

The fancytooltips package*

Robert Marik
marik@mendelu.cz

May 7, 2007

1 Introduction

The package `fancytooltips` is a package for L^AT_EX. The pdf can be created by `pdflatex` or by `latex + dvips + AdobeDistiller`¹ + Adobe Acrobat². It allows to create tooltips in a similar way like `cooltooltips` package, but the tooltip is a page from another PDF file. In this way you can use mathematics, pictures and animations in your tooltips. The resulting PDF file can be used also with free Adobe Reader.

The tooltips are activated by clicking the active area on the screen and deactivated after closing page or by moving mouse outside the link. You can try the links [here](#) (Einstein's formula) and also [here](#) (animation – numbers from 1 to 6). You have to use the free Adobe Reader or nonfree Adobe Acrobat to see the effect (`xpdf`, `evince` and others fail to work with JavaScripts). For more examples how the presentation may look like see the `example.pdf` and `example-min.pdf` files in the `examples` subdirectory.

The buttons are created using `eforms.sty` which is a part of AcroTeX bundle.

2 Usage

2.1 The file with tooltips

The file with tooltips is an ordinary pdf file, one tooltip per page, tooltips should be in the top right corner at the page, in a colored box and the rest of the page should be transparent. If you consider to use `movetips` option (see below), then every page should have the dimensions equal to the dimensions of the colored box with tooltip³. We also provide simple cross referencing mechanism to refer to the tooltips. If the pdf file is created by L^AT_EX, you can define keywords to refer to the pages using `\keytip` command. Sim-

`\keytip`

*This document corresponds to `fancytooltips` v1.3, dated 2007/05/08.

¹not free `ps2pdf`

²not free Adobe Reader.

³Look at the files `tooltipy.tex` and `tooltipy.pdf` from `examples` subdirectory for a simple example how to meet this condition under pdfL^AT_EX

ply put `\usepackage[createtips]{fancytooltips}` into preamble and write `\keytip{<foo>}` in document. This writes information about keyword `<foo>` and the pagenumber into file `fancytips.tex`.

2.2 The file with presentation – pdf \LaTeX users

In the file with presentation, the user is responsible

- input either `color` or `xcolor` package in the preamble
- \LaTeX the file two times (we write some macros into `aux` file).

This is not comfortable for the user, but everybody uses different set of packages and from this reason, this part is left to the user. (And among others, the `color` or `xcolor` package is probably inputted by the package which is used to build the presentation.)

filename option To input the tooltips from file `<foo.pdf>` call the package with `filename` option: `\usepackage[filename=foo]{fancytooltips}`.

movetips option By default, tooltip appears in the top right corner of the page (use View–PageLayout–Single Page in your Adobe Reader, please). If the option `movetips` is used, then tooltip appears close to the mouse pointer. More precisely, tooltip appears with left down corner at the mouse position, if there is enough place. If not, tooltip appears with right down corner at the mouse position. Finally, the tooltip is shifted down to fit the page, if necessary⁴.

\tooltip The user can put the tooltip into her or his presentation using the command `\tooltip{<stuff>}{<keyword-or-pagenumber>}` where `<stuff>` is the printed text in `<tooltipcolor>` color and `<keyword-or-pagenumber>` is either the pagenumber of the tooltip in the external file or the keyword defined by `\keytip` command. The printed text `<stuff>` is followed by `\TooltipExtratext` command. The default value is small blue soap in a box with zero dimensions, as you have seen in the second paragraph of this documentation. There is a package option `noextratext` which defines `\TooltipExtratext` to be empty.

\TooltipExtratext

noextratext option

\tooltipanim The user can put a series (animation) of tooltips into the presentation by using `\tooltipanim{<stuff>}{<start>}{<end>}` command, where `<start>` and `<end>` are keywords defined by `\keytip` command or page numbers. The delay between two frames is `\delayinterval` milliseconds. The default value is 200, you can change it by command `\def\delayinterval{100}`.

\delayinterval

The file `example.tex` from `exmaples` subdirectory shows, how to redefine these macros to gain different behavior, see the demo file `example.pdf`.

