The fancytooltips package*

Robert Marik
marik@mendelu.cz

February 19, 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 also mathematics, pictures and animations in your tooltips. The resulting PDF file can be used also with free Abobe Reader.

The tooltips are activated by `MouseOver` action with pressed `Shift` button down or by `MouseDown` action. The tooltips are 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). Tooltip appears on the right boundary of the first paragraph. 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[3]. We also provide simple cross referencing mechanism to reffer to the tooltips. If the pdf file is created by LaTeX,

*This document corresponds to fancytooltips v1.1, dated 2007/02/20.

[1] not free ps2pdf

[2] not free Adobe Reader.

[3] See the example file `tooltipy.tex` and `tooltipy.pdf` which make use of `geometry.sty` package to set the page dimensions and `\colorbox` and `minipage` to make the box with tooltip

\keytip you can define keywords to reffer to the pages using \keytip command. Simply 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 include package eforms with pdftex option before loading fancytooltips package, and
- input either color or xcolor package in the preamble.

This is not convenient for the user, but everybody uses different package 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.)

To input the tooltips from file ⟨*foo.pdf*⟩ call the package with ⟨*filename*⟩ option \usepackage[filename=foo]{fancytooltips}, input images for tooltips somewhere in the document using macro \TooltipHidden and put the macro \frametip on every page[4] in the right top corner of your document[5]. The dimensions of button with tooltips (created by \frametip macro) are stored in the \buttontipwidth and \buttontipheight length variables. If the page with tooltip is bigger, it is resized to fit into the button. You can change it for example by \buttontipwidth=4in. To get expected results (tooltips in the top right corner) these lengths should be equal or constant multiples of the page dimensions of the file with tooltips. The button is inside a box with zero dimensions and hence the box produced by \frametip command has zero dimensions.

\TooltipHidden \frametip \buttontipwidth \buttontipheight

\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

\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

[4] In the current document the \frametip command was used only once in the first paragraph, since we have tooltips in this paragraph only and the tooltips in the top right corner of the page would be outside of the screen in most cases.

[5] See the file example.tex and example-min.tex to know how to achieve this with the pdfscrren.sty presentation bundle and pure article.cls class, respectively.

2.3 Changes for dvips users

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` and `dvips` your file. 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 `ImportIcons();` which is defined for the document and it imports the icons and returns 1. The file with pdf icon must be in the working directory. Then save the file under another name.

3 Implementation

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

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

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

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

\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=1em]%
        {fancytooltipsmark.pdf}}}\hss}}
\else
\def\TooltipExtratext{\hbox to 0 pt{\smash
    {\raisebox{0.5em}{\includegraphics[width=1em]%
        {fancytooltipsmark.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
```

If dvips is used, we define icons by the command `\eqIcon` stolen from `exerquiz.sty`. The following definitions are verbatim copy from `exerquiz.sty`. The definition of `\eqIconDefaults` contains some customizations.

```
\ifTooltip@usepdftex\else
\ifx\everyeqIcon\undefined
\newcommand\everyeqIcon[1]{\def\every@eqIcon{#1}}
\def\every@eqIcon{}
\newcommand\eqIcon[4][]
{%
  \push@@Button{#1}{#2}{#3}{#4}{}{\eq@setButtonProps\eq@Button@driver}%
  {\eqIconDefaults\every@ButtonField\every@eqIcon}%
}
\fi%\ifx\everyeqIcon\undefined
\def\eqIconDefaults
{%
  \rawPDF{}\S{}\mkIns{/TP 1 /IF<</A[1.0 1.0]/SW/B>>}\R{270}
  \CA{}\RC{}\AC{}\BC{}\BG{}\H{B}
  \textColor{0 g}\Ff{\FfReadOnly}
}
\fi%\ifTooltip@usepdftex
```

Macro `\frametip` creates button in which tooltips appear. You have to use macro `\frametip` on every page in your presentation in right top corner (see the directory `examples`).

