The graphicscache package

Max Schwarz
max.schwarz@online.de

CTAN: http://www.ctan.org/pkg/graphicscache

v0.3 from 2021/08/02

1 Introduction

The `graphicx` package offers the versatile `\includegraphics` command, which offers image transformations like scaling, cropping, rotation, etc. However, these transformations have to be performed on every compilation of the document. Users can avoid this with the `draft` option at the cost of not seeing the images.

Furthermore, images are always included as-is with full resolution, even if they are shown at a very small actual size. This increases compilation time again and leads to large output files. A typical solution is to resize the input images to a lower resolution—but here, the user has to manually calculate or guess the required resolution. What we really want is to specify a *document DPI*, which automatically leads to the correct image resolution. This is possible using post-processing tools like ghostscript, but these do not help with compilation times and are typically not applicable for preprint servers or journals which require LaTeX sources.

`graphicscache` aims to solve these problems by decoupling the rendering of images from the actual inclusion. Images are rendered to the correct size using a separate `pdflatex` call, post-processed with ghostscript, and then included as PDF. As a bonus, the resulting PDF is cached, resulting in very fast recompilation times.

2 Usage

`graphicscache` requires the usage of the `\write18` call (also called shell escape). For `pdflatex`, you have to specify the `-shell-escape` argument during compilation. After enabling shell-escape, simply call

```
\usepackage{graphicscache}
```

to enable caching.

`graphicscache` overrides the `\includegraphics` command so that you can use it as usual. Internally, it calculates a hash from the `includegraphics` arguments and the package options to generate a cache key. If you change the input image file or the options, the file will be automatically rendered again.

Note: The first compilation process might take a while, since the cached PDFs are generated one-by-one.

2.1 Generated files

`graphicscache` will create a folder called `graphicscache` in the compilation directory. Additionally, latex output files under the jobname `graphicscacheout` will be created. All of these files and folders are temporary and can be deleted safely (at the cost of re-creating the cache).

2.2 Package options

`compress=false|flat|jpeg`

Specifies the image compression algorithm. If `false`, the ghostscript call is skipped, thus embedding the image at its original resolution and format. If `flat`, the lossless `FlatEncode` algorithm is chosen. Finally, `jpeg` indicates that JPEG encoding using `DCTEncode` is to be performed.

Note: In the `flat` case, the images are still downsampled to match the DPI specified using the `dpi` key.

`dpi=`⟨*number*⟩

Controls the image resolution in dots per inch. The default value is 300. This option only takes effect if `compress` is not `false`.

`qfactor=`⟨*number*⟩

This controls the quality parameter of the JPEG encoder (see `compress`). Smaller values give higher quality. The default value is 0.15, which corresponds to the "Maximum" setting mentioned by Adobe.

`listing=true|false`

If enabled, `graphicscache` will write an extra `.graphicscache` file with mappings from `includegraphics` arguments to cache files. This can be used to produce a version of the TeX source code that directly references the PDF files instead of the original sources.

`render=true|false`

Controls whether rendering is allowed. The default is `true`. If `false`, `graphicscache` is not allowed to create new cache files. Instead, it will attempt to use the appropriate cache file. If it does not exist, `graphicscache` will fall back to `graphicx` in-place rendering. This can be used to perform a final release (see **??**).

`cachedir=`⟨*dir*⟩

This key can be used to move the cache directory to another location. The default value is `graphicscache`.

2.3 Macros

`\includegraphics[args]{path}`

This behaves exactly like the original `graphicx` `\includegraphics`. However, only a limited number of keys in `args` is supported at the moment: `width`, `height`, `trim`, `clip`, `angle`, `origin`, `keepaspectratio`, `scale`. In addition, you can specify any of the package options above here as well. For example, you might want to disable compression for a particular image with

`\includegraphics[width=...,compress=false]{...}`.

3 Tips & Tricks

3.1 Releasing your manuscript

In case you want to upload your manuscript to a journal or preprint server, you can use `graphicscache` to make your work easier:

1. Use `latexpand` to strip comments and flatten your tex sources into one file (optional).
2. Compile the file as usual to generate the cache files.
3. Compile the file again with the `-recorder` command line option. Here, we want to hide accesses of the original image files (`render=false`), but changing the package options will change the cache hash. For this purpose, `graphicspath` also reacts to a global definition of `\graphicscache@inhibit`, which has the same effect as `render=false`.
4. The generated `.fls` file will contain all cache files that are needed to compile your manuscript. Package them and your `.tex` file.

This process is automated in the `release.sh` script included in the distribution.

