the page where it has been launched.

`sticky=`⟨*name*⟩

Launched only once, the application keeps running and remains visible when turning pages. It is also resized and moved as necessary to fit the page size.

`raw=`⟨*name*⟩

The application is launched each time its embedding command is encountered. A `raw` application is not managed by Active-DVI, except for the initial launching and the final clean-up that occurs when Active-DVI exits. You can monitor `raw` applications with `advikillembed` and the associated window mapping facilities for `raw` applications (see below).

`width=`⟨*dim*⟩
`height=`⟨*dim*⟩

The application takes ⟨*dim*⟩ width (respectively height) space in LaTeX. Both values default to 0pt.

These dimensions are also substituted for all occurrences of `!g` in the command string.

6.7.2 Monitoring embedded applications

To monitor embedded applications, Active-DVI provides the `advikillembed` primitive to send a signal to any named embedded application. For `raw` applications, there are additional functions to map or unmap the window allocated to a named `raw` application. Mapping or unmapping windows of non-`raw` applications is unspecified, since it may interfere in a non trivial way with Active-DVI's automatic treatment of those applications.

Monitoring a single application

`\advikillembed{`⟨*name*⟩`}`

Kill the embedded application named ⟨*name*⟩. An optional signal value or symbolic name can be given to send to the designed process: for instance, `\advikillembed[SIGUSR1]{clock}` will send the `SIGUSR1` signal to the embedded application named `clock`.

Signal value defaults to -9.

`\advimapembed{`⟨*name*⟩`}`

Map the window of the embedded applications named ⟨*name*⟩.

`\adviunmapembed{`⟨*name*⟩`}`

Unmap the window of the embedded applications named ⟨*name*⟩.

Monitoring a group of embedded applications

The primitives `advikillallembed`, `advimapallembed`, and `adviunmapallembed` behave the same as their non-`all` counterparts, except that they operate on all the applications that have been launched with the given name.

`\advikillallembed{`⟨*name*⟩`}`

Similar to `advikillembed` but kill all the embedded applications named ⟨*name*⟩.

`\advimapallembed{`⟨*name*⟩`}`

Map the windows of all the embedded applications named ⟨*name*⟩.

`\adviunmapallembed{`⟨*name*⟩`}`

Unmap the windows of all the embedded applications named ⟨*name*⟩.

6.8 Active anchors

Active anchors are annotated pieces of text that get associated activation records. To define an active anchor, the command is

`\advianchor[`⟨*decl*⟩`]{`⟨*tag*⟩`}{`⟨*text*⟩`}`
`\begin{advianchor}[`⟨*decl*⟩`]{`⟨*tag*⟩`}`⟨*text*⟩`\end{advianchor}`

The text is first displayed as usual, then the anchor is drawn according to the style given by ⟨*decl*⟩, and made active. Its activation, which depends on the mode given by ⟨*decl*⟩, will play the record named ⟨*tag*⟩.

The declarations ⟨*decl*⟩ are of the following form:

`over` or `click`

this defines the mode of activation, which is either by moving mouse over or clicking on the anchor. The default mode is `over`.

`box`, `invisible`, or `underline`
: this defines the style in which the anchor should be drawn. The default style is `boxed`.

In the environment form, ⟨*text*⟩ may contain fragile commands.

`\adviemphasize[`⟨*color*⟩`]{`⟨*text*⟩`}`

This makes an invisible anchor around ⟨*text*⟩, which when activated will redraw text in a box colored with ⟨*color*⟩, which defaults to `yellow`.

6.9 Postscript specials

Active-DVI can deal with most of PStricks by calling `ghostscript` on included Postscripts. However, the interactivity between Active-DVI and Postscripts is not always properly working.

- Since characters are rendered by Active-DVI, some PStricks are not allowed.
- Some change of repairs are also not yet correctly performed.

6.9.1 Overlays

The `overlay` class implements overlays with PStricks. By contrast, Active-DVI implements overlays directly, using records and plays. This is more efficient, and of course more natural. (In fact, Active-DVI chooses the cumulative semantics of overlays, displaying all layers below the current overlay.)

