The graphicx-psmin package *

Hendri Adriaens

v1.2 (2020/11/14)

Abstract

This package is an extension of the standard graphics bundle [1] and provides a way to include repeated PostScript graphics (ps, eps) only once in a PostScript document. This provides a way to get smaller PostScript documents when having, for instance, a logo on every page. This package only works when post-processed with dvips, *which should at least have version 5.95b!* The difference for the pdf file is minimal (as Ghostscript already includes only a single copy of graphics files).

Contents

1 Introduction

Including a PostScript graphic only once in a PostScript document has been a hot issue for a long time in the LaTeX world. It pops up regularly at many newsgroups, but never a fully satisfying answer was supplied. Some answers vaguely describe some `\special` commands, but do not give the details, some suggest to use a TeX box (which does prevent LaTeX from reading the graphic many times, but still includes it many times in the PostScript) and some do suggest to modify the PostScript file manually, which is far too cumbersome. This package is an extension of the standard graphics bundle [1] and supplies an automated method to do this job.

The technique used by this package has been described by Thomas Greer [2] (see for more details [3]). The implementation in LaTeX boils down to the following steps.

1. Scan the graphics file for the bounding box.
2. Wrap the graphics file in a PostScript function that defines it as an object which can be used multiple times. This function needs the bounding box.
3. Write the entire code to the header of the PostScript file, via the dvi file, using `\specials`.

*This package can be downloaded from the CTAN mirrors: `/macros/latex/contrib/graphicx-psmin`. See `graphicx-psmin.dtx` for information on installing graphicx-psmin into your TeX or LaTeX distribution and for the license of this package.

This will load the graphic in the header of the PostScript file. When reusing the graphic, there is another PostScript function to be inserted which will load the graphic from the graphic object.

In this scheme, step 2 is the hardest one. The first implementation of this method in this package attempted to load the entire graphics file, line by line (which is slow), in a macro, add the necessary PostScript code and write the content of the macro to the dvi using `\special{!...}`. Unfortunately, this didn't work as dvips applied its own line breaking mechanism to the content without taking into account manual line breaks. This meant that the content of the graphics file was modified when it arrived in the PostScript document, often too much to be processed successfully again.

The second attempt in this package wrote the entire graphics file to hard disk, together with the PostScript function so that it could be loaded using `\special{header=...}`. Besides still being slow, this also meant a lot of hard disk overhead.

After some discussion with Karl Berry a proposal for extending the header special was posted on the TEX-k mailinglist. The alternative of changing the line breaking mechanism of dvips was not considered as that could change the behavior in existing documents. The proposal was to extend the header special by allowing

```
\special{header={some file.ps} pre={pre code} post={post code}}
```

The package uses the two extra parameters in this special to pass the PostScript function that will make an object of the graphics file in `some file.ps` to the PostScript file. The `header` itself still copies the file directly into the PostScript file, which avoids needing to read the entire file with TEX.

As mentioned in the abstract, this only works when post-processed with dvips, *version v5.95b or newer!* It will, in general, not make a significant difference in the size of your pdf file generated by ps2pdf (Ghostscript).[1] However, it will decrease the size of your PostScript file if you use the same graphic over and over in your document.

2 Using the package

Using the package is very simple. Replace the loading of the graphicx package, like

```
\usepackage[your options]{graphicx}
```

by

```
\usepackage[your options]{graphicx-psmin}
```

This will load the graphicx package internally, pass on all of the options specified in `your options` to the graphicx package (for instance, `draft,dvips`) and make all definitions necessary to do the job.

`\loadgraphics[`⟨*bb*⟩`]{`⟨*list of graphics*⟩`}`

`\loadgraphics` Each graphic that you want to reuse in the document, but which you want to be included only once, should be listed in the command `\loadgraphics`. This command can only be used *in the preamble* of your document. Example:

```
\loadgraphics{mylogo1.ps,mylogo2.eps}
```

[1] As Ghostscript already embeds graphics with size larger than `MaxInlineImageSize` only once in the pdf.