4 Implementation

```
\NeedsTeXFormat{LaTeX2e}[1994/06/01]
\ProvidesPackage{graphicscache}[2018/10/02 Graphics Cache]
\RequirePackage{graphicx}
\RequirePackage{xstring}
\RequirePackage{filemod}
\RequirePackage{letltxmacro}
\RequirePackage{pgfopts}
\RequirePackage{ifplatform}
\RequirePackage{pdftexcmds}
\RequirePackage{ltxcmds}
\newif\ifgraphicscache@render
\newif\ifgraphicscache@compress
\newif\ifgraphicscache@listing
\newif\ifgraphicscache@hashshortnames
\def\graphicscache@graphicsargs{}
\newlength\graphicscache@tmplen
\newcommand{\graphicscache@addarg}[1]{%
  \ifx\graphicscache@graphicsargs\empty
    \edef\graphicscache@graphicsargs{#1}%
  \else
    \edef\graphicscache@graphicsargs{\
       graphicscache@graphicsargs,#1}%
  \fi
}
\pgfkeys{
  /graphicscache/.cd,
  render/.is if=graphicscache@render,
  render=true,

  cachedir/.store in=\graphicscache@cachedir,
  cachedir={graphicscache},

  compress/.is choice,
  compress/false/.code={\
     graphicscache@compressfalse},
  compress/jpeg/.code={\graphicscache@compresstrue
     \def\graphicscache@compress@mode{DCTEncode}},
```

```
compress/flat/.code={\graphicscache@compresstrue \def\graphicscache@compress@mode{FlatEncode}},
compress=jpeg,

dpi/.store in=\graphicscache@dpi,
dpi=300,

qfactor/.store in=\graphicscache@qfactor,
qfactor={0.15},

hashshortnames/.is if=graphicscache@hashshortnames,
hashshortnames=false,
```

We now define the list of supported graphicx arguments:

```
width/.code={%
  \setlength\graphicscache@tmplen{#1}%
  \graphicscache@addarg{width=\the\graphicscache@tmplen}%
},
height/.code={%
  \setlength\graphicscache@tmplen{#1}%
  \graphicscache@addarg{height=\the\graphicscache@tmplen}%
},
trim/.code={\graphicscache@addarg{trim=#1}},
clip/.code={\graphicscache@addarg{clip}},
angle/.code={%
  \edef\graphicscache@tmp{#1}%
  \graphicscache@addarg{angle=\graphicscache@tmp}%
},
origin/.code={\graphicscache@addarg{origin=#1}},
keepaspectratio/.code={\graphicscache@addarg{keepaspectratio}},
scale/.code={%
  \edef\graphicscache@tmp{#1}%
  \graphicscache@addarg{scale=\graphicscache@tmp}%
},
page/.code={%
  \edef\graphicscache@tmp{#1}%
  \graphicscache@addarg{page=\graphicscache@tmp}%
},
```

```
  listing/.is if=graphicscache@listing,
  listing=false,
%
% adjustbox package
%
  frame/.code={%
    \edef\graphicscache@tmp{#1}%
    \graphicscache@addarg{frame=\graphicscache@tmp}%
  },
  valign/.code={%
    \edef\graphicscache@tmp{#1}%
    \graphicscache@addarg{valign=\graphicscache@tmp}%
  },
  raise/.code={%
    \edef\graphicscache@tmp{#1}%
    \graphicscache@addarg{raise=\graphicscache@tmp}%
  },
}
\ProcessPgfOptions{/graphicscache}\relax
\ifdefined\graphicscache@inhibit
  \pgfkeys{/graphicscache/render=false}%
\fi
\ifgraphicscache@listing
  \newwrite\graphicscache@listout
  \immediate\openout\graphicscache@listout=\jobname
    .graphicscache
\fi
```

`\graphicscache@dorender`

Here, we actually perform the rendering. Sadly, this is quite complex due to cross-platform support.

```
\newcommand{\graphicscache@dorender}{%
  \message{Rendering \graphicscache@outputhash: \
    graphicscache@fname\space with args: \
    graphicscache@graphicsargs\space (master file)
    }%
  \ifwindows
    \immediate\write18{md "\graphicscache@cachedir"
      2>NUL}%
```

```
\else
  \immediate\write18{mkdir -p "\↙
     graphicscache@cachedir"}%
\fi
```

