

The pdfrender package

Heiko Oberdiek

<heiko.oberdiek at gmail.com>

2010/01/28 v1.2

Abstract

The PDF format has some graphics parameter like line width or text rendering mode. This package provides an interface for setting these parameters.

Contents

1	Documentation	2
1.1	Usage	2
1.2	Macros	2
1.3	Parameters	2
1.3.1	Details	3
1.4	Color stack	3
2	Implementation	4
2.1	Look for pdfTeX, its mode and features	5
2.2	Enable color support of L ^A T _E X	7
2.3	Hook into \normalcolor	7
2.4	Declare and setup parameters	12
2.5	Fill and stroke color support	13
3	Test	17
3.1	Catcode checks for loading	17
3.2	Simple test file	19
3.3	Further tests	19
3.4	Compatibility with plain-T _E X	21
4	Installation	21
4.1	Download	21
4.2	Bundle installation	22
4.3	Package installation	22
4.4	Refresh file name databases	22
4.5	Some details for the interested	22
5	Acknowledgement	23
6	References	23
7	History	23
	[2010/01/26 v1.0]	23
	[2010/01/27 v1.1]	23
	[2010/01/28 v1.2]	24
8	Index	25

1 Documentation

This package `pdfrender` defines an interface for PDF specific parameters that affects the rendering of graphics or text. The interface and its implementation uses the same technique as package `color` for color settings. Therefore this package is loaded to enable L^AT_EX's color interface.

At different places L^AT_EX uses `\normalcolor` to avoid that header, footer or floats are print in the current color of the main text. `\setgroup@color` is used to start a save box with the color that is set at box saving time. Package `pdfrender` extends these macros to add its own hooks of its parameters. Therefore L^AT_EX3 should generalize L^AT_EX 2_ε's color interface.

1.1 Usage

In L^AT_EX the package is loaded as normal package. Options are not defined for this package.

```
\usepackage{pdfrender}
```

This package can also be used in plain-T_EX and even iniT_EX:

```
input pdfrender.sty
```

1.2 Macros

`\pdfrender {⟨key value list⟩}`

The first parameter *⟨key value list⟩* contains a list of parameter settings. The key entry is the parameter name. The macro works like `\color` (without optional argument) for color setting.

`\textpdfrender {⟨key value list⟩} {⟨text⟩}`

In the same way as `\pdfrender` the first argument specifies the parameters that should be set. This parameter setting affects *⟨text⟩* only. Basically it works the same way as `\textcolor` (without optional argument).

1.3 Parameters

The following table shows an overview for the supported parameters and values:

Parameter	Value	Alias
TextRenderingMode	0	Fill
	1	Stroke
	2	FillStroke
	3	Invisible
	4	FillClip
	5	StrokeClip
	6	FillStrokeClip
	7	Clip
LineWidth	<i>positive number, unit is bp</i>	<i>T_EX dimen</i>
LineCapStyle	0	Butt
	1	Round
	2	ProjectingSquare

Parameter	Value	Alias
LineJoinStyle	0	Miter
	1	Round
	2	Bevel
MiterLimit	<i>positive number</i>	
Flatness	<i>number between 0 and 100</i>	
LineDashPattern	<i>numbers in square brackets, followed by number, units are bp</i>	
RenderingIntent	AbsoluteColorimetric RelativeColorimetric Saturation Perceptual	
FillColor		<i>color specification</i>
StrokeColor		<i>color specification</i>

1.3.1 Details

The description and specification of these parameters are available in the PDF specification [1]. Therefore they are not repeated here.

Value: The values in the second column lists or describe the values that are specified by the PDF specification.

Alias: Instead of magic numbers the package also defines some aliases that can be given as value. Example: `LineCapStyle=Round` has the same effect as `LineCapStyle=1`.

Number: The term *number* means an integer or real number. The real number is given as plain decimal number without exponent. The decimal separator is a period. At least one digit must be present.

LineWidth: As alias a \TeX dimen specification can be given. This includes explicit specifications with number and unit, e.g. `LineWidth=0.5pt`. Also \LaTeX length registers may be used. If $\varepsilon\text{-TeX}$'s `\dimexpr` is available, then it is automatically added. However package `calc` is not supported.

FillColor, StrokeColor: Package `color` or `xcolor` must be loaded before these options can be used (since version 1.2). \LaTeX 's color support sets both colors at the same time to the same value. However parameter `TextRenderingMode` offers the value `FillStroke` that makes only sense, if the two color types can be set separately. If one of the options `FillColor` or `StrokeColor` is specified, then also the color is set. For compatibility with the \LaTeX color packages (`color` or `xcolor`), always both colors must be set. Thus if one of them is not specified, it is taken from the current color.