2.3 Changes for dvips users

pages option dvips users have to specify option `dvips` in `fancytips` package. They have to use also a `pages` option with the number of pages in the PDF file with tooltips. You

⁴This option works in this way if every page of the file with tooltips has dimensions of the box with tooltip. See the `examples` subdirectory.

have to call the package by something like this:

```
\usepackage[dvips,filename=tooltips,pages=27]{fancytooltips}
```

You have to `latex` (two times) and `dvips` your file first. This produces `filename.ps` and `Tooltipsdljs.fdf` files. Distill the pdf file into `filename.pdf` and open this file by Adobe Acrobat - this imports macros from `Tooltipsdljs.fdf` file. In Acrobat's JavaScript console (`Ctrl+J`) run (`Ctrl+Enter`) the command `ImportTooltips()`; which is defined for the document and it creates invisible buttons on the first page, imports icons (the file with icons specified as `<filename>` parameter when loading `fancytooltips` must be in working directory) and returns 1. Then save the file under another name.

3 Known problems

The package works only with the last `eforms.sty`, version 2006/10/03 v1.0a. You can download this version from www.arotex.net site. The version on CTAN and in MikTeX repositories is old and this package does not work with this old version.

4 Implementation

```
1 <*package>
2 \RequirePackage{everyshi}
3 \RequirePackage{graphicx}
4 \RequirePackage{xkeyval}
5 \RequirePackage{eso-pic}
6
7 \newif\ifcreatetips\createtipsfalse
8 \DeclareOptionX{createtips}{\createtipstrue}
9
10 \newif\ifTooltip@usepdftex\Tooltip@usepdftexttrue
11 \DeclareOptionX{dvips}{\Tooltip@usepdftextfalse}
12
13 \newif\ifextratext\extratexttrue
14 \DeclareOptionX{noextratext}{\extratextfalse}
15
16 \newif\ifmovetips\movetipsfalse
17 \DeclareOptionX{movetips}{\movetipstrue}
18
19 \DeclareOptionX{filename}{\xdef\TooltipFilename{#1}}
20 \DeclareOptionX{pages}{\xdef\TooltipPages{#1}}
21
22 \ProcessOptionsX
23
24 \ifx\TooltipFilename\undefined
25 \PackageWarning{fancytooltips}{** The filename with tooltips is not given. **}
26 \fi
27
28 \ifTooltip@usepdftex
```

```

29 \RequirePackage[pdftex]{eforms}
30 \def\TooltipExtratext{\hbox to 0 pt{\smash
31   {\raisebox{0.5em}{\includegraphics[width=0.7em]{%
32     {fancytipmark.pdf}}}\hss}}
33 \else
34 \RequirePackage[dvips]{eforms}
35 \def\TooltipExtratext{\hbox to 0 pt{\smash
36   {\raisebox{0.5em}{\includegraphics[width=0.7em]{%
37     {fancytipmark.eps}}}\hss}}
38 \fi%\ifTooltip@usepdftex
39 \ifextratext\else\let\TooltipExtratext\relax\fi
40
41 \ifcreatetips

```

This part (three lines) is processed if the option `createtips` is used. In the opposite case we process the second part, up to the end of the package.

```

42 \newwrite\tipfile
43 \immediate\openout\tipfile fancytips.tex
44 \def\keytip#1{\write\tipfile{\string\tooltipname{#1}{\arabic{page}}}}
45 \else

```

This part is processed if the option `createtips` is not used. We define macros which put the hidden button with the name `ikona.n` in the background of the page `n`, if one of the commands `\tooltip` or `\tooltipan` has been used on this page. Javascripts defined by `\tooltip` and `\tooltipan` commands then unhide this button and show the corresponding picture.

```

46
47 \newdimen\buttontipwidth
48 \newdimen\buttontipheight
49 \AtBeginDocument{
50 \buttontipwidth=\paperwidth
51 \buttontipheight=\paperheight
52 }
53
54 \ifTooltip@usepdftex
55 \def\frametip{%
56   \pdfstartlink user{%
57     /Subtype /Widget
58     /F 6
59     /T (ikona.\thepage)
60     /FT /Btn
61     /Ff 65536
62     /H /N
63     /BS << /W 1 /S /S >>
64     /MK << /TP 1 /IF <</A[1.0 1.0]/SW /B>> >>
65   }%
66   \vbox to \buttontipheight {\vss\hbox to \buttontipwidth{\hss}\pdfendlink}
67 \else