First, render the graphics.

```
\ifwindows
  \immediate\write18{del /q \↙
     graphicscache@cachedir\string\↙
     graphicscacheout.pdf}
  \immediate\write18{pdflatex
    -jobname graphicscacheout
    -interaction nonstopmode
    -output-directory "\graphicscache@cachedir"
    "\string\documentclass{standalone}
    \string\usepackage{graphicx}
    \string\usepackage[export]{adjustbox}
    \string\begin{document}\string\↙
       includegraphics[\↙
       graphicscache@graphicsargs]{\↙
       graphicscache@fname}\string\end{document}"
  }%
  \IfFileExists{\graphicscache@cachedir/↙
     graphicscacheout.pdf}{}{%
    \PackageError{graphicscache}{External ↙
       pdflatex call failed (see above)}{}%
    \def\graphicscache@output{}%
  }
\else
  \immediate\write18{pdflatex
    -jobname graphicscacheout
    -interaction nonstopmode
    -output-directory "\graphicscache@cachedir"
    '\string\documentclass{standalone}
    \string\usepackage{graphicx}
    \string\usepackage[export]{adjustbox}
    \string\begin{document}\string\↙
       includegraphics[\↙
       graphicscache@graphicsargs]{\↙
       graphicscache@fname}\string\end{document}'
    > /dev/null || rm "\graphicscache@cachedir/↙
       graphicscacheout.pdf"
  }%
\fi
```

Now, call ghostscript for compression—if required, otherwise just copy the file.

```
\ifgraphicscache@compress
  \message{With compression: \graphicscache@compress@mode}%
  \ifwindows
    \immediate\write18{mgs
      -sOutputFile=\graphicscache@output\space
      -sDEVICE=pdfwrite
      -dCompatibilityLevel=1.4
      -dPDFSETTINGS=/prepress
      -dNOPAUSE -dQUIET -dBATCH
      -c ".setpdfwrite <<
        /AutoFilterColorImages false
        /EncodeColorImages true
        /ColorImageFilter /\graphicscache@compress@mode\space
        /ColorImageDict << /ColorTransform 1 /QFactor \graphicscache@qfactor\space /Blend 1 /HSamples [1 1 1 1] /VSamples [1 1 1 1] >>
        /ColorImageResolution \graphicscache@dpi\space
        /AutoFilterGrayImages false
        /EncodeGrayImages true
        /GrayImageFilter /\graphicscache@compress@mode\space
        /GrayImageDict << /ColorTransform 1 /QFactor \graphicscache@qfactor\space /Blend 1 /HSamples [1 1 1 1] /VSamples [1 1 1 1] >>
        /GrayImageResolution \graphicscache@dpi\space
      >> setdistillerparams"
      -f \graphicscache@cachedir\string\graphicscacheout.pdf
    }%
  \else
    \immediate\write18{gs
      -sOutputFile=\graphicscache@output\space
      -sDEVICE=pdfwrite
      -dCompatibilityLevel=1.4
      -dPDFSETTINGS=/prepress
```

```
154         -dNOPAUSE -dQUIET -dBATCH
155         -c '.setpdfwrite <<
156           /AutoFilterColorImages false
157           /EncodeColorImages true
158           /ColorImageFilter /\↙
                 graphicscache@compress@mode\space
159           /ColorImageDict << /ColorTransform 1 /↙
                 QFactor \graphicscache@qfactor\space /↙
                 Blend 1 /HSamples [1 1 1 1] /VSamples ↙
                 [1 1 1 1] >>
160           /ColorImageResolution \graphicscache@dpi\↙
                 space
161           /AutoFilterGrayImages false
162           /EncodeGrayImages true
163           /GrayImageFilter /\↙
                 graphicscache@compress@mode\space
164           /GrayImageDict << /ColorTransform 1 /↙
                 QFactor \graphicscache@qfactor\space /↙
                 Blend 1 /HSamples [1 1 1 1] /VSamples ↙
                 [1 1 1 1] >>
165           /GrayImageResolution \graphicscache@dpi\↙
                 space
166         >> setdistillerparams'
167         -f \graphicscache@cachedir/graphicscacheout↙
              .pdf || rm \graphicscache@output
168       }%
169     \fi
170   \else
171     \message{Direct}%
172     \ifwindows
173       \immediate\write18{
174         copy \graphicscache@cachedir\string\↙
              graphicscacheout.pdf \↙
              graphicscache@output
175       }%
176     \else
177       \immediate\write18{
178         cp \graphicscache@cachedir/graphicscacheout↙
              .pdf \graphicscache@output
179       }%
180     \fi
181   \fi
182 }
```

save original \includegraphics

```
\LetLtxMacro\graphicscache@includegraphics\includegraphics%
\newcommand\graphicscache@native{%
  \expandafter\graphicscache@includegraphics\expandafter[\graphicscache@graphicsargs]{\graphicscache@fname}%
}
```

\graphicscache@work

This macro performs the update check: Do we need to render the file again or can we just include the cached version?

```
\newcommand{\graphicscache@work}{%
  \ifgraphicscache@render
```

Check if output file exists and is newer than input

```
    \filemodcmp{\graphicscache@fname}{\graphicscache@output}{% input is newer
      \graphicscache@dorender%
    }{% Output is newer
      \message{Already have \graphicscache@outputhash: \graphicscache@fname}%
    }%
```