If a graphics file does not contain a bounding box, you can include that in the optional argument. This argument will be used for every graphic in the list. Example

```
\loadgraphics[0 0 250 420]{mylogo1.ps,mylogo2.eps}
\loadgraphics{mylogo3.ps}
```

\includegraphics

That's all! You can use the usual graphicx `\includegraphics` command to include and scale or rotate your graphics. This command will pick up the loaded graphics and use a PostScript function to insert the graphic from the PostScript header. If the graphic has not been loaded with `\loadgraphic`, the `\includegraphics` command works as usual (see also the graphicx documentation [1] or a LaTeX manual like [4]).

If the graphic does not contain bounding box information, then this should be included in the `\loadgraphics` command as described above. You can include it in the `\includegraphics` command as well, but if the two bounding boxes are not equal, the result might be unexpected.

As mentioned before, the package only works with the dvips driver, version 5.95b or newer. When running LaTeX (not pdfLaTeX), the dvips driver will be loaded automatically by graphicx, but you can also specify it yourself when loading the graphicx-psmin package. Any options that you specify will be passed on to the graphicx package. When graphicx-psmin finds out that you are using another driver than dvips, it will generate an error message and disable itself, after which the `\loadgraphics` command has no function anymore and all graphics inclusion jobs are left to the graphicx package.

To finish with, here is another example demonstrating the use of the package. The example assumes that `figure.ps` is present in the `figures` subdirectory and that `graphic.eps` is in the `graphics` subdirectory.

```
\documentclass{article}
\usepackage{graphicx-psmin}
\graphicspath{{figures/}{graphics/}}
\loadgraphics{figure.ps,graphic.eps}
\begin{document}
\includegraphics[scale=.2]{figure.ps}
\includegraphics[scale=.4]{figure.ps}\\
\includegraphics[angle=45]{graphic.eps}
\includegraphics[angle=90]{graphic.eps}
\end{document}
```

When running a file like the one above and generating the PostScript using dvips, one will see that both graphics are included only once in the PostScript document and that including one of those graphics another time, has almost no effect on the size of the generated PostScript document.

3 Implementation

```
% Initialize.
⟨*graphicx-psmin⟩
\NeedsTeXFormat{LaTeX2e}[1995/12/01]
\ProvidesPackage{graphicx-psmin}
  [2020/11/14 v1.2 single PostScript graphics inclusion (HA)]
\DeclareOption*{\PassOptionsToPackage\CurrentOption{graphicx}}
\ProcessOptions\relax
\RequirePackage{graphicx}
```

Check the requested driver.

```
\def\gxpsm@tempa{dvips.def}
\ifx\Gin@driver\gxpsm@tempa\else
  \PackageError{graphicx-psmin}{This package cannot be used with any
    \MessageBreak back end driver other than dvips!}\@ehd
  \def\loadgraphics{\@testopt\gxpsm@loadgraphics{}}
  \def\gxpsm@loadgraphics[#1]#2{}
  \expandafter\endinput
\fi
```

In draft mode, no graphics should be loaded, so we eat the argument and hence all graphics will be handled by graphicx from here.

```
\ifGin@draft
  \def\loadgraphics{\@testopt\gxpsm@loadgraphics{}}
  \def\gxpsm@loadgraphics[#1]#2{}
  \expandafter\endinput
\fi
```

`\gxpsm@loaded` Initialize the list of loaded graphics.

```
\def\gxpsm@loaded{}
```

`\@namexdef` `{`⟨*csname*⟩`}`

Defines the macro with name ⟨*csname*⟩ using `\xdef`.

```
\def\@namexdef#1{\expandafter\xdef\csname#1\endcsname}
```

`\loadgraphics` Load graphics.

```
\def\loadgraphics{\@testopt\gxpsm@loadgraphics{}}
```

`\gxpsm@loadgraphics` `[`⟨*bb*⟩`]{`⟨*list of graphics*⟩`}`

Load each of the graphics in ⟨*list of graphics*⟩ into the dvi and eventually the PostScript using `\specials`. Use ⟨*bb*⟩ for the bounding box for all these graphics if specified. If not, we search the file for the bounding box.

```
\def\gxpsm@loadgraphics[#1]#2{%
```

Loop over all graphics.

```
  \@for\gxpsm@file:=#2\do{%
    \begingroup
```

If the file exists in the graphics path, continue.

```
    \gxpsm@checkfile\gxpsm@file{%
```