Both options `FillColor` and `StrokeColor` expect a \LaTeX color specification as value. Also the optional color model argument is supported. Example:

```
FillColor=yellow,
StrokeColor=[cmyk]{1,.5,0,0}
```

1.4 Color stack

If the `pdf \TeX` version provides color stacks, then each parameter is assigned a page based color stack. The assignment of a stack takes place, when its parameter is set the first time. This avoids the use of color stacks that are not needed.

2 Implementation

```

1 (*package)
Reload check, especially if the package is not used with LATEX.
2 \begingroup
3   \catcode44 12 % ,
4   \catcode45 12 % -
5   \catcode46 12 % .
6   \catcode58 12 % :
7   \catcode64 11 % @
8   \catcode123 1 % {
9   \catcode125 2 % }
10  \expandafter\let\expandafter\x\csname ver@pdfrender.sty\endcsname
11  \ifx\x\relax % plain-TeX, first loading
12  \else
13    \def\empty{}%
14    \ifx\x\empty % LaTeX, first loading,
15      % variable is initialized, but \ProvidesPackage not yet seen
16    \else
17      \catcode35 6 % #
18      \expandafter\ifx\csname PackageInfo\endcsname\relax
19        \def\x#1#2{%
20          \immediate\write-1{Package #1 Info: #2.}%
21        }%
22      \else
23        \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
24      \fi
25      \x{pdfrender}{The package is already loaded}%
26    \aftergroup\endinput
27  \fi
28 \fi
29 \endgroup
Package identification:
30 \begingroup
31   \catcode35 6 % #
32   \catcode40 12 % (
33   \catcode41 12 % )
34   \catcode44 12 % ,
35   \catcode45 12 % -
36   \catcode46 12 % .
37   \catcode47 12 % /
38   \catcode58 12 % :
39   \catcode64 11 % @
40   \catcode91 12 % [
41   \catcode93 12 % ]
42   \catcode123 1 % {
43   \catcode125 2 % }
44   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
45     \def\x#1#2#3[#4]{\endgroup
46       \immediate\write-1{Package: #3 #4}%
47       \xdef#1{#4}%
48     }%
49   \else
50     \def\x#1#2[#3]{\endgroup
51       #2[#3]}%
52     \ifx#1\@undefined
53       \xdef#1{#3}%
54     \fi
55     \ifx#1\relax
56       \xdef#1{#3}%
57     \fi
58   }%

```

```

59 \fi
60 \expandafter\x\csname ver@pdfrender.sty\endcsname
61 \ProvidesPackage{pdfrender}%
62 [2010/01/28 v1.2 Access to some PDF graphics parameters (H0)]
63 \begingroup
64 \catcode123 1 % {
65 \catcode125 2 % }
66 \def\x{\endgroup
67 \expandafter\edef\csname PdfRender@AtEnd\endcsname{%
68 \catcode35 \the\catcode35\relax
69 \catcode64 \the\catcode64\relax
70 \catcode123 \the\catcode123\relax
71 \catcode125 \the\catcode125\relax
72 }%
73 }%
74 \x
75 \catcode35 6 % #
76 \catcode64 11 % @
77 \catcode123 1 % {
78 \catcode125 2 % }
79 \def\TMP@EnsureCode#1#2{%
80 \edef\PdfRender@AtEnd{%
81 \PdfRender@AtEnd
82 \catcode#1 \the\catcode#1\relax
83 }%
84 \catcode#1 #2\relax
85 }
86 \TMP@EnsureCode{10}{12}% ^^J
87 \TMP@EnsureCode{36}{3}% $
88 \TMP@EnsureCode{39}{12}% '
89 \TMP@EnsureCode{40}{12}% (
90 \TMP@EnsureCode{41}{12}% )
91 \TMP@EnsureCode{42}{12}% *
92 \TMP@EnsureCode{43}{12}% +
93 \TMP@EnsureCode{44}{12}% ,
94 \TMP@EnsureCode{45}{12}% -
95 \TMP@EnsureCode{46}{12}% .
96 \TMP@EnsureCode{47}{12}% /
97 \TMP@EnsureCode{58}{12}% :
98 \TMP@EnsureCode{59}{12}% ;
99 \TMP@EnsureCode{60}{12}% <
100 \TMP@EnsureCode{61}{12}% =
101 \TMP@EnsureCode{62}{12}% >
102 \TMP@EnsureCode{63}{12}% ?
103 \TMP@EnsureCode{91}{12}% [
104 \TMP@EnsureCode{93}{12}% ]
105 \TMP@EnsureCode{94}{7}% ^ (superscript)
106 \TMP@EnsureCode{96}{12}% '
107 \TMP@EnsureCode{124}{12}% |
108 \def\PdfRender@AtEndHook{}
109 \expandafter\def\expandafter\PdfRender@AtEnd\expandafter{%
110 \expandafter\PdfRender@AtEndHook
111 \PdfRender@AtEnd
112 \endinput
113 }

