The fancytooltips package*

Robert Marik
marik@mendelu.cz

March 12, 2007

1 Introduction

The package `fancytooltips` is a package for LaTeX. The pdf can be created by pdflatex or by latex + dvips + AdobeDistiller[1] + Adobe Acrobat[2]. 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 Abobe 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 tooltip[3]. We also provide simple cross referencing mechanism to reffer to the tooltips. If the pdf file is created by LaTeX, you can define keywords to reffer to the pages using `\keytip` command. Sim-

`\keytip`

*This document corresponds to fancytooltips v1.2, dated 2007/03/01.

[1] not free ps2pdf

[2] not free Adobe Reader.

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

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 – pdfLaTeXusers

In the file with presentation, the user is responsible

- to load package `eforms` with `pdftex` option before loading fancytooltips package,
- 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 preciselly, 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 nesessary[4].

`\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

`\TooltipExtratext` 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`

`noextratext option` which defines `\TooltipExtratext` to be empty.

`\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

`\delayinterval` 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}`.

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

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

2.3 Changes for dvips users

pages option

dvips users have to specify option `dvips` in both `fancytips` and `eforms` packages. They have to use also a `pages` option with the number of pages in the PDF file with tooltips. You have to call the package by something like this:

```
\usepackage[dvips]{eforms}
\usepackage[dvips,filename=tooltipy,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 (`Crtl+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 Implementation

```
⟨*package⟩
\RequirePackage{everyshi}
\RequirePackage{graphicx}
\RequirePackage{xkeyval}
\RequirePackage{eso-pic}

\newif\ifcreatetips\createtipsfalse
\DeclareOptionX{createtips}{\createtipstrue}

\newif\ifTooltip@usepdftex\Tooltip@usepdftextrue
\DeclareOptionX{dvips}{\Tooltip@usepdftexfalse}

\newif\ifextratext\extratexttrue
\DeclareOptionX{noextratext}{\extratextfalse}

\newif\ifmovetips\movetipsfalse
\DeclareOptionX{movetips}{\movetipstrue}

\DeclareOptionX{filename}{\xdef\TooltipFilename{#1}}
\DeclareOptionX{pages}{\xdef\TooltipPages{#1}}

\ProcessOptionsX

\ifx\TooltipFilename\undefined
\PackageWarning{fancytooltips}{** The filename with tooltips is not given. **}
\fi

\ifTooltip@usepdftex
\def\TooltipExtratext{\hbox to 0 pt{\smash
    {\raisebox{0.5em}{\includegraphics[width=0.7em]%
```

```
        {fancytipmark.pdf}}}\hss}}
\else
\def\TooltipExtratext{\hbox to 0 pt{\smash
    {\raisebox{0.5em}{\includegraphics[width=0.7em]%
        {fancytipmark.eps}}}\hss}}
\fi%\ifTooltip@usepdftex
\ifextratext\else\let\TooltipExtratext\relax\fi

\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.

```
\newwrite\tipfile
\immediate\openout\tipfile fancytips.tex
\def\keytip#1{\write\tipfile{\string\tooltipname{#1}{\arabic{page}}}}
\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 backgoud of the page `n`, if one of the commands `\tooltip` or `\tooltipanim` has been used on this page. Javascripts defined by `\tooltip` and `\tooltipanim` commands then unhide this button and show the corresponding picture.

```

\newdimen\buttontipwidth
\newdimen\buttontipheight
\AtBeginDocument{
\buttontipwidth=\paperwidth
\buttontipheight=\paperheight
}

\ifTooltip@usepdftex
\def\frametip@{%
  \pdfstartlink user{%
    /Subtype /Widget
    /F 6
    /T (ikona.\thepage)
    /FT /Btn
    /Ff 65536
    /H /N
    /BS << /W 1 /S /S >>
    /MK << /TP 1 /IF <</A[1.0 1.0]/SW /B>>  >>
  }%
  \vbox to \buttontipheight {\vss\hbox to \buttontipwidth{\hss}}\pdfendlink}
\else
```

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

```
\def\everyeqIcon#1{\def\every@eqIcon{#1}}
\def\every@eqIcon{}
\newcommand\eqIconFTT[4][]
{%
  \push@@Button{#1}{#2}{#3}{#4}{}{\eq@setButtonProps\eq@Button@driver}%
```

```
  {\eqIconDefaults\every@ButtonField\every@eqIcon}%
}
\def\eqIconDefaults
{%
  \rawPDF{}\S{}\mkIns{/TP 1 /IF<</A[1.0 1.0]/SW/B>>}\R{0}
  \CA{}\RC{}\AC{}\BC{}\BG{}\H{B}
  \textColor{0 g}\Ff{\FfReadOnly}
}
\def\frametip@{\eqIconFTT[\BC{}\BG{}\F{\FHidden}]%
  {ikona.\thepage}{\paperwidth}{\paperheight}}%
\fi%\ifTooltip@usepdftex