If no explicit bounding box in `#1`, try finding one in the file, otherwise use the one in `#1`.

```
      \ifx\@empty#1\@empty
        \Gread@eps{\Gin@base\Gin@ext}%
      \else
        \Gread@parse@bb#1 \\
      \fi
```

Save the bounding box for this graphic.

```
      \@namexdef{\Gin@base\Gin@ext @llx}{\Gin@llx}%
      \@namexdef{\Gin@base\Gin@ext @lly}{\Gin@lly}%
      \@namexdef{\Gin@base\Gin@ext @urx}{\Gin@urx}%
      \@namexdef{\Gin@base\Gin@ext @ury}{\Gin@ury}%
```

Transform the PostScript internal name of the graphic as '/' can't be used in variable names.

```
      \gxpsm@getcfile
```

Write the graphic body together with the extra functions to the dvi file using a header special. This requires dvips v5.95b or newer to work!

```
      \AtBeginDvi{\special{header={\Gin@base\Gin@ext}
        pre={/\gxpsm@cfile-data^^Jcurrentfile^^J%
          << /Filter /SubFileDecode^^J/DecodeParms << /EODCount 0
          /EODString (*HA-EOD-??3.1416926!!*) >>^^J>>
          /ReusableStreamDecode filter^^J%
          \@percentchar\@percentchar BeginDocument:
          \Gin@base\Gin@ext^^J%
        }
        post={\@percentchar\@percentchar EndDocument^^J%
          *HA-EOD-??3.1416926!!*^^Jdef^^J/\gxpsm@cfile-form^^J%
          << /FormType 1^^J/BBox
          [\Gin@llx\space\Gin@lly\space\Gin@urx\space\Gin@ury]^^J%
          /Matrix [1 0 0 1 0 0]^^J/PaintProc^^J{ pop^^J%
          /ostate save def^^J/showpage {} def^^J%
          /setpagedevice /pop load def^^J%
          \gxpsm@cfile-data 0 setfileposition
          \gxpsm@cfile-data cvx exec^^J%
          ostate restore^^J} bind^^J>> def%
        }
      }}%
```

Add the file to the list of loaded graphics for `\includegraphics`.

```
      \xdef\gxpsm@loaded{%
        \gxpsm@loaded\ifx\gxpsm@loaded\@empty\else,\fi
        \Gin@base\Gin@ext
      }%
    }%
    \endgroup
  }%
}
```

Avoid using `\loadgraphics` outside the preamble.

```
\@onlypreamble\loadgraphics
\@onlypreamble\gxpsm@loadgraphics
```

`\gxpsm@getcfile` `\gxpsm@g@tcfile` These two macros replace any occurrence of '/' by '_' so that the name can be used inside PostScript. This uses a well known `\lowercase` trick.

```
\def\gxpsm@getcfile{%
  \edef\gxpsm@tempa{%
    \noexpand\gxpsm@g@tcfile\Gin@base\Gin@ext\noexpand\@nil
  }%
  \gxpsm@tempa
}
\def\gxpsm@g@tcfile#1\@nil{%
  \begingroup\lccode`\/`\_\lowercase{\endgroup\def\gxpsm@cfile{#1}}%
}
```

`\Ginclude@graphics` {⟨*file*⟩}

We redefine this internal of graphics. If the graphic has been loaded, use a PostScript function to reload it from the header. Else, follow the usual track of graphicx.

```
\def\Ginclude@graphics#1{%
  \begingroup
```

Graphic exists? This will also produce `\Gin@base` and `\Gin@ext`.

```
\gxpsm@checkfile{#1}{%
```

Graphic loaded?

```
  \@expandtwoargs\in@{,\Gin@base\Gin@ext,}{,\gxpsm@loaded,}%
  \ifin@
```

If the user didn't supply a bounding box in the `\includegraphics` command, use the one that we found while scanning the graphic.

```
    \ifGin@bbox\else
      \Gin@bboxtrue
      \edef\Gin@llx{\@nameuse{\Gin@base\Gin@ext @llx}}%
      \edef\Gin@lly{\@nameuse{\Gin@base\Gin@ext @lly}}%
      \edef\Gin@urx{\@nameuse{\Gin@base\Gin@ext @urx}}%
      \edef\Gin@ury{\@nameuse{\Gin@base\Gin@ext @ury}}%
    \fi
```