```

2.1 Look for pdfTeX, its mode and features

\PdfRender@newif

```

114 \def\PdfRender@newif#1{%
115 \expandafter\edef\csname PdfRender@#1true\endcsname{%
116 \let
117 \expandafter\noexpand\csname ifPdfRender@#1\endcsname

```

```

118     \noexpand\iftrue
119   }%
120   \expandafter\edef\csname PdfRender@#1false\endcsname{%
121     \let
122     \expandafter\noexpand\csname ifPdfRender@#1\endcsname
123     \noexpand\iffalse
124   }%
125   \csname PdfRender@#1false\endcsname
126 }

```

\ifPdfRender@Stack

```

127 \PdfRender@newif{Stack}

```

\ifPdfRender@Match

```

128 \PdfRender@newif{Match}

```

\PdfRender@RequirePackage

```

129 \begingroup\expandafter\expandafter\expandafter\endgroup
130 \expandafter\ifx\csname RequirePackage\endcsname\relax
131   \def\PdfRender@RequirePackage#1[#2]{%
132     \expandafter\def\expandafter\PdfRender@AtEndHook\expandafter{%
133       \PdfRender@AtEndHook
134       \ltx@ifpackagelater{#1}{#2}{-}{%
135         \@PackageWarningNoLine{pdfrender}{%
136           You have requested version\MessageBreak
137           ‘#2’ of package ‘#1’,\MessageBreak
138           but only version\MessageBreak
139           ‘\csname ver@#1.\ltx@pkgextension\endcsname’\MessageBreak
140           is available%
141         }%
142       }%
143     }%
144     \input #1.sty\relax
145   }%
146 \else
147   \let\PdfRender@RequirePackage\RequirePackage
148 \fi

149 \PdfRender@RequirePackage{ifpdf}[2010/01/28]
150 \PdfRender@RequirePackage{infwarerr}[2007/09/09]
151 \PdfRender@RequirePackage{ltxcmds}[2010/01/28]

152 \ifpdf
153   \ltx@ifundefined{pdfcolorstackinit}{%
154     \@PackageWarning{pdfrender}{%
155       Missing \string\pdfcolorstackinit
156     }%
157   }{%
158     \PdfRender@Stacktrue
159   }%
160   \ltx@ifundefined{pdfmatch}{%
161     \@PackageInfoNoLine{pdfrender}{%
162       \string\pdfmatch\ltx@space not found. %
163       Therefore the values\MessageBreak
164       of some parameters are not validated%
165     }%
166   }{%
167     \PdfRender@Matchtrue
168   }%
169 \else
170   \@PackageWarning{pdfrender}{%
171     Missing pdfTeX in PDF mode%
172   }%

```

```

173 \ltx@ifundefined{newcommand}{%
\pdfrender
174 \def\pdfrender#1{%
\textpdfrender
175 \long\def\textpdfrender#1#2{#2}%
176 }{%
\pdfrender
177 \newcommand*\pdfrender[1]{}%
\textpdfrender
178 \newcommand{\textpdfrender}[2]{#2}%
179 }%
180 \expandafter\PdfRender@AtEnd
181 \fi

```

2.2 Enable color support of L^AT_EX

```

182 \ltx@ifpackageloaded{color}{}%
183 \def\color@setgroup{\begingroup\set@color}%
184 \let\color@begingroup\begingroup
185 \def\color@endgroup{\endgraf\endgroup}%
186 \def\color@hbox{\hbox\bgroup\color@begingroup}%
187 \def\color@vbox{\vbox\bgroup\color@begingroup}%
188 \def\color@endbox{\color@endgroup\egroup}%
189 \ltx@ifundefined{bgroup}{%
190 \let\bgroup={\let\egroup=}
191 }{}%
192 \ltx@ifundefined{endgraf}{%
193 \let\endgraf=\par
194 }{}%
195 }

```

2.3 Hook into \normalcolor

The problem is that packages `color` and `xcolor` each overwrite `\normalcolor`. For example, after the package loading order `color`, `pdfrender` and `xcolor` the patched version of `\normalcolor` is overwritten by package `xcolor`. Also using `\AtBeginDocument` for patching is not enough. If package `hyperref` is loaded later, it might load package `color` using `\AtBeginDocument`.

```
\PdfRender@NormalColorHook
```

```
196 \def\PdfRender@NormalColorHook{}
```

```
\PdfRender@ColorSetGroupHook
```

```
197 \def\PdfRender@ColorSetGroupHook{}
```

```
\PdfRender@TestBox
```

```

198 \def\PdfRender@TestBox#1{%
199 \setbox0=\color@hbox#1\color@endbox
200 }

