

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 `latexexpand` 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

```
1 \NeedsTeXFormat{LaTeX2e}[1994/06/01]
2 \ProvidesPackage{graphicscache}[2018/10/02 Graphics✓
   Cache]
3 \RequirePackage{graphicx}
4 \RequirePackage{xstring}
5 \RequirePackage{filemod}
6 \RequirePackage{letltxmacro}
7 \RequirePackage{pgfopts}
8 \RequirePackage{ifplatform}
9 \RequirePackage{pdftexcmds}
10 \RequirePackage{ltxcmds}
11 \newif\ifgraphicscache@render
12 \newif\ifgraphicscache@compress
13 \newif\ifgraphicscache@listing
14 \newif\ifgraphicscache@hashshortnames
15 \def\graphicscache@graphicsargs{}
16 \newlength\graphicscache@tmplen
17 \newcommand{\graphicscache@addarg}[1]{%
18   \ifx\graphicscache@graphicsargs\empty
19     \edef\graphicscache@graphicsargs{#1}%
20   \else
21     \edef\graphicscache@graphicsargs{\✓
       graphicscache@graphicsargs ,#1}%
22   \fi
23 }
24 \pgfkeys{
25   /graphicscache/.cd,
26   render/.is if=graphicscache@render ,
27   render=true ,
28   cachedir/.store in=\graphicscache@cachedir ,
29   cachedir={graphicscache} ,
30   compress/.is choice ,
31   compress/false/.code={\✓
     graphicscache@compressfalse} ,
32   compress/jpeg/.code={\graphicscache@compresstrue ✓
     \def\graphicscache@compress@mode{DCTEncode}} ,
```

```

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

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

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

39 hashshortnames/.is if=✓
    graphicscache@hashshortnames,
40 hashshortnames=false,

    We now define the list of supported graphicx arguments:

41 width/.code={%
42     \setlength\graphicscache@tmplen{#1}%
43     \graphicscache@addarg{width=\the\✓
        graphicscache@tmplen}%
44 },
45 height/.code={%
46     \setlength\graphicscache@tmplen{#1}%
47     \graphicscache@addarg{height=\the\✓
        graphicscache@tmplen}%
48 },
49 trim/.code={\graphicscache@addarg{trim=#1}},
50 clip/.code={\graphicscache@addarg{clip}},
51 angle/.code={%
52     \edef\graphicscache@tmp{#1}%
53     \graphicscache@addarg{angle=\graphicscache@tmp}✓
        %
54 },
55 origin/.code={\graphicscache@addarg{origin=#1}},
56 keepaspectratio/.code={\graphicscache@addarg{✓
    keepaspectratio}},
57 scale/.code={%
58     \edef\graphicscache@tmp{#1}%
59     \graphicscache@addarg{scale=\graphicscache@tmp}✓
        %
60 },
61 page/.code={%
62     \edef\graphicscache@tmp{#1}%
63     \graphicscache@addarg{page=\graphicscache@tmp}%
64 },

```

```

65   listing/.is if=graphicscache@listing,
66   listing=false,
67   %
68   % adjustbox package
69   %
70   frame/.code={%
71     \edef\graphicscache@tmp{#1}%
72     \graphicscache@addarg{frame=\graphicscache@tmp}✓
73     %
74   },
75   valign/.code={%
76     \edef\graphicscache@tmp{#1}%
77     \graphicscache@addarg{valign=\graphicscache@tmp}✓
78     %
79   },
80   raise/.code={%
81     \edef\graphicscache@tmp{#1}%
82     \graphicscache@addarg{raise=\graphicscache@tmp}✓
83     %
84   },
85 }
86 \ProcessPgfOptions{/graphicscache}\relax
87 \ifdefined\graphicscache@inhibit
88   \pgfkeys{/graphicscache/render=false}%
89 \fi
90 \ifgraphicscache@listing
91   \newwrite\graphicscache@listout
92   \immediate\openout\graphicscache@listout=\jobname✓
93     .graphicscache
94 \fi

```

`\graphicscache@dorender`

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

```

91 \newcommand{\graphicscache@dorender}{%
92   \message{Rendering \graphicscache@outpuhash: \✓
93     graphicscache@fname\space with args: \✓
94     graphicscache@graphicsargs\space (master file)✓
95   }%
96 }
97 \ifwindows
98   \immediate\write18{md "\graphicscache@cachedir"✓
99     2>NUL}%