```

For dvips users we use the macros from `eqxerquiz.sty` package.

```

68 \def\everyeqIcon#1{\def\every@eqIcon{#1}}

```

```

69 \def\every@eqIcon{
70 \newcommand\eqIconFTT[4] []
71 {%
72 \push@@Button{#1}{#2}{#3}{#4}{-}{\eq@setButtonProps\eq@Button@driver}%
73 {\eqIconDefaults\every@ButtonField\every@eqIcon}%
74 }
75 \def\eqIconDefaults
76 {%
77 \rawPDF{-}\S{-}\mkIns{/TP 1 /IF<</A[1.0 1.0]/SW/B>>}\R{0}
78 \CA{-}\RC{-}\AC{-}\BC{-}\BG{-}\H{B}
79 \textColor{0 g}\Ff{\FfReadOnly}
80 }
81 \def\frametip@{\eqIconFTT[\BC{-}\BG{-}\F{\FHidden}]}%
82 {ikona.\thepage}{\paperwidth}{\paperheight}}%
83 \fi%\ifTooltip@usepdftex
84
85 \def\frametip{%
86 \expandafter\ifx \csname TooltipPage\thepage\endcsname\relax
87 \else
88 \setbox0=\hbox{\frametip@}%
89 \hbox{\raise \dp0 \box0}
90 \fi}%
91 \AddToShipoutPicture{\hbox to 0 pt{\frametip\hss}}

```

In the macros \tooltip and \tooltipanim we print the text into box with zero dimensions and then we build a button which covers this text and has an associated JavaScript action. The important part is the \PushButton macro. You can adjust these macros or write similar macros which do what you need. For some examples see the file example.tex from the examples directory.

```

92 \definecolor{tooltipcolor}{rgb}{0,0,1}
93 \def\TooltipPage#1{\expandafter\gdef\csname TooltipPage#1\endcsname{#1}}
94 \def\tooltip#1#2{%
95 \write\@auxout{\noexpand\TooltipPage{\thepage}}%
96 \checkTipNumber{#2}\edef\TipNumber{\FindTipNumber{#2}}%
97 \leavevmode
98 \setbox0=\hbox{{\color{tooltipcolor}{#1}}}\hbox to 0 pt{{\copy0\TooltipExtratext\hss}}%
99 \pushButton[\BC{-}\BG{-}\S{-}\AA{\AAMouseExit{\JS{CloseTooltips();}}}%
100 \A{\JS{this.getField("ikona."+(this.pageNum+1)).hidden=false;
101 try {app.clearInterval(animace);}catch (e) {}
102 \ifmovetips nastav(\TipNumber);\fi
103 zobraz(\TipNumber);
104 }}]
105 {TooltipField}{\wd0}{\ht0}}
106 \def\delayinterval{200}
107 \def\tooltipanim#1#2#3{%
108 \write\@auxout{\noexpand\TooltipPage{\thepage}}%
109 \checkTipNumber{#2}\edef\TipNumberA{\FindTipNumber{#2}}%
110 \checkTipNumber{#3}\edef\TipNumberB{\FindTipNumber{#3}}%
111 \leavevmode
112 \setbox0=\hbox{{\color{tooltipcolor}{#1}}}\hbox to 0 pt{{\copy0\TooltipExtratext\hss}}%