```

```
\PdfRender@PatchNormalColor
```

```

201 \def\PdfRender@PatchNormalColor{%
202 \ltx@ifundefined{normalcolor}{%
203 \gdef\normalcolor{\PdfRender@NormalColorHook}%
204 }{%

```

```

205 \begingroup
206 \def\PdfRender@NormalColorHook{\let\PdfRender@temp=Y}%
207 \PdfRender@TestBox{%
208 \let\set@color\relax
209 \normalcolor
210 \ifx\PdfRender@temp Y%
211 \else
212 \ltx@GlobalAppendToMacro\normalcolor{%
213 \PdfRender@NormalColorHook
214 }%
215 \fi
216 }%
217 \endgroup
218 }%
219 \ifx\@nodocument\relax
220 \global\let\PdfRender@PatchNormalColor\relax
221 \fi
222 }%

```

\PdfRender@PatchColorSetGroup

```

223 \def\PdfRender@PatchColorSetGroup{%
224 \begingroup
225 \def\PdfRender@ColorSetGroupHook{\let\PdfRender@temp=Y}%
226 \PdfRender@TestBox{%
227 \let\set@color\relax
228 \color@setgroup\color@endgroup
229 \ifx\PdfRender@temp Y%
230 \else
231 \ltx@GlobalAppendToMacro\color@setgroup{%
232 \PdfRender@ColorSetGroupHook
233 }%
234 \fi
235 }%
236 \endgroup
237 \ifx\@nodocument\relax
238 \global\let\PdfRender@PatchColorSetGroup\relax
239 \fi
240 }%

```

\PdfRender@PatchColor

```

241 \def\PdfRender@PatchColor{%
242 \PdfRender@PatchNormalColor
243 \PdfRender@PatchColorSetGroup
244 }
245 % \begin{macrocode}
246 \PdfRender@PatchColor
247 \ltx@ifundefined{AtBeginDocument}{-}{%
248 \AtBeginDocument{\PdfRender@PatchColor}%
249 }

```

\AfterPackage is provided by package scrfile.

```

250 \ltx@ifundefined{AfterPackage}{-}{%
251 }{%
252 \AfterPackage{color}{\PdfRender@PatchColor}%
253 \AfterPackage{xcolor}{\PdfRender@PatchColor}%
254 \AfterPackage{etoolbox}{-}{%
255 \AfterEndPreamble{\PdfRender@PatchColor}%
256 }%
257 }%

```

\AfterEndPreamble is provided by package etoolbox.

```

258 \ltx@ifundefined{AfterEndPreamble}{-}{%
259 }{-}%

```



```

260 \AfterEndPreamble{\PdfRender@PatchColor}%
261 }%

262 \PdfRender@RequirePackage{kvsetkeys}[2010/01/28]

```

\PdfRender@texorpdfstring

```

263 \def\PdfRender@texorpdfstring{%
264   \ltx@ifundefined{texorpdfstring}\ltx@firstoftwo\texorpdfstring
265 }

```

\pdfrender

```

266 \ltx@ifundefined{DeclareRobustCommand}%
267 \ltx@firstoftwo\ltx@secondoftwo
268 {%
269   \def\pdfrender#1%
270 }{%
271   \newcommand{\pdfrender}{}%
272   \DeclareRobustCommand*\pdfrender[1]%
273 }%
274 {%
275   \PdfRender@texorpdfstring{%
276     \PdfRender@PatchNormalColor
277     \global\let\PdfRender@FillColor\ltx@empty
278     \global\let\PdfRender@StrokeColor\ltx@empty
279     \kvsetkeys{PDFRENDER}{#1}%
280     \PdfRender@SetColor
281   }{}%
282 }

```

\textpdfrender

```

283 \ltx@ifundefined{DeclareRobustCommand}%
284 \ltx@firstoftwo\ltx@secondoftwo
285 {%
286   \long\def\textpdfrender#1#2%
287 }{%
288   \newcommand{\textpdfrender}{}%
289   \DeclareRobustCommand{\textpdfrender}[2]%
290 }%
291 {%
292   \PdfRender@texorpdfstring{%
293     \begingroup
294       \pdfrender{#1}%
295       #2%
296     \endgroup
297   }{#2}%
298 }

```

\ifPdfRender@Values

```

299 \PdfRender@newif{Values}

```

\PdfRender@NewClassValues

```

300 \def\PdfRender@NewClassValues#1#2#3#4{%
301   \PdfRender@Valuestrue
302   \PdfRender@NewClass{#1}{#2}{#3}{#4}{}%
303 }

```

\PdfRender@NewClass

```

304 \def\PdfRender@NewClass#1#2#3#4#5{%
305   \PdfRender@newif{Active#1}%
306   \expandafter\def\csname PdfRender@Default#1\endcsname{#2}%
307   \expandafter\let\csname PdfRender@Current#1\endcsname