The `xprosper` style, derived from the `prosper` class, uses the `overlay` class and works with Active-DVI in exactly the same way (relaxing the @loop macro inhibits all layers, but the first page).

6.9.2 PStricks

Active-DVI can deal with most of PStricks.

+ Simple drawings work
+ `\SpecialCoor` works, *i.e.* commands of the form `\rput{A}{bla bla}` works where `A` is a node
+ Connections between nodes `\ncarc`, `\ncarc`, also works.

However, some PStricks are known not to work.

– Labels over arrows `\Aput`, `Bput`, etc. (they change the Postscript coordinates...)

– `pspicture` (idem, but drawings that are not embedded in pspictures work).

7 Auxilliary LaTeX packages

7.1 The superpose package

This package allows superposition of horizontal material, creating the smallest horizontal box that fits all of the superpositions.

```
\usepackage{superpose}
```

The package defines a single environment:

`\begin{superpose}[`⟨*alignment*⟩`]`⟨*list*⟩`\end{superpose}`

The ⟨*alignment*⟩ can be one the letters `c` (default value), `l`, or `r`.

Items of the list are separated by `\\` as in tabular environments. Each item should be a horizontal material.

7.2 The bubble package

This package draws bubbles over some text.

```
\usepackage{bubble}
\usepackage[ps]{bubble}
```

By default bubbles are produced using the `epic` and `eepic` packages, for portability. However, for better rendering and easier parameterization, bubbles can also be drawn using the `pst-node` package of the pstricks collection. This is what the `ps` option is designed for.

The package defines a single command:

`\bubble[`⟨*key=value list*⟩`]{`⟨*anchor*⟩`}[`⟨*ps options*⟩`](`⟨*pos*⟩`){`⟨*text*⟩`}`

The ⟨*key=value list*⟩ is a list of bindings of the following kind:

`bg=`⟨*color*⟩ (default value is `yellow`)

The background color for annotations.

`unit=`⟨*dim*⟩ (default value is `yellow`)

Set the package unit to ⟨*dim*⟩.

`col=`⟨*colspec*⟩ (default value is `c`)

Where ⟨*colspec*⟩ is a column specification for the tabular environment. Moreover, the following abbreviations are recognized:

key	expands to
`c`	`col=c`
`l`	`col=l`
`r`	`col=r`
`p=`⟨*w*⟩	`col=p{`⟨*w*⟩`}`

key	expands to
`C`	`col={>{$}c<{$}}`
`L`	`col={>{$}l<{$}}`
`R`	`col={>{$}r<{$}}`
`P`⟨*w*⟩	`col={>{$}p{`⟨*w*⟩`}<{$}}`

⟨*pos*⟩ is the optional relative position of the annotation, it defaults to $1, 1$, and is counted in the package units.

⟨*ps options*⟩ are passed to the command `\psset`) in `ps` mode and ignored otherwise.

Parameters (color and tabular columns specifications) can also be set globally using the command:

`\setkeys{bubbleset}{`⟨*key=value list*⟩`}`

7.3 The annotations package

This package uses active anchors and the bubbles package to provide annotations by raising a bubble when the cursor is over the anchor.

The package defines a single command

`\adviannot[`⟨*key=value list*⟩`]{`⟨*anchor*⟩`}[`⟨*ps options*⟩`](`⟨*pos*⟩`){`⟨*text*⟩`}`

whose options are identical to those of the `\bubble`macro; however the bubble appears within an active anchor.

7.4 The advi-graphicx package

This 3-lines long package loads the `graphicx.sty` package and provides declarations so that JPEG, EPS, TIF, TIFF source images can be embedded: Active-DVI will preview these images directly while other drivers will translate them on demand.

A Limitations

Postscript Fonts

Postscript fonts are not natively handled by Active-DVI. You must use the command `dvicopy` to expand those virtual fonts to base fonts before visualization with Active-DVI. (For instance, `dvicopy talk.dvi talk.expanded; advi talk.expanded` very often does the trick.)