```

```

113 \pushButton[\BC{}\BG{}\S{}\AA{\AAMouseExit{\JS{CloseTooltips();}}}]
114 \A{\JS{
115     try {app.clearInterval(animace);}catch (e) {}
116     var cislo=\TipNumberA;
117     \ifmovetips nastav(\TipNumberA);\fi
118     function animuj()
119     {
120         if (cislo<\TipNumberB) cislo=cislo+1;
121         this.getField('ikona.'+(this.pageNum+1)).buttonSetIcon(this.getField("animtiph."+cislo)
122     };
123     this.getField('ikona.'+(this.pageNum+1)).buttonSetIcon(this.getField("animtiph."+TipNumb
124     this.getField("ikona."+(this.pageNum+1)).hidden=false;
125     animace=app.setInterval('animuj();', \delayinterval);
126     }}
127 ]{TooltipField}{\wd0}{\ht0}}

```

This code closes tooltip if the page is closed.

```

128 \ifTooltip@usepdfTeX
129 \def\TooltipPageopencloseJS{ \global\pdfpageattr{%
130     /AA << /O << /S /JavaScript /JS (CloseTooltips();) >> >>}%
131 }
132 \pdfximage{\TooltipFilename.pdf}%
133 \edef\TooltipPages{the\pdflastximagepages}%
134 \else
135 \def\TooltipPageopencloseJS{
136 \literalps@out{%
137     [ {ThisPage} << /AA <<
138     /O << /S /JavaScript /JS (CloseTooltips();) >>
139     >> >> /PUT pdfmark}}
140 \OpenAction{/S /JavaScript /JS (CloseTooltips();)}
141 \fi%\ifTooltip@usepdfTeX
142 \EveryShipout{\TooltipPageopencloseJS}%
143
144 \ifTooltip@usepdfTeX
145 \begin{insDLJS}[fancyTooltipsLoaded]{Tooltipsdljs}{DLJS for Tooltips}
146     var animace;
147     var fancyTooltipsLoaded = true;
148
149     function CloseTooltips()
150     {
151         try {this.getField("ikona").hidden=true;}catch (e) {}
152         try {app.clearInterval(animace);}catch (e) {}
153     }
154
155     function nastav(cislo)
156     {
157         var f=this.getField("ikona."+(this.pageNum+1));
158         var g=this.getField("animtiph."+cislo);
159         var sourf=f.rect;
160         var sourg=g.rect;

```

```

161     if ((mouseX+sourg[2]-sourf[0])<sourf[2])
162     var percX=100*(mouseX-sourf[0])/((sourf[2]-sourf[0])-(sourg[2]-sourg[0]));
163     else
164     var percX=100*(mouseX-sourf[0]-(sourg[2]-sourg[0]))/((sourf[2]-sourf[0])-(sourg[2]-sourg[0]));
165     var percY=100*(mouseY-sourf[3])/((sourf[1]-sourf[3])-(sourg[1]-sourg[3]));
166     if (percX>100) percX=100;
167     if (percY>100) percY=100;
168     if (percX<0) percX=0;
169     if (percY<0) percY=0;
170     f.buttonAlignX=percX;
171     f.buttonAlignY=percY;
172 }
173
174 function zobraz(cislo)
175 {
176     var f=this.getField("ikona."+this.pageNum+1));
177     var g=this.getField("animtiph."+cislo);
178     f.hidden=false;
179     f.buttonSetIcon(g.buttonGetIcon());
180 }
181 \end{insDLJS}
182 \else
183 \begin{insDLJS}[fancyTooltipsLoaded]{Tooltipsdljs}{DLJS for Tooltips}
184     var animace;
185     var fancyTooltipsLoaded = true;
186
187     function CloseTooltips()
188     {
189         try {this.getField("ikona").hidden=true;}catch (e) {}
190         try {app.clearInterval(animace);}catch (e) {}
191     }
192
193     function ImportTooltips()
194     {
195         console.println("importing pictures");
196         for (var i=1;i<=\TooltipPages;i++)
197         {
198             this.insertPages(this.numPages-1,"\TooltipFilename.pdf",(i-1),(i-1));
199             var rozm=this.getPageBox("Crop",this.numPages-1);
200             this.deletePages(this.numPages-1);
201             var p=this.addField("animtiph."+i,"button",0,rozm);
202             p.buttonPosition=position.iconOnly;
203             p.hidden=true;
204             this.getField("animtiph."+i).buttonImportIcon("\TooltipFilename.pdf",(i-1));
205         }
206         console.println("imported \TooltipPages pictures");
207         return(1);
208     }
209
210     function nastav(cislo)