```

```

308     \csname PdfRender@Default#1\endcsname
309 \ifPdfRender@Stack
310     \expandafter\edef\csname PdfRender@Init#1\endcsname{%
311         \global\chardef
312         \expandafter\noexpand\csname PdfRender@Stack#1\endcsname=%
313             \noexpand\pdfcolorstackinit page direct{%
314                 \noexpand#3%
315             \expandafter\noexpand\csname PdfRender@Default#1\endcsname
316         }\relax
317     \noexpand\@PackageInfo{pdfrender}{%
318         New color stack '#1' = \noexpand\number
319         \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
320     }%
321     \gdef\expandafter\noexpand\csname PdfRender@Init#1\endcsname{}}%
322 }%
323 \expandafter\edef\csname PdfRender@Set#1\endcsname{%
324     \expandafter\noexpand\csname PdfRender@Init#1\endcsname
325     \noexpand\pdfcolorstack
326     \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
327     push{%
328         #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
329     }%
330     \noexpand\aftergroup
331     \expandafter\noexpand\csname PdfRender@Reset#1\endcsname
332 }%
333 \expandafter\edef\csname PdfRender@Reset#1\endcsname{%
334     \expandafter\noexpand\csname PdfRender@Init#1\endcsname
335     \noexpand\pdfcolorstack
336     \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
337     pop\relax
338 }%
339 \else
340     \expandafter\edef\csname PdfRender@Set#1\endcsname{%
341         \noexpand\pdfliteral direct{%
342             #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
343         }%
344         \noexpand\aftergroup
345         \expandafter\noexpand\csname PdfRender@Reset#1\endcsname
346     }%
347     \expandafter\edef\csname PdfRender@Reset#1\endcsname{%
348         \noexpand\pdfliteral direct{%
349             #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
350         }%
351     }%
352 \fi
353 \expandafter\edef\csname PdfRender@Normal#1\endcsname{%
354     \let
355     \expandafter\noexpand\csname PdfRender@Current#1\endcsname
356     \expandafter\noexpand\csname PdfRender@Default#1\endcsname
357     \noexpand\PdfRender@Set{#1}%
358 }%
359 \expandafter\ltx@GlobalAppendToMacro\expandafter\PdfRender@NormalColorHook
360 \expandafter{%
361     \csname PdfRender@Normal#1\endcsname
362 }%
363 \ltx@GlobalAppendToMacro\PdfRender@ColorSetGroupHook{%
364     \PdfRender@Set{#1}%
365 }%
366 \ifPdfRender@Values
367     \kv@parse@normalized{#4}{%
368         \expandafter\let\csname PdfRender@#1@\kv@key\endcsname\kv@key
369         \ifx\kv@value\relax

```

```

370     \else
371     \expandafter\let\csname PdfRender@#1@\kv@value\endcsname\kv@key
372     \fi
373     \ltx@gobbletwo
374 }%
375 \PdfRender@define@key{PDFRENDER}{#1}{%
376     \global\csname PdfRender@Active#1true\endcsname
377     \def\PdfRender@Current{##1}%
378     \PdfRender@SetValidateValues{#1}%
379 }%
380 \PdfRender@Valuesfalse
381 \else
382 \PdfRender@define@key{PDFRENDER}{#1}{%
383     \global\csname PdfRender@Active#1true\endcsname
384     \expandafter\def\csname PdfRender@Current#1\endcsname{##1}%
385     \ltx@ifundefined{PdfRender@PostProcess#1}{%
386     }{%
387     \csname PdfRender@PostProcess#1\endcsname
388     }%
389     \PdfRender@SetValidate{#1}{#4}{#5}%
390 }%
391 \fi
392 }%

```

\PdfRender@define@key

```

393 \ltx@ifundefined{define@key}{%
394     \def\PdfRender@define@key#1#2{%
395     \expandafter\def\csname KV@#1@#2\endcsname##1%
396     }%
397 }{%
398     \let\PdfRender@define@key\define@key
399 }

```

\PdfRender@Set

```

400 \def\PdfRender@Set#1{%
401     \csname ifPdfRender@Active#1\endcsname
402     \csname PdfRender@Set#1\expandafter\endcsname
403     \fi
404 }

```

\PdfRender@Reset

```

405 \def\PdfRender@Reset#1{%
406     \csname ifPdfRender@Active#1\endcsname
407     \csname PdfRender@Reset#1\expandafter\endcsname
408     \fi
409 }

```

\PdfRender@ErrorInvalidValue

```

410 \def\PdfRender@ErrorInvalidValue#1{%
411     \PackageError{pdfrender}{%
412     Ignoring parameter setting for ‘#1’\MessageBreak
413     because of invalid value %
414     ‘\csname PdfRender@Current#1\endcsname’%
415     }{@ehc
416     \expandafter\let\csname PdfRender@Current#1\endcsname\ltx@empty
417 }%