\def\frametip{%
  \expandafter\ifx \csname TooltipPage\thepage\endcsname\relax
  \else
  \setbox0=\hbox{\frametip@}%
  \hbox{\raise \dp0 \box0}
  \fi}%
\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 exmaples see the file `example.tex` from the examples directory.

```
\definecolor{tooltipcolor}{rgb}{0,0,1}
\def\TooltipPage#1{\expandafter\gdef\csname TooltipPage#1\endcsname{#1}}
\def\tooltip#1#2{%
  \write\@auxout{\noexpand\TooltipPage{\thepage}}%
  \checkTipNumber{#2}\edef\TipNumber{\FindTipNumber{#2}}%
  \leavevmode
  \setbox0=\hbox{{\color{tooltipcolor}{#1}}}\hbox to 0 pt{{\copy0\TooltipExtratext\hss}}%
  \pushButton[\BC{}\BG{}\S{}\AA{\AAMouseExit{\JS{CloseTooltips();}}}
  \A{\JS{this.getField("ikona."+(this.pageNum+1)).hidden=false;
      try {app.clearInterval(animace);}catch (e) {}
      \ifmovetips nastav(\TipNumber);\fi
      zobraz(\TipNumber);
    }}]
  {TooltipField}{\wd0}{\ht0}}
\def\delayinterval{200}
\def\tooltipanim#1#2#3{%
  \write\@auxout{\noexpand\TooltipPage{\thepage}}%
  \checkTipNumber{#2}\edef\TipNumberA{\FindTipNumber{#2}}%
  \checkTipNumber{#3}\edef\TipNumberB{\FindTipNumber{#3}}%
  \leavevmode
  \setbox0=\hbox{{\color{tooltipcolor}{#1}}}\hbox to 0 pt{{\copy0\TooltipExtratext\hss}}%
  \pushButton[\BC{}\BG{}\S{}\AA{\AAMouseExit{\JS{CloseTooltips();}}}
  \A{\JS{
      try {app.clearInterval(animace);}catch (e) {}
      var cislo=\TipNumberA;
```

```
      \ifmovetips nastav(\TipNumberA);\fi
      function animuj()
      {
        if (cislo<\TipNumberB) cislo=cislo+1;
        this.getField('ikona.'+(this.pageNum+1)).buttonSetIcon(this.getField("animtiph."+cislo)
      };
      this.getField('ikona.'+(this.pageNum+1)).buttonSetIcon(this.getField("animtiph."+\TipNumb
      this.getField("ikona."+(this.pageNum+1)).hidden=false;
      animace=app.setInterval('animuj();', \delayinterval);
    }}
  ]{TooltipField}{\wd0}{\ht0}}
```