```

```

95 \else
96   \immediate\write18{mkdir -p "\graphicscache@cachedir"}%
97 \fi

First, render the graphics.

98 \ifwindows
99   \immediate\write18{del /q \graphicscache@cachedir\string\graphicscacheout.pdf}
100  \immediate\write18{pdflatex
101    -jobname graphicscacheout
102    -interaction nonstopmode
103    -output-directory "\graphicscache@cachedir"
104    "\string\documentclass{standalone}
105    \string\usepackage{graphicx}
106    \string\usepackage[export]{adjustbox}
107    \string\begin{document}\string\includegraphics[\graphicscache@graphicsargs]{\graphicscache@fname}\string\end{document}}%
108  }%
109  \IfFileExists{\graphicscache@cachedir\graphicscacheout.pdf}{\PackageError{graphicscache}{External pdflatex call failed (see above)}}%
110  \def\graphicscache@output{}%
111  }
112 \else
113   \immediate\write18{pdflatex
114     -jobname graphicscacheout
115     -interaction nonstopmode
116     -output-directory "\graphicscache@cachedir"
117     '\string\documentclass{standalone}
118     \string\usepackage{graphicx}
119     \string\usepackage[export]{adjustbox}
120     \string\begin{document}\string\includegraphics[\graphicscache@graphicsargs]{\graphicscache@fname}\string\end{document}}'
121   > /dev/null || rm "\graphicscache@cachedir\graphicscacheout.pdf"
122 }%
123 \fi

```

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

```
125 \ifgraphicscache@compress
126   \message{With compression: \✓
        graphicscache@compress@mode}%
127   \ifwindows
128     \immediate\write18{mgs
129       -sOutputFile=\graphicscache@output\space
130       -sDEVICE=pdfwrite
131       -dCompatibilityLevel=1.4
132       -dPDFSETTINGS=/prepress
133       -dNOPAUSE -dQUIET -dBATCH
134       -c ".setpdfwrite <<
135         /AutoFilterColorImages false
136         /EncodeColorImages true
137         /ColorImageFilter /\✓
            graphicscache@compress@mode\space
138         /ColorImageDict << /ColorTransform 1 /✓
            QFactor \graphicscache@qfactor\space /✓
            Blend 1 /HSamples [1 1 1 1] /VSamples ✓
            [1 1 1 1] >>
139         /ColorImageResolution \graphicscache@dpi\✓
            space
140         /AutoFilterGrayImages false
141         /EncodeGrayImages true
142         /GrayImageFilter /\✓
            graphicscache@compress@mode\space
143         /GrayImageDict << /ColorTransform 1 /✓
            QFactor \graphicscache@qfactor\space /✓
            Blend 1 /HSamples [1 1 1 1] /VSamples ✓
            [1 1 1 1] >>
144         /GrayImageResolution \graphicscache@dpi\✓
            space
145       >> setdistillerparams"
146       -f \graphicscache@cachedir\string\✓
            graphicscacheout.pdf
147   }%
148 \else
149   \immediate\write18{gs
150     -sOutputFile=\graphicscache@output\space
151     -sDEVICE=pdfwrite
152     -dCompatibilityLevel=1.4
153     -dPDFSETTINGS=/prepress
```



```

154     -dNOPAUSE -dQUIET -dBATCH
155     -c '.setpdfwrite <<
156         /AutoFilterColorImages false
157         /EncodeColorImages true
158         /ColorImageFilter /\
159             graphicscache@compress@mode\space
160         /ColorImageDict << /ColorTransform 1 /\
161             QFactor \graphicscache@qfactor\space /\
162             Blend 1 /HSamples [1 1 1 1] /VSamples
163             [1 1 1 1] >>
164         /ColorImageResolution \graphicscache@dpi\
165             space
166         /AutoFilterGrayImages false
167         /EncodeGrayImages true
168         /GrayImageFilter /\
169             graphicscache@compress@mode\space
170         /GrayImageDict << /ColorTransform 1 /\
171             QFactor \graphicscache@qfactor\space /\
172             Blend 1 /HSamples [1 1 1 1] /VSamples
173             [1 1 1 1] >>
174         /GrayImageResolution \graphicscache@dpi\
175             space
176     >> setdistillerparams '
177     -f \graphicscache@cachedir/graphicscacheout\
178         .pdf || rm \graphicscache@output
179     }%
180     \fi
181 \else
182     \message{Direct}%
183     \ifwindows
184         \immediate\write18{
185             copy \graphicscache@cachedir\string\
186                 graphicscacheout.pdf \
187                 graphicscache@output
188         }%
189     \else
190         \immediate\write18{
191             cp \graphicscache@cachedir/graphicscacheout\
192                 .pdf \graphicscache@output
193         }%
194     \fi
195 \fi
196 }

```

```

save original \includegraphics
183 \LetLtxMacro\graphicscache@includegraphics\✓
includegraphics%
184 \newcommand\graphicscache@native{%
185 \expandafter\graphicscache@includegraphics\✓
expandafter[\graphicscache@graphicsargs]{\✓
graphicscache@fname}%
186 }