```

\PdfRender@SetValidate

```

418 \ifPdfRender@Match
419 \def\PdfRender@SetValidate#1#2#3{%
420     \ifnum\pdfmatch{^(#2)$}\csname PdfRender@Current#1\endcsname}=1 %

```

```

421     \csname PdfRender@Set#1\expandafter\endcsname
422     \else
423     \PdfRender@ErrorInvalidValue{#1}%
424     \fi
425 }%
426 \else
427 \def\PdfRender@SetValidate#1#2#3{%
428     \expandafter\let\expandafter\PdfRender@Current
429     \csname PdfRender@Current#1\endcsname
430     #3%
431     \ifx\PdfRender@Current\@empty
432     \PdfRender@ErrorInvalidValue{#1}%
433     \else
434     \csname PdfRender@Set#1\expandafter\endcsname
435     \fi
436 }%
437 \fi

```

\PdfRender@SetValidateValues

```

438 \def\PdfRender@SetValidateValues#1{%
439     \ltx@ifundefined{PdfRender@#1@\PdfRender@Current}{%
440         \expandafter\let\csname PdfRender@Current#1\endcsname
441             \PdfRender@Current
442         \PdfRender@ErrorInvalidValue{#1}%
443     }{%
444         \expandafter\edef\csname PdfRender@Current#1\endcsname{%
445             \csname PdfRender@#1@\PdfRender@Current\endcsname
446         }%
447         \csname PdfRender@Set#1\endcsname
448     }%
449 }

```

\PdfRender@OpValue

```

450 \def\PdfRender@OpValue#1#2{#2\ltx@space#1}%

```

\PdfRender@OpName

```

451 \def\PdfRender@OpName#1#2{/#2\ltx@space#1}%

```

2.4 Declare and setup parameters

```

452 \PdfRender@NewClassValues{TextRenderingMode}%
453     {0}%
454     {\PdfRender@OpValue{Tr}}}%
455 0=Fill,%
456 1=Stroke,%
457 2=FillStroke,%
458 3=Invisible,%
459 4=FillClip,%
460 5=StrokeClip,%
461 6=FillStrokeClip,%
462 7=Clip,%
463 }%
464 \PdfRender@NewClass{LineWidth}{1}{\PdfRender@OpValue{w}}{%
465     [0-9]+\string\.[0-9]*\string\.[0-9]+%
466 }{%
467 \ltx@ifundefined{dimexpr}{%
468     \def\PdfRender@dimexpr{%
469     }{%
470     \let\PdfRender@dimexpr\dimexpr
471     }
472 \def\PdfRender@PostProcessLineWidth{%
473     \begingroup

```

```

474 \afterassignment\PdfRender@@PostProcessLineWidth
475 \dimen0=\PdfRender@dimexpr\PdfRender@CurrentLineWidth bp %
476 \PdfRender@let\PdfRender@relax\PdfRender@relax
477 }
478 \let\PdfRender@let\let
479 \let\PdfRender@relax\relax
480 \def\PdfRender@@PostProcessLineWidth#1\PdfRender@let{%
481   \ifx\#1\%
482     \endgroup
483   \else
484     \dimen0=.996264\dimen0 % 72/72.27
485     \edef\x{\endgroup
486       \def\noexpand\PdfRender@CurrentLineWidth{%
487         \strip@pt\dimen0%
488       }%
489     }%
490     \expandafter\x
491   \fi
492 }
493 \PdfRender@NewClassValues{LineCapStyle}{0}{\PdfRender@OpValue{J}}{%
494   0=Butt,%
495   1=Round,%
496   2=ProjectingSquare,%
497 }%
498 \PdfRender@NewClassValues{LineJoinStyle}{0}{\PdfRender@OpValue{j}}{%
499   0=Miter,%
500   1=Round,%
501   2=Bevel,%
502 }%
503 \PdfRender@NewClass{MiterLimit}{10}{\PdfRender@OpValue{M}}{%
504   [0-9]*[1-9][0-9]*\string\.[0-9]*|
505   [0-9]*\string\.[0-9]*[1-9][0-9]*%
506 }{}%
507 \PdfRender@NewClass{Flatness}{0}{\PdfRender@OpValue{i}}{%
508   100(\string\.[0-9]*)?|[0-9][0-9](\string\.[0-9]*)?\string\.[0-9]+%
509 }{}%
510 \PdfRender@NewClass{LineDashPattern}{[]0}{\PdfRender@OpValue{d}}{%
511   \string\[%
512   ( ?([0-9]+\string\.[0-9]*|\string\.[0-9]+) ?)*%
513   \string\] ?%
514   ([0-9]+\string\.[0-9]*|\string\.[0-9]+)%
515 }{}%
516 \PdfRender@NewClassValues{RenderingIntent}%
517   {RelativeColorimetric}%
518   {\PdfRender@OpName{ri}}{%
519   AbsoluteColorimetric,%
520   RelativeColorimetric,%
521   Saturation,%
522   Perceptual,%
523 }%