If it still does not exist, we are likely in a strange environment (e.g. tabu). In that case, fall back to original includegraphics.

```
    \filemodcmp{\graphicscache@fname}{\graphicscache@output}{% input is newer/output does not exist
      \graphicscache@native
    }{% otherwise, use the generated file!
      \graphicscache@includegraphics{\graphicscache@output}%
    }%
  \else
```

Here, we just look if the output file exists. If not, fall back to original includegraphics.

```
    \IfFileExists{\graphicscache@output}{%
      \graphicscache@includegraphics{\graphicscache@output}%
```

```
    }{%
      \PackageWarning{graphicscache}{Could not find cache file \graphicscache@output, for \graphicscache@fname, falling back to native...}{}%
      \graphicscache@native
    }%
  \fi
}
```

`\graphicscache@getfname`

This macro resolves base names (i.e. includegraphics arguments with or without extensions) to relative paths.

```
\catcode`\*=11
\newif\ifgraphicscache@exists
\newcommand{\graphicscache@getfname}[1]{%
  \ifx\detokenize\@undefined\else
    \edef\Gin@extensions{\detokenize\expandafter{\Gin@extensions}}%
  \fi
  \begingroup
  \global\graphicscache@existstrue
  \let\input@path\Ginput@path
  \ltx@ifpackagelater{graphics}{2017/06/26}{%
    \set@curr@file{#1}%
    \expandafter\filename@parse\expandafter{\@curr@file}%
    \ifx\filename@ext\Gin@gzext
      \expandafter\filename@parse\expandafter{\filename@base}%
      \ifx\filename@ext\relax
        \let\filename@ext\Gin@gzext
      \else
        \edef\Gin@ext{\Gin@ext\Gin@sepdefault\Gin@gzext}%
      \fi
    \fi
  }{%
    \filename@parse{#1}%
  }%
  \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}%
    \ltx@ifpackagelater{graphics}{2017/06/26}{%
      \ifx\Gin@ext\relax
      \let\Gin@savedbase\filename@base
      \let\Gin@savedext\filename@ext
        \edef\filename@base{\filename@base\↙
           Gin@sepdefault\filename@ext}%
        \let\filename@ext\relax
         \@for\Gin@temp:=\Gin@extensions\do{%
            \ifx\Gin@ext\relax
              \Gin@getbase\Gin@temp
            \fi}%
        \ifx\Gin@ext\relax
          \let\filename@base\Gin@savedbase
          \let\filename@ext\Gin@savedext
        \fi
      \fi
    }{}%
  \fi
  \ifx\Gin@ext\relax
    \global\graphicscache@existsfalse
  \else
    \@ifundefined{Gin@rule@\Gin@ext}%
      {\global\graphicscache@existsfalse}%
      {}%
  \fi
  \ifgraphicscache@exists
    \xdef\graphicscache@fname{\Gin@base\Gin@ext}%
  \fi
  \endgroup
}
\catcode`\*=12
```

`\includegraphics`

Main entry point.

```
\renewcommand{\includegraphics}[2][]{%
  \begingroup
  \expandarg
```

Hash everything!

```
271   \edef\graphicscache@options{\@nameuse{↙
          opt@graphicscache.sty}}%
272   \pgfkeys{/graphicscache/.cd,#1}%
```

If we are rendering, we need the actual filename, so that we can check modification times. Otherwise, just assume the file exists, `\includegraphics` will throw an error itself otherwise.

```
273   \ifgraphicscache@render
274     \graphicscache@getfname{#2}%
275   \else
276     \edef\graphicscache@fname{#2}%
277     \graphicscache@existstrue
278   \fi
279   \ifgraphicscache@exists
280     \ifgraphicscache@hashshortnames
281       \edef\graphicscache@hashedname{#2}%
282     \else
283       \edef\graphicscache@hashedname{\↙
            graphicscache@fname}%
284     \fi
285     \edef\graphicscache@outputhash{\pdf@mdfivesum{\↙
          graphicscache@options\↙
          graphicscache@graphicsargs\↙
          graphicscache@hashedname}}%
286     \edef\graphicscache@output{\↙
          graphicscache@cachedir/\↙
          graphicscache@outputhash.pdf}%
287     \ifgraphicscache@listing
288       \message{graphicscache: includegraphics\{#2\}↙
              => \graphicscache@output}%
289       \immediate\write\graphicscache@listout{#2 \↙
            graphicscache@fname\space \↙
            graphicscache@output}%
290     \fi
291     \graphicscache@work
292   \else
293     \PackageError{graphicscache}{Could not find ↙
          file #2}{}%
294   \fi
295   \endgroup
296 }
```

\includegraphicscache

```
\newcommand{\includegraphicscache}[3][]{%
  \begingroup
  \expandarg
  \pgfkeys{/graphicscache/.cd,#2}%
  \includegraphics[#1]{#3}%
  \endgroup
}

\endinput
```