Inlined Postscript and Ghostscript

PS relies on `ghostscript` to display Postscript inlined specials. However, some earlier releases of `ghostscript` implements the Postscript `flushpage` command as a `XFlush` call which does not force the evaluation of commands, and thus makes the synchronization between `ghostscript` and Active-DVI drawings uncontrollable. In this case, the interleaving of inlined postscript and other material may be inconsistent.

Fortunately, recent versions of ghostscript (> 6.5) have fixed this problem by using `XSync(false)` instead. If you use those versions of ghostscript, inlined specials should be correctly rendered.

Unfortunately, some releases of version 6.5x also carry a small but fatal bug for Active-DVI, that will hopefully be fixed in future releases. A workaround is available here `http://cristal.inria.fr/~remy/ghostscript/`.

Inlined Postscript change of coordinates

So far, the implementation of inlined Postscript does not correctly handle complex change of coordinates. (See PStricks section).

B Reporting bugs

Please, send bug reports to `mailto:advi@pauillac.inria.fr`.
See `http://pauillac.inria.fr/advi` for up to date information.

C Key bindings

Advi recognizes the keystrokes listed below when typed in its window. Some keystrokes may optionally be preceded by a number, called arg below, whose interpretation is keystroke dependant. If arg is unset, its value is 1, unless specified otherwise.

Advi maintains an history of previously visited pages organized as a stack. Additionnally, the history contains marked pages which are stronger than unmarked pages.

?	info	–	Quick info and key bindings help.
q	quit	–	End of show.
space	continue	–	Move arg pauses forward if any, or do as return otherwise.
n	next	–	Move arg physical pages forward, leaving the history unchanged.
p	previous	–	Move arg physical pages backward, leaving the history unchanged.
g	go	–	If arg is unset move to the last page. If arg is the current page do nothing. Otherwise, push the current page on the history as marked, and move to the physical pages arg .
,	begin	–	Move to the first page.
.	end	–	Move to the last page.
T	Thumbnails	–	Process thumbnails.
t	toc	–	Display thumbnails if processed, or floating table of content if available, or do nothing.
a	active/passive	–	toggle advi effects (so that reloading is silent).

N	next pause	–	Move arg pauses forward (equivalent to continue).
P	previous pause	–	Move arg pauses backward.
ˆf	fullscreen	–	Adjust the size of the page to fit the entire screen or reset the page to the default size (toggle).
<	smaller	–	Scale down the resolution by scalestep (default $\sqrt{\sqrt{\sqrt{2}}}$).
>	bigger	–	Scale up the resolution by scalestep (default $\sqrt{\sqrt{\sqrt{2}}}$).
c	center	–	Center the page in the window, and resets the default resolution.

#	fullpage	–	Remove margins around the page and change the resolution accordingly.
ˆL	redisplay	–	Redisplay the current page to the first pause of the page.
r	redraw	–	Redraw the current page to the current pause.
R	reload	–	Reload the file and redraw the current page.

h	page left	–	Moves one screen width toward the left of the page. Does nothing if the left part of the page is already displayed
l	page right	–	Moves one screen width toward the right of the page. Does nothing if the right part of the page is already displayed
j	page down	–	Moves one screen height toward the bottom of the page. Jumps to the top of next page, if there is one, and if the bottom of the page is already displayed.
k	page up	–	Moves one screen height toward the top of the page. Jumps to the bottom previous page, if there is one, and if the top of the page is already displayed.

return	forward	–	Push the current page on the history stack, and move forward n physical pages.
tab	mark and next	–	Push the current page on the history as marked, and move forward n physical pages.
backspace	back	–	Move arg pages backward according to the history. The history stack is poped, accordingly.
escape	find mark	–	Move arg marked pages backward according to the history. Do nothing if the history does no contain any marked page.
f	load fonts	–	Load all the fonts used in the documents. By default, fonts are loaded only when needed.
F	make fonts	–	Does the same as f, and precomputes the glyphs of all characters used in the document. This takes more time than loading the fonts, but the pages are drawn faster.
C	clear	–	Erase the image cache.
s	scratch	–	Give a pencil to type characters on the page.
S	scratch	–	Give a pencil to draw lines on the page.

D Index