```

2.5 Fill and stroke color support

```

524 \PdfRender@define@key{PDFRENDER}{FillColor}{%
525   \begingroup
526     \def\PdfRender@Color{#1}%
527     \ifx\PdfRender@Color\ltx@empty
528       \global\let\PdfRender@FillColor\ltx@empty
529     \else
530       \PdfRender@ColorAvailable{%
531         \PdfRender@TestBox{%
532           \expandafter\PdfRender@TryColor\PdfRender@Color\ltx@empty
533         \PdfRender@GetFillColor

```

```

534         \ifx\PdfRender@FillColor\ltx@empty
535         \@PackageWarning{pdfrender}{%
536             Cannot extract fill color\MessageBreak
537             from value '#1'%
538         }%
539         \fi
540     }%
541 }%
542 \fi
543 \endgroup
544 }
545 \PdfRender@define@key{PDFRENDER}{StrokeColor}{%
546     \begingroup
547     \def\PdfRender@Color{#1}%
548     \ifx\PdfRender@Color\ltx@empty
549         \global\let\PdfRender@StrokeColor\ltx@empty
550     \else
551         \PdfRender@ColorAvailable{%
552             \PdfRender@TestBox{%
553                 \expandafter\PdfRender@TryColor\PdfRender@Color\ltx@empty
554                 \PdfRender@GetStrokeColor
555                 \ifx\PdfRender@StrokeColor\ltx@empty
556                     \@PackageWarning{pdfrender}{%
557                         Cannot extract stroke color\MessageBreak
558                         from value '#1'%
559                     }%
560                 \fi
561             }%
562         }%
563     \fi
564 \endgroup
565 }

```

\PdfRender@ColorAvailable

```

566 \def\PdfRender@ColorAvailable{%
567     \@ifundefined{set@color}{%
568         \@PackageError{pdfrender}{%
569             Ignoring color options, because neither\MessageBreak
570             package 'color' nor package 'xcolor' is loaded%
571         }\@ehc
572     \global\let\PdfRender@ColorAvailable\ltx@gobble
573 }{%
574     \global\let\PdfRender@ColorAvailable\ltx@firstofone
575 }%
576 \PdfRender@ColorAvailable
577 }

```

\PdfRender@TryColor

```

578 \def\PdfRender@TryColor{%
579     \@ifnextchar[\color\PdfRender@@TryColor
580 }

```

\PdfRender@@TryColor

```

581 \def\PdfRender@@TryColor#1\ltx@empty{%
582     \expandafter\color\expandafter{\PdfRender@Color}%
583 }

```

\PdfRender@SetColor

```

584 \def\PdfRender@SetColor{%
585     \chardef\PdfRender@NeedsCurrentColor=0 %
586     \ifx\PdfRender@FillColor\ltx@empty
587         \ifx\PdfRender@StrokeColor\ltx@empty

```

```

588 \else
589 \edef\PdfRender@CurrentColor{%
590 \noexpand\PdfRender@FillColor\ltx@space\PdfRender@StrokeColor
591 }%
592 \chardef\PdfRender@NeedsCurrentColor=1 %
593 \fi
594 \else
595 \ifx\PdfRender@StrokeColor\ltx@empty
596 \edef\PdfRender@CurrentColor{%
597 \PdfRender@FillColor\ltx@space\noexpand\PdfRender@StrokeColor
598 }%
599 \chardef\PdfRender@NeedsCurrentColor=2 %
600 \else
601 \edef\current@color{%
602 \PdfRender@FillColor\ltx@space\PdfRender@StrokeColor
603 }%
604 \set@color
605 \fi
606 \fi
607 \ifnum\PdfRender@NeedsCurrentColor=1 %
608 \PdfRender@GetFillColor
609 \ifx\PdfRender@FillColor\ltx@empty
610 \@PackageWarning{pdfrender}{%
611 Cannot extract current fill color%
612 }%
613 \else
614 \edef\current@color{\PdfRender@CurrentColor}%
615 \set@color
616 \fi
617 \else
618 \ifnum\PdfRender@NeedsCurrentColor=2 %
619 \PdfRender@GetStrokeColor
620 \ifx\PdfRender@StrokeColor\ltx@empty
621 \@PackageWarning{pdfrender}{%
622 Cannot extract current stroke color%
623 }%
624 \else
625 \edef\current@color{\PdfRender@CurrentColor}%
626 \set@color
627 \fi
628 \fi
629 \fi
630 }

