The phflplx package — include PDF graphics with hyperlinks

Philippe Faist *philippe.faist@bluewin.ch*

April 09, 2021

phflplx—A handy LaTeX package for including graphics defined via LaTeX source code files. Designed for use with the `ltxpdflinks` tool in order to include PDF graphics in documents with external or internal hyperlinks.

1 Introduction

This package is designed to be used in conjunction with the `ltxpdflinks` command-line utility to extract PDF links into `.lplx` files.

Usage:

1. Run `ltxpdflinks` on your PDF files before you compile your document:

   ```
   > ltxpdflinks myfigure.pdf
   ```

2. Use the following commands in your document preamble:

   ```
   \usepackage{phflplx}
   \DeclareGraphicsExtensions{.lplx,.pdf}
   ```

3. Drop a copy of `phflplx.sty` in the same folder as your document in case you don’t have it installed user-wide or system-wide.

4. Enjoy!

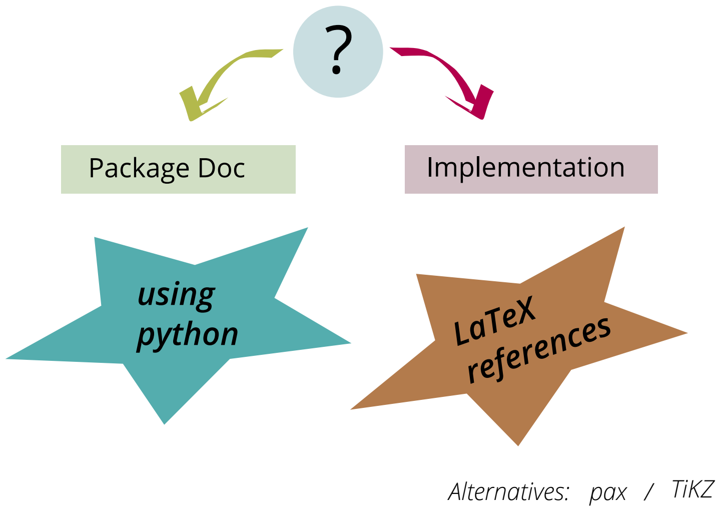

Figure 1: A diagram with clickable PDF hyperlinks, including internal links to other parts of the document.

In fact, LPLX files are simple LaTeX sources that are included directly by the graphics driver. The LPLX file typically includes the PDF itself using a lower-level graphics command and draws invisible hyperlinks on top of the graphics. (The LPLX file also takes care of resizing and cropping the graphics if necessary.)

Figure 1 provides a glimpse of what you can do with `ltxpdflinks` and phfIplx.

2 Using the `ltxpdflinks` script

Use the `ltxpdflinks` python script (`pip install ltxpdflinks`) to process your PDF files and generate an accompanying LPLX file. The LPLX file is then picked up and links are rendered accordingly.

The `ltxpdflinks` script will understand the following type of links:

- `https://example.com/` — Regular URLs generate a web link to the given location, as expected.
- `latexref://ref/<anchor>` and `latexref://cite/<citekey>` — Special URLs with the fictitious `latexref://` scheme are treated as internal LaTeX references and citations (`ref` or `cite` fictitious domain). The `<anchor>` and `<citekey>` are arguments that you would provide to `\ref` or `\cite`, respectively.
- `latexbox://<boxdomain>/<boxname>`, optionally `latexbox://<boxdomain>/<boxname>?<key1>=<value1>&<key2>=<value2>...` — Special URLs with the fictitious `latexbox://` protocol are treated

as an instruction to look up data provided from the LaTeX document and include it in the figure. You can use this to include math, custom references, or any LaTeX content you'd like to add in the figure.

The `<boxdomain>` and `<boxname>` refer to a named "box" defined using the `\phflplxDefineBox` command (below).

You can specify the additional arguments as standard URL query string arguments:

– `valign=...` How to align the content vertically. One of `t` (top), `c` (center), or `b` (bottom).
– `halign=...` How to align and typeset the content horizontally. One of `l` (left), `c` (center), `r` (right), `p` (just typeset the paragraph normally w/o additional alignment commands), `ml` (left on one line, using a `\makebox`), `mc` (center on one line, using a `\makebox`), `mr` (right on one line, using a `\makebox`), or `ms` (spread on one line, using a `\makebox`).
– `stylemacro=macroname` Wrap the contents within a call to the given styling macro (name without the backslash). Use for instance `stylemacro=textbf` to typeset the content in boldface font.