```

```

211 {
212   var f=this.getField("ikona."+this.pageNum+1));
213   var g=this.getField("animtiph."+cislo);
214   var sourf=f.rect;
215   var sourg=g.rect;
216   if ((mouseX+sourg[2]-sourf[0])<sourf[2])
217   var percX=100*(mouseX-sourf[0])/((sourf[2]-sourf[0])-(sourg[2]-sourf[0]));
218   else
219   var percX=100*(mouseX-sourf[0]-(sourg[2]-sourf[0]))/((sourf[2]-sourf[0])-(sourg[2]-sourf[0]));
220   var percY=100*(mouseY-sourf[3])/((sourf[1]-sourf[3])-(sourg[1]-sourf[3]));
221   if (percX>100) percX=100;
222   if (percY>100) percY=100;
223   if (percX<0) percX=0;
224   if (percY<0) percY=0;
225   f.buttonAlignX=percX;
226   f.buttonAlignY=percY;
227 }
228
229 function zobraz(cislo)
230 {
231   var f=this.getField("ikona."+this.pageNum+1));
232   var g=this.getField("animtiph."+cislo);
233   f.hidden=false;
234   f.buttonSetIcon(g.buttonGetIcon());
235 }
236 \end{insDLJS}
237 \fi

```

A cycle is used to create hidden buttons. Each button has associated a page from the file with tooltips as icon. These icons are invoked by JavaScripts defined in `\tooltip` and `\tooltipanim` macros.

```

238 \newcount\tooltip@count
239 \ifTooltip@usepdfTeX
240 \newcommand*{\TooltipHidden}{%
241   \count@=0
242   \@whilenum\count@<\TooltipPages \do{%
243     \tooltip@count=\count@
244     \advance \tooltip@count by 1%
245     \bgroup
246     \immediate\pdfximage
247     page \the\tooltip@count{\TooltipFilename.pdf}%
248     \mbox{\leavevmode
249       \vbox to 0 pt{\vss\hbox to 0 pt{\pdfstartlink user{%
250         /Subtype /Widget
251         /F 6
252         /T (animtiph.\the\tooltip@count)
253         /FT /Btn
254         /Ff 65536
255         /H /N
256         /BS << /W 1 /S /S >>

```



```

257      /MK <<
258      /TP 1
259      /I \the\pdflastximage\space 0 R
260      /IF << /SW /A >>
261      >>
262      }%
263      \phantom{\pdfrefximage \pdflastximage}%
264      \pdfendlink\hss}}}%
265      \egroup
266      \advance\count@\@ne}%
267 }
268 \AddToShipoutPicture*{\hbox to 0 pt{\TooltipHidden}}
269 \else
270 \let\TooltipHidden\relax
271 \fi

The keywords for the tooltips can be stored in the file fancytips.tex. The topics
in this file are created by \keytip macro (see the first part of the code).
272 \AtBeginDocument{\IfFileExists{fancytips.tex}{\input{fancytips.tex}}
273 \PackageInfo{fancytooltips}{Inputting fancytips.tex.}}%
274 {\PackageWarning{fancytooltips}{No file fancytips.tex!
275     Your keywords for tooltips will not work!}}
276
277 \def\tooltipname#1#2{\expandafter\xdef\csname FancyToolTip@#1\endcsname{#2}}
278
279 \def\FindTipNumber#1{\expandafter\ifx \csname FancyToolTip@#1\endcsname\relax
280     #1\else\csname FancyToolTip@#1\endcsname\fi}
281
282 \def\checkTipNumber#1{\expandafter\ifx
283     \csname FancyToolTip@#1\endcsname\relax \PackageWarning{fancytooltips}{No
284     framenumbers assigned to keyword #1. I assume that #1 is the
285     number of the frame.}%
286     \fi}
287
288 \fi
289 \end{package}

```