Use graphics internals to do computations etcetera and in the end, use the graphic.

```
    \Gin@setfile{psdirect}{}{\Gin@base\Gin@ext}%
  \else
```

This is the usual route from `\Ginclude@graphics` from graphics for non-loaded graphics.

```
    \@ifundefined{Gin@rule@\Gin@ext}{%
      \ifx\Gin@rule@*\@undefined
        \@latex@error{Unknown graphics extension: \Gin@ext}\@ehc
      \else
        \expandafter\Gin@setfile\Gin@rule@*{\Gin@base\Gin@ext}%
      \fi
    }{%
      \expandafter\expandafter\expandafter\Gin@setfile
        \csname Gin@rule@\Gin@ext\endcsname{\Gin@base\Gin@ext}%
    }%
  \fi
}%
\endgroup
}
```

`\gxpsm@checkfile` {⟨*file*⟩}{⟨*actions*⟩}

This is part of graphics' `\Ginclude@graphics` which checks a graphic file in the graphics path. We perform ⟨*actions*⟩ when the file is all right. We separated this part as it is reused several times in this package.

```
\def\gxpsm@checkfile#1#2{%
  \let\input@path\Ginput@path
  \ifx\unquote@name\@undefined
    \filename@parse{#1}%
  \else
    \expandafter\filename@parse\expandafter{\detokenize{#1}}%
  \fi
  \ifx\filename@ext\relax
    \@for\Gin@temp:=\Gin@extensions\do{%
      \ifx\Gin@ext\relax
        \Gin@getbase\Gin@temp
      \fi
    }%
```

```
  \else
    \Gin@getbase{\Gin@sepdefault\filename@ext}%
    \ifx\Gin@ext\relax
      \@warning{File `#1' not found}%
      \def\Gin@base{\filename@area\filename@base}%
      \edef\Gin@ext{\Gin@sepdefault\filename@ext}%
    \fi
  \fi
  \ifx\Gin@ext\relax
    \@latex@error{File `#1' not found}%
      {I could not locate the file with any of these extensions:^^J%
      \Gin@extensions^^J\@ehc}%
  \else#2\fi
}
```

`\Ginclude@psdirect` `{`⟨*file*⟩`}`

This inserts the PostScript function needed to reload ⟨*file*⟩ from the PostScript header. This is based on `\Ginclude@eps` from dvips.def (graphics).

```
\def\Ginclude@psdirect#1{%
  \message{<#1>}%
  \bgroup
  \def\@tempa{!}%
  \gxpsm@getcfile
  \dimen@\Gin@req@width
  \dimen@ii.1bp%
  \divide\dimen@\dimen@ii
  \@tempdima\Gin@req@height
  \divide\@tempdima\dimen@ii
  \special{ps:@beginspecial
    \Gin@llx\space @llx \Gin@lly\space @lly
    \Gin@urx\space @urx \Gin@ury\space @ury
    \ifx\Gin@scalex\@tempa\else\number\dimen@\space @rwi\fi
    \ifx\Gin@scaley\@tempa\else\space\number\@tempdima\space @rhi\fi
    \ifGin@clip\space @clip\fi\space @setspecial^^J
    save \gxpsm@cfile-form execform restore showpage @endspecial
  }%
  \egroup
}

%</graphicx-psmin>
```

References

[1] David Carlisle. graphics bundle. `CTAN:/macros/latex/required/graphics`.

[2] Thomas Greer. Reusable content caching in postscript. `http://www.tgreer.com/eps_vdp2.html`.

[3] Adobe Systems Incorporated. Postscript language reference manual. `http://www.adobe.com/products/postscript/pdfs/PLRM.pdf`.

[4] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, and Chris Rowley. *The LaTeX Companion, Second Edition.* Addison-Wesley, 2004.

Acknowledgements

The author is grateful to Thomas Greer and Uwe Kern for help and suggestions. Many thanks to Akira Kakuto for providing a dvips patch which makes this package possible. The author is greatly indebted to Karl Berry for support and for providing a test environment for the dvips patch on the TUG server. Finally a word of thanks for David Carlisle for providing a fix to make the package work with the latest graphicx package.

Version history

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.