`\phflplxDefineBox` To provide content from the LaTeX document that the figure can include via a `latexbox://`-type url, use the `\phflplxDefineBox` macro:

`\phflplxDefineBox{`⟨*boxdomain*⟩`}{`⟨*boxname*⟩`}{`⟨*content*⟩`}`

The ⟨*boxdomain*⟩ and ⟨*boxname*⟩ are your user-defined keys. We suggest having `boxdomain` matching the name of the figure and `boxname` to be a meaningful name describing the content you're saving. The ⟨*content*⟩ can be arbitrary content that will be placed where the figure requested the given box.

■ 3 Implementation

Load some useful packages.

```
\RequirePackage{etoolbox}
```

Check that the user is loading the hyperref package! (We don't load it automatically to avoid issues of package loading order, link appearance, etc.)

```
\AtBeginDocument{%
  \@ifpackageloaded{hyperref}{}{%
    \PackageWarning{phflplx}{The package ‘hyperref’ was not loaded. You
      probably forgot to load it.}%
  }%
}
```

3.1 Integration into graphics/graphicx package

Declaring the graphics rule for the LPLX extension. Note that the LPLX file has the line `%%BoundingBox 0 0 <W> <H>` so that the size of the graphics can be parsed by the default graphics graphic size inspector, which is designed to parse `%%BoundingBox` commands in EPS files.

```
\DeclareGraphicsRule{.lplx}{lplx}{*}{}
```

Declare the driver functions for the graphics package internals.

```
\def\Ginclude@lplx#1{%
  \message{<#1>}%
  \input{#1}%
}
```

3.2 Commands called from the LPLX file

The contents of the LPLX file is wrapped around the `\LPLX` macro. The first argument is a dictionary `key1=value1,key2=value2,...` of meta-information about the current LPLX file format. Keys should include `version` (a basic file format version), `ltxpdflinksversion` (version of `ltxpdflinks` used to generate this file), `features` (a list of "features" the file provides, might add new "features" in the future). The "`bbox`" feature means that the LPLX file makes a call to `\lplxSetBbox` with the graphic bounding box information.

```
%%\long\def\LPLX#1#2{\begingroup #2\endgroup}
\let\LPLX\@gobble
```

In the contents of the LPLX data (second argument of `\LPLX`), we have a sequence of `\lplxAbcDef` commands in a declarative-style interface. The commands are the following.

`\lplxGraphic` Specify the original (PDF) graphic file name. Specify base file name (#1) and extension (#2 including dot) separately

```
\def\lplxGraphic#1#2{%
  \def\lplxiGraphicBaseFname{#1}%
  \def\lplxiGraphicExt{#2}%
  \def\lplxiGraphicFname{#1#2}%
  \lplx@IncludeGraphics
}
```

`\lplxUserSpaceUnitLength` Set the base unit of the graphic. Width, height, coordinates, etc., are specified in this unit. Usually this is 1 bp = 1/72 in.

```
\def\lplxUserSpaceUnitLength#1{%
  \unitlength=#1\relax
}
```

`\lplxSetBbox` Declare what the bounding box of the graphic is. Currently the two first arguments should be `{0}{0}`.

```
\newdimen\lplxiBboxWcrdim
\newdimen\lplxiBboxHcrdim
\def\lplxSetBbox#1#2#3#4{%
  \def\lplxiBboxX{#1}%
  \def\lplxiBboxY{#2}%
  \def\lplxiBboxW{#3}%
  \def\lplxiBboxH{#4}%
  \ifdim\lplxiBboxX\p@>\z@\relax\lplx@bboxzerowarn\fi
  \ifdim\lplxiBboxX\p@<\z@\relax\lplx@bboxzerowarn\fi
  \ifdim\lplxiBboxY\p@>\z@\relax\lplx@bboxzerowarn\fi
  \ifdim\lplxiBboxY\p@<\z@\relax\lplx@bboxzerowarn\fi
}
\def\lplx@bboxzerowarn{%
  \PackageWarning{phflplx}{LPLX bounding box is not pinned at (0,0), not supported}%
}
```

`\lplxPicture` Gather all the links you want to place as an argument to a call to `\lplxPicture`.

```
\long\def\lplxPicture#1{%
  \lplx@SetupScaleAndBbox
  \lplx@a@DoScale{%
    \begin{picture}(\lplxvCropW,\lplxvCropH)(\lplxvCropX,\lplxvCropY)%
      #1%
    \end{picture}%
  }%
}
```

