Active-DVI

Reference manual
Version 1.2

Didier Rémy and Pierre Weis

January 20, 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, written by Alexandre Miquel, which constitutes its core. Mldvi alone is distributed under LGPL license and is available at `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 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.

2.1 Command line options

Active-DVI is invoked with the following command syntax

```
advi [options] dvifile
```

The `advi` commands recognized the following options:

`-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

Helpers specification

`-pager`
`-browser`

Debugging options

`--debug`
`--debug_pages`
`--show_ps`
`--verbose_image_access`

Rendering options

`-A` Set Postscript antialiasing

2.2 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.

2.3 Hyperref

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

2.4 Moving around

See the key bindings in the appendix.

2.5 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 LaTeXpackage 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 extension to the previewer itself.

3 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` package is the main package to include when writing a presentation for Active-DVI. It provides 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.

3.1 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.

3.2 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.

3.3 Images

Images can be encapsulated into the presentation using the package CamlImages provided with the distribution (see section 4.4)

3.4 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]{`⟨*key=value list*⟩`}`

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

`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 declared 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 should be one of the following (unchecked): `difference`, XXX? (`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.

3.5 Transitions

`\advitransition[global]{`⟨*key=val list*⟩`}`

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

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

Set the transition mode to the corresponding key. One of this key is mandatory (several is should be illegal, but in fact overrides the previous one).

`from=`⟨*direction*⟩

Make the transition come from ⟨*direction*⟩. Directions are not checked but should be one of the following:

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

`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.

3.6 Hyperref

Active-DVI supports the `hyperref` LaTeX package.
Explain `<a name>XXX</a>`

3.7 Embedded applications

Active-DVI can launch arbitrary applications you need to animate the show.
The LaTeX command 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 embedded application is killed when the page is turned.

`persistent=`⟨*name*⟩

The application keeps running in the background, but will only be visible in the page where it has been launched.

`sticky=`⟨*name*⟩

The application keeps running and remains visible when turning pages.

`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.

`\advikillembed{`⟨*name*⟩`}`

Kill all embedded applications named ⟨*name*⟩

3.8 Active anchors

The command is

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

> The text is first display as usual and mark as active. Moving the mouse above the text will display the record ⟨*tag*⟩.
>
> In the environment form, ⟨*text*⟩ may contain fragile commands.

3.9 Postscript specials

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

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

3.9.1 Overlays

The `overlay` class implements overlays with PStricks. Instead, 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 display if all layers below the current overlay.)

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

3.9.2 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 works)

4 Auxiliary LaTeX packages

4.1 The superpose package

This package allows to easily superpose horizontal material, creating the smallest horizontal box to fit 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.

4.2 The bubble package

This package allows to easily draw 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*⟩`}`

4.3 The annotations package

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

The package provides a single command

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

whose options are identical to those of `\bubble`, but the bubble appears withing an active anchor.

4.4 The advi-graphicx package

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

A Limitations

Postscript Fonts

Postscript fonts are not currently supported.

Inlined Postscript and Ghostscript

PS relies on `ghostscript` to display Postscript inlined specials. However, some earlier releases of `ghostscript` implements 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 Bug Reports

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.

N	next pause	–	Move arg pauses forward (equivalent to continue).
P	previous pause	–	Move arg pauses backward.
^f	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.
^L	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