```

`\graphicscache@work`

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

```

187 \newcommand{\graphicscache@work}{%
188 \ifgraphicscache@render
Check if output file exists and is newer than input
189 \filemodcmp{\graphicscache@fname}{\✓
graphicscache@output}{% input is newer
190 \graphicscache@dorender%
191 }{% Output is newer
192 \message{Already have \✓
graphicscache@outputhash: \✓
graphicscache@fname}%
193 }%

```

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

```

194 \filemodcmp{\graphicscache@fname}{\✓
graphicscache@output}{% input is newer/✓
output does not exist
195 \graphicscache@native
196 }{% otherwise, use the generated file!
197 \graphicscache@includegraphics{\✓
graphicscache@output}%
198 }%
199 \else

```

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

```

200 \IfFileExists{\graphicscache@output}{%
201 \graphicscache@includegraphics{\✓
graphicscache@output}%

```

```

202     }{%
203     \PackageWarning{graphicscache}{Could not find
        cache file \graphicscache@output, for \
        graphicscache@fname, falling back to
        native...}{}%
204     \graphicscache@native
205     }%
206 \fi
207 }

```

\graphicscache@getfname

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

```

208 \catcode'\*=11
209 \newif\ifgraphicscache@exists
210 \newcommand{\graphicscache@getfname}[1]{%
211   \ifx\detokenize\@undefined\else
212     \edef\Gin@extensions{\detokenize\expandafter{\
        Gin@extensions}}%
213   \fi
214   \begingroup
215   \global\graphicscache@existstrue
216   \let\input@path\Gininput@path
217   \ltx@ifpackagelater{graphics}{2017/06/26}{%
218     \set@curr@file{#1}%
219     \expandafter\filename@parse\expandafter{\
        @curr@file}%
220     \ifx\filename@ext\Gin@gzext
221       \expandafter\filename@parse\expandafter{\
        filename@base}%
222       \ifx\filename@ext\relax
223         \let\filename@ext\Gin@gzext
224       \else
225         \edef\Gin@ext{\Gin@ext\Gin@sepdefault\
        Gin@gzext}%
226       \fi
227     \fi
228   }{%
229     \filename@parse{#1}%
230   }%
231   \ifx\filename@ext\relax
232     \@for\Gin@temp:=\Gin@extensions\do{%

```

```

233     \ifx\Gin@ext\relax
234         \Gin@getbase\Gin@temp
235     \fi}%
236 \else
237     \Gin@getbase{\Gin@sepdefault\filename@ext}%
238     \ltx@ifpackagelater{graphics}{2017/06/26}{%
239         \ifx\Gin@ext\relax
240             \let\Gin@savabase\filename@base
241             \let\Gin@savext\filename@ext
242             \edef\filename@base{\filename@base\Gin@sepdefault\filename@ext}%
243             \let\filename@ext\relax
244             \@for\Gin@temp:=\Gin@extensions\do{%
245                 \ifx\Gin@ext\relax
246                     \Gin@getbase\Gin@temp
247                 \fi}%
248             \ifx\Gin@ext\relax
249                 \let\filename@base\Gin@savabase
250                 \let\filename@ext\Gin@savext
251             \fi
252         \fi
253     }{}%
254 \fi
255 \ifx\Gin@ext\relax
256     \global\graphicscache@existsfalse
257 \else
258     \@ifundefined{Gin@rule@\Gin@ext}%
259     {\global\graphicscache@existsfalse}%
260     {}%
261 \fi
262 \ifgraphicscache@exists
263     \xdef\graphicscache@fname{\Gin@base\Gin@ext}%
264 \fi
265 \endgroup
266 }
267 \catcode '\*=12

```

| |
|-------------------------------|
| <code>\includegraphics</code> |
|-------------------------------|

Main entry point.

```

268 \renewcommand{\includegraphics}[2][ ]{%
269     \begingroup
270     \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@outpuhash{\pdf@mdfivesum{\↵  
        graphicscache@options\↵  
        graphicscache@graphicsargs\↵  
        graphicscache@hashedname}}%  
286   \edef\graphicscache@output{\↵  
        graphicscache@cachedir/\↵  
        graphicscache@outpuhash.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 }
```

| |
|------------------------------------|
| <code>\includegraphicscache</code> |
|------------------------------------|

```
297 \newcommand{\includegraphicscache}[3][]{%
298   \begingroup
299   \expandarg
300   \pgfkeys{/graphicscache/.cd,#2}%
301   \includegraphics[#1]{#3}%
302   \endgroup
303 }

304 \endinput
```