Finally, individual links are placed with `\lplxPutLink`. Usage is `\lplxPutLink{`⟨*x*⟩`}{`⟨*y*⟩`}{`⟨*w*⟩`}{`⟨*h*⟩`}{`⟨*hyperstartcmd*⟩`}{`⟨*hyperendtokens*⟩`}`. Here `{`⟨*hyperstartcmd*⟩`}` and `{`⟨*hyperendtokens*⟩`}` are any tokens that will be inserted immediately before and immediately after an invisible rule of the given width and height. The rule will be in curly braces, so it can be considered as a mandatory argument to the last macro token in `{`⟨*hyperstartcmd*⟩`}`. Don't forget to escape URL characters that have a LaTeX meaning that is very special (e.g. `#` → %23) because these characters are already in a macro argument and will have catcodes assigned before `\href` and friends get the chance to take care of catcodes etc. [Actually, some characters still work, like `^`, `_`, `&`, … I'm not sure what determines this. (?)]

```
\def\lplxPutLink{%
  \ifGin@clip
    \expandafter\lplx@clipputlink
  \else
    \expandafter\lplx@doputlink
  \fi
}
```

Helper to place other content than a link, using the `\lplxPutLink` mechanism (for placing latex boxes). The `\@gobble` simply gobbles the default invisible rectangular box that is supposed to be the hyperlink's content.

```
\def\lplxPlaceContent#1#2#3#4#5{% x,y,w,h,content
  \lplxPutLink{#1}{#2}{#3}{#4}{\@gobble}{#5}%
}
```

A simple helper to percent-quote special characters in an URL.

```
{\catcode`\%=12\relax
  \gdef\lplx@percent{%}
}
\def\lplxHexChar#1{%
  \lplx@percent#1%
}
```

3.3 Providing content to the figure from LaTeX — "placing boxes"

Special links in the PDF figure of the form `latexbox://<domain>/<boxname>?<arguments>` allow you to place custom content in the figure from LaTeX. You can define that content using the `\phflplxDefineBox` command:

`\phflplxDefineBox`

```
\long\def\phflplxDefineBox#1#2#3{%
  \expandafter\gdef\csname lplx@def@box@#1@#2\endcsname{#3}%
}
```

Internal command that will be used to render the box from the generated LPLX code:

```
\long\def\lplx@render@box@set@halign@p#1{#1}
\long\def\lplx@render@box@set@halign@l#1{%
  \raggedright#1}
\long\def\lplx@render@box@set@halign@c#1{%
  \centering#1}
\long\def\lplx@render@box@set@halign@r#1{%
  \raggedleft#1}
\def\lplx@render@box@set@halign@ml{\makebox[\hsize][l]}
\def\lplx@render@box@set@halign@mc{\makebox[\hsize][c]}
\def\lplx@render@box@set@halign@mr{\makebox[\hsize][r]}
\def\lplx@render@box@set@halign@ms{\makebox[\hsize][s]}
\def\lplxRenderBox[#1]#2[#3]#4#5#6#7{% halign,w,valign,h,stylemacro,boxdomain,boxname
%%  \message{***LPLX BOX *** |\detokenize{#1}| |\detokenize{#2}| |\detokenize{#3}| |\detokeni
  \parbox[b][#4][#3]{#2}{%
    \csname lplx@render@box@set@halign@#1\endcsname
    {#5{\lplx@render@box@contents{#6}{#7}}}%
  }%
}
\def\lplx@render@box@contents#1#2{%
  \csname lplx@def@box@#1@#2\endcsname
}
```

3.4 Internal Implementation Commands

\lplx@IncludeGraphics This command actually includes the underlying graphics. TODO: support for \lplxiGraphicFname in case the PDF base file name differs from the LPLX file base name? (Though that sounds like asking for trouble.)

```
\def\lplx@IncludeGraphics{%
  \edef\x{\noexpand\hbox to 0pt{%
      \noexpand\Ginclude@graphics{\Gin@base\lplxiGraphicExt}}}%
  \x
}
```

\lplx@SetupScaleAndBbox Utility to set up the appropriate command arguments to use for \scalebox, etc.

```
\def\lplx@noscale{1,!}
\def\lplx@exclam{!}
\def\lplx@a@DoScale{}%
\def\lplx@SetupScaleAndBbox{%
```

First, we locally define a macro \lplx@a@DoScale{...} that will generate the correct \scalebox call with the given contents, according to the requested size.

```
  \def\lplx@tmp{\Gin@scalex,\Gin@scaley}%
  \ifx\lplx@tmp\lplx@noscale%
    \def\lplx@a@DoScale{}%
  \else
    \ifx\Gin@scaley\lplx@exclam
      \edef\lplx@a@DoScale{\noexpand\scalebox{\Gin@scalex}}%
    \else
      \ifx\Gin@scalex\lplx@exclam
        \edef\lplx@a@DoScale{\noexpand\scalebox{\Gin@scaley}}%
      \else
        \edef\lplx@a@DoScale{\noexpand\scalebox{\Gin@scalex}[\Gin@scaley]}%
      \fi
    \fi
  \fi
```