This code closes tooltip if the page is closed.

```
\ifTooltip@usepdftex
\def\TooltipPageopencloseJS{  \global\pdfpageattr{%
    /AA << /O << /S /JavaScript /JS (CloseTooltips();) >>  >>}%
}
\pdfximage{\TooltipFilename.pdf}%
\edef\TooltipPages{\the\pdflastximagepages}%
\else
\def\TooltipPageopencloseJS{
\literalps@out{%
     [ {ThisPage} << /AA <<
    /O << /S /JavaScript /JS (CloseTooltips();) >>
    >> >> /PUT pdfmark}}
\OpenAction{/S /JavaScript /JS (CloseTooltips();)}
\fi%\ifTooltip@usepdftex
\EveryShipout{\TooltipPageopencloseJS}%

\ifTooltip@usepdftex
\begin{insDLJS}[clearTooltips]{Tooltipsdljs}{My Private DLJS for Tooltips}
  var animace;
  function CloseTooltips()
  {
    try {this.getField("ikona").hidden=true;}catch (e) {}
    try {app.clearInterval(animace);}catch (e) {}
  }

  function nastav(cislo)
  {
    var f=this.getField("ikona."+(this.pageNum+1));
    var g=this.getField("animtiph."+cislo);
    var sourf=f.rect;
    var sourg=g.rect;
    if ((mouseX+sourg[2]-sourg[0])<sourf[2])
    var percX=100*(mouseX-sourf[0])/((sourf[2]-sourf[0])-(sourg[2]-sourg[0]));
    else
    var percX=100*(mouseX-sourf[0]-(sourg[2]-sourg[0]))/((sourf[2]-sourf[0])-(sourg[2]-sourg[0]
    var percY=100*(mouseY-sourf[3])/((sourf[1]-sourf[3])-(sourg[1]-sourg[3]));
    if (percX>100) percX=100;
```

```
    if (percY>100) percY=100;
    if (percX<0) percX=0;
    if (percY<0) percY=0;
    f.buttonAlignX=percX;
    f.buttonAlignY=percY;
  }

  function zobraz(cislo)
  {
    var f=this.getField("ikona."+(this.pageNum+1));
    var g=this.getField("animtiph."+cislo);
    f.hidden=false;
    f.buttonSetIcon(g.buttonGetIcon());
  }
\end{insDLJS}
\else
\begin{insDLJS}[clearTooltips]{Tooltipsdljs}{My Private DLJS for Tooltips}
  var animace;
  function CloseTooltips()
  {
    try {this.getField("ikona").hidden=true;}catch (e) {}
    try {app.clearInterval(animace);}catch (e) {}
  }

  function ImportTooltips()
  {
    console.println("importing pictures");
    for (var i=1;i<=\TooltipPages;i++)
    {
      this.insertPages(this.numPages-1,"\TooltipFilename.pdf",(i-1),(i-1));
      var rozm=this.getPageBox("Crop",this.numPages-1);
      this.deletePages(this.numPages-1);
      var p=this.addField("animtiph."+i,"button",0,rozm);
      p.buttonPosition=position.iconOnly;
      p.hidden=true;
      this.getField("animtiph."+i).buttonImportIcon("\TooltipFilename.pdf",(i-1));
    }
    console.println("imported \TooltipPages pictures");
    return(1);
  }

  function nastav(cislo)
  {
    var f=this.getField("ikona."+(this.pageNum+1));
    var g=this.getField("animtiph."+cislo);
    var sourf=f.rect;
    var sourg=g.rect;
    if ((mouseX+sourg[2]-sourg[0])<sourf[2])
    var percX=100*(mouseX-sourf[0])/((sourf[2]-sourf[0])-(sourg[2]-sourg[0]));
    else
```

```
    var percX=100*(mouseX-sourf[0]-(sourg[2]-sourg[0]))/((sourf[2]-sourf[0])-(sourg[2]-sourg[0]
    var percY=100*(mouseY-sourf[3])/((sourf[1]-sourf[3])-(sourg[1]-sourg[3]));
    if (percX>100) percX=100;
    if (percY>100) percY=100;
    if (percX<0) percX=0;
    if (percY<0) percY=0;
    f.buttonAlignX=percX;
    f.buttonAlignY=percY;
  }

  function zobraz(cislo)
  {
    var f=this.getField("ikona."+(this.pageNum+1));
    var g=this.getField("animtiph."+cislo);
    f.hidden=false;
    f.buttonSetIcon(g.buttonGetIcon());
  }
\end{insDLJS}
\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.

```
\newcount\tooltip@count
\ifTooltip@usepdftex
\newcommand*{\TooltipHidden}{%
  \count@=0
  \@whilenum\count@<\TooltipPages \do{%
    \tooltip@count=\count@
    \advance \tooltip@count by 1%
    \bgroup
    \immediate\pdfximage
    page \the\tooltip@count{\TooltipFilename.pdf}%
    \mbox{\leavevmode
      \vbox to 0 pt{\vss\hbox to 0 pt{\pdfstartlink user{%
        /Subtype /Widget
        /F 6
        /T (animtiph.\the\tooltip@count)
        /FT /Btn
        /Ff 65536
        /H /N
        /BS << /W 1 /S /S >>
        /MK <<
        /TP 1
        /I \the\pdflastximage\space 0 R
        /IF << /SW /A >>
        >>
      }%
      \phantom{\pdfrefximage \pdflastximage}%
      \pdfendlink\hss}}}%
```

```
    \egroup
    \advance\count@\@ne}%
}
\AddToShipoutPicture*{\hbox to 0 pt{\TooltipHidden}}
\else
\let\TooltipHidden\relax
\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).

```
\AtBeginDocument{\IfFileExists{fancytips.tex}{\input{fancytips.tex}
\PackageInfo{fancytooltips}{Inputting fancytips.tex.}}%
  {\PackageWarning{fancytooltips}{No file fancytips.tex!
      Your keywords for tooltips will not work!}}}

\def\tooltipname#1#2{\expandafter\xdef\csname FancyToolTip@#1\endcsname{#2}}

\def\FindTipNumber#1{\expandafter\ifx \csname FancyToolTip@#1\endcsname\relax
  #1\else\csname FancyToolTip@#1\endcsname\fi}

\def\checkTipNumber#1{\expandafter\ifx
  \csname FancyToolTip@#1\endcsname\relax \PackageWarning{fancytooltips}{No
    framenumber is assigned to keyword #1. I assume that #1 is the
    number of the frame.}%
  \fi}

\fi
⟨/package⟩
```