```
\def\frametip{\vbox to 0 pt{\hbox to 0 pt{\hss\buttontip}\vss}}
\newdimen\buttontipwidth \buttontipwidth=5in
\newdimen\buttontipheight \buttontipheight=5in
\ifTooltip@usepdftex
\def\buttontip{%
\pdfstartlink user{%
  /Subtype /Widget
  /F 6
  /T (ikona)
  /FT /Btn
  /Ff 65536
  /H /N
  /BS << /W 1 /S /S >>
  /MK << /TP 1 /IF <</SW /B>>  >>
}%
```

```
\vbox to \buttontipheight {\vss\hbox to \buttontipwidth{\hss}}\pdfendlink}
\else
\def\buttontip{%
   \eqIcon[\BC{}\BG{}\F{\FHidden}]{ikona}{\buttontipwidth}{\buttontipheight}
 }
\fi%\ifTooltip@usepdftex
```

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.

```
\definecolor{tooltipcolor}{rgb}{0,0,1}
\def\tooltip#1#2{\checkTipNumber{#2}%
  \edef\TipNumber{\FindTipNumber{#2}}%
  \setbox0=\hbox{{\color{tooltipcolor}{#1}}}\hbox to 0 pt{{\copy0\TooltipExtratext\hss}}%
  \pushButton[\BC{}\BG{}\S{}\AA{
    \AAMouseExit{\JS{CloseTooltips();}}
    \AAMouseEnter{\JS{
        try {app.clearInterval(animace);}catch (e) {}
        if (event.shift) {this.getField("ikona").hidden=false;
          this.getField('ikona').buttonSetIcon(this.getField("animtiph\TipNumber").buttonGetIco
  }
  \A{\JS{this.getField("ikona").hidden=false;
      try {app.clearInterval(animace);}catch (e) {}
      this.getField('ikona').buttonSetIcon(this.getField("animtiph\TipNumber").buttonGetIcon())
  {TooltipField}{\wd0}{\ht0}}
\def\delayinterval{200}
\def\tooltipanim#1#2#3{%
  \checkTipNumber{#2}\edef\TipNumberA{\FindTipNumber{#2}}%
  \checkTipNumber{#3}\edef\TipNumberB{\FindTipNumber{#3}}%
  \setbox0=\hbox{{\color{tooltipcolor}{#1}}}\hbox to 0 pt{{\copy0\TooltipExtratext\hss}}%
  \pushButton[\BC{}\BG{}\S{}\AA{
    \AAMouseExit{\JS{CloseTooltips();}}
    \AAMouseEnter{\JS{
        var cislo=\TipNumberA;
        try {app.clearInterval(animace);}catch (e) {}
        function animuj()
        {
          if (cislo<\TipNumberB) cislo=cislo+1;
          this.getField('ikona').buttonSetIcon(this.getField("animtiph"+cislo).buttonGetIcon())
        };
        if (event.shift) {
          this.getField('ikona').buttonSetIcon(this.getField("animtiph"+\TipNumberA).buttonGetI
          this.getField("ikona").hidden=false;
          animace=app.setInterval('animuj();', \delayinterval);
        }
      }}
  }
  \A{\JS{
      try {app.clearInterval(animace);}catch (e) {}
      var cislo=\TipNumberA;
```

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

This code for pdftex 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()
  {
    this.getField("ikona").hidden=true;
    try {app.clearInterval(animace);}catch (e) {}
  }
\end{insDLJS}
\else
\begin{insDLJS}[clearTooltips]{Tooltipsdljs}{My Private DLJS for Tooltips}
  var animace;
  function CloseTooltips()
  {
    this.getField("ikona").hidden=true;
    try {app.clearInterval(animace);}catch (e) {}
  }
  function ImportTooltips()
  {
    for (var i=1;i<=\TooltipPages;i++)
    {this.getField("animtiph"+i).buttonImportIcon("\TooltipFilename.pdf",(i-1));}
    return(1);
```

```
  }
\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
    width 0 pt height 0 pt depth 0 pt
    page \the\tooltip@count{\TooltipFilename.pdf}%
    \mbox{\leavevmode
      \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 >>
        >>
      }%
      \pdfendlink}%
    \egroup
    \advance\count@\@ne}%
}
\else
\def\TooltipHidden{%
  \count@=0
  \@whilenum\count@<\TooltipPages \do{%
    \tooltip@count=\count@
    \advance \tooltip@count by 1
    \bgroup
    \eqIcon[\BC{}\BG{}\F{\FHidden}]
    {animtiph\the\tooltip@count}{0cm}{0cm}%
    \egroup
    \advance\count@\@ne}%
}
\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⟩