Second, we need to take care of setting the bounding box correctly. Define the macros \lplxvCropX and \lplxvCropY which are the (X, Y) position of the lower left corner the part of the image we want to pick out, in user space units. Define \lplxvCropW and \lplxvCropH as the requested width & height of the subimage we want to use.

```
  \edef\lplxvCropX{\Gin@llx}%
  \edef\lplxvCropY{\Gin@lly}%
  \edef\lplxvCropW{\strip@pt\dimexpr\Gin@urx pt-\Gin@llx pt\relax}%
  \edef\lplxvCropH{\strip@pt\dimexpr\Gin@ury pt-\Gin@lly pt\relax}%
}
```

Tools to place links & clip them if necessary.

```
\def\lplx@phantomrule#1#2{% w, h (dimensions given as the string "Xbp")
  {\phantom{\rule{#1}{#2}}}%
}
\def\lplx@doputlink#1#2#3#4#5#6{% x,y,w,h,hyperstart,hyperend
  \put(#1,#2){#5{\lplx@phantomrule{#3bp}{#4bp}}#6}%
}%
\newdimen\lplx@tmpx
\newdimen\lplx@tmpy
\newdimen\lplx@tmpw
\newdimen\lplx@tmph
\def\lplx@clipputlink#1#2#3#4#5#6{% x,y,w,h,hyperstart,hyperend
```

Some notes: 1) We use “pt” as dummy unit of measure here just to do the floating point arithmetic and we use `\strip@pt` at the end. 2) Here `\lplx@maybeskip` serves as a flag that if set, asserts the link was entirely cropped out of the picture. Initially it expands to an empty string but when set it expands to “`\p@<\z@\relax`” (= “`1pt < 0pt`”), so it can be placed in front of all `\ifdim`’s so that they are skipped if the link was determined to be out of the picture.

```
  \def\lplx@maybeskip{}%
  \def\lplx@setskip{\def\lplx@maybeskip{\p@<\z@\relax}}%
  \lplx@tmpx=#1pt\relax
  \lplx@tmpy=#2pt\relax
  \lplx@tmpw=#3pt\relax
  \lplx@tmph=#4pt\relax
  \ifdim\lplx@maybeskip\lplx@tmpx<\lplxvCropX\p@\relax
    \ifdim\dimexpr\lplx@tmpx+\lplx@tmpw>\lplxvCropX\p@\relax
      \lplx@tmpw=\dimexpr\lplx@tmpx+\lplx@tmpw-\lplxvCropX\p@\relax
      \lplx@tmpx=\lplxvCropX\p@\relax
    \else
      \lplx@setskip
    \fi
  \fi
  \ifdim\lplx@maybeskip\dimexpr\lplx@tmpx+\lplx@tmpw
        >\dimexpr\lplxvCropX\p@+\lplxvCropW\p@\relax
    \ifdim\lplx@tmpx<\dimexpr\lplxvCropX\p@+\lplxvCropW\p@\relax
      \lplx@tmpw=\dimexpr\lplxvCropX\p@+\lplxvCropW\p@-\lplx@tmpx\relax
    \else
      \lplx@setskip
    \fi
  \fi
  \ifdim\lplx@maybeskip\lplx@tmpy<\lplxvCropY\p@\relax
```

```
    \ifdim\dimexpr\lplx@tmpy+\lplx@tmph>\lplxvCropY\p@\relax
      \lplx@tmph=\dimexpr\lplx@tmpy+\lplx@tmph-\lplxvCropY\p@\relax
      \lplx@tmpy=\lplxvCropY\p@\relax
    \else
      \lplx@setskip
    \fi
  \fi
  \ifdim\lplx@maybeskip\dimexpr\lplx@tmpy+\lplx@tmph
        >\dimexpr\lplxvCropY\p@+\lplxvCropH\p@\relax
    \ifdim\lplx@tmpy<\dimexpr\lplxvCropY\p@+\lplxvCropH\p@\relax
      \lplx@tmph=\dimexpr\lplxvCropY\p@+\lplxvCropH\p@-\lplx@tmpy\relax
    \else
      \lplx@setskip
    \fi
  \fi
  \ifdim\lplx@maybeskip\p@>\z@\relax
    \edef\x{\noexpand\lplx@doputlink%
      {\strip@pt\lplx@tmpx}{\strip@pt\lplx@tmpy}%
      {\strip@pt\lplx@tmpw}{\strip@pt\lplx@tmph}}%
    \x{#5}{#6}%
  \fi
}%

```

Change History

v0.1
General: Initial version 1

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.