```

\PdfRender@PatternFillColor

```

631 \edef\PdfRender@PatternFillColor{ % space
632 (%)
633 [0-9\string\.] + g|%
634 [0-9\string\.] + [0-9\string\.] + [0-9\string\.] + rg|%
635 [0-9\string\.] + [0-9\string\.] + %
636 [0-9\string\.] + [0-9\string\.] + k%
637 ) % space
638 (.* )$%
639 }

```

\PdfRender@PatternStrokeColor

```

640 \edef\PdfRender@PatternStrokeColor{ % space
641 (%)
642 [0-9\string\.] + G|%
643 [0-9\string\.] + [0-9\string\.] + [0-9\string\.] + RG|%
644 [0-9\string\.] + [0-9\string\.] + %
645 [0-9\string\.] + [0-9\string\.] + K%

```

```

646 ) % space
647 (.*)$%
648 }

```

\PdfRender@MatchPattern

```

649 \def\PdfRender@MatchPattern#1{%
650   \ifnum\pdfmatch{\PdfRender@Pattern}{\PdfRender@String}=1 %
651     \xdef#1{%
652       \expandafter\strip@prefix\pdflastmatch 1%
653     }%
654   \edef\PdfRender@String{%
655     \expandafter\strip@prefix\pdflastmatch 2%
656   }%
657   \ifx\PdfRender@String\ltx@empty
658   \else
659     \expandafter\expandafter\expandafter\PdfRender@MatchPattern
660     \expandafter\expandafter\expandafter#1%
661   \fi
662 \fi
663 }

```

\PdfRender@GetFillColor

```

664 \def\PdfRender@GetFillColor{%
665   \global\let\PdfRender@FillColor\ltx@empty
666   \begingroup
667     \ifPdfRender@Match
668       \let\PdfRender@Pattern\PdfRender@PatternFillColor
669       \edef\PdfRender@String{\ltx@space\current@color\ltx@space}%
670       \PdfRender@MatchPattern\PdfRender@FillColor
671     \else
672       \edef\current@color{\current@color\ltx@space}%
673       \let\PdfRender@OP\relax
674       \PdfRender@FindOp{g}0%
675       \PdfRender@FindOp{G}1%
676       \PdfRender@FindOp{rg}0%
677       \PdfRender@FindOp{RG}1%
678       \PdfRender@FindOp{k}0%
679       \PdfRender@FindOp{K}1%
680       \PdfRender@FilterOp 0\PdfRender@FillColor
681     \fi
682   \endgroup
683 }

```

\PdfRender@GetStrokeColor

```

684 \def\PdfRender@GetStrokeColor{%
685   \global\let\PdfRender@StrokeColor\ltx@empty
686   \begingroup
687     \ifPdfRender@Match
688       \let\PdfRender@Pattern\PdfRender@PatternStrokeColor
689       \edef\PdfRender@String{\ltx@space\current@color\ltx@space}%
690       \PdfRender@MatchPattern\PdfRender@StrokeColor
691     \else
692       \edef\current@color{\current@color\ltx@space}%
693       \let\PdfRender@OP\relax
694       \PdfRender@FindOp{g}0%
695       \PdfRender@FindOp{G}1%
696       \PdfRender@FindOp{rg}0%
697       \PdfRender@FindOp{RG}1%
698       \PdfRender@FindOp{k}0%
699       \PdfRender@FindOp{K}1%
700       \PdfRender@FilterOp 1\PdfRender@StrokeColor
701     \fi

```



```

702 \endgroup
703 }

704 \ifPdfRender@Match
705 \expandafter\PdfRender@AtEnd
706 \fi

\PdfRender@FindOp

707 \def\PdfRender@FindOp#1#2{%
708 \def\PdfRender@temp##1 #1 ##2\@nil{%
709 ##1%
710 \ifx\\##2\\%
711 \expandafter\@gobble
712 \else
713 \PdfRender@OP{#1}#2%
714 \expandafter\@firstofone
715 \fi
716 {%
717 \PdfRender@temp##2\@nil
718 }%
719 }%
720 \edef\current@color{%
721 \@firstofone{\expandafter\PdfRender@temp\current@color} #1 \@nil
722 }%
723 }

\PdfRender@FilterOp

724 \def\PdfRender@FilterOp#1#2{%
725 \expandafter\PdfRender@@FilterOp\expandafter#1\expandafter#2%
726 \current@color\PdfRender@OP{#1}{#2}%
727 }

\PdfRender@@FilterOp

728 \def\PdfRender@@FilterOp#1#2#3\PdfRender@OP#4#5{%
729 \ifx\\#4#5\\%
730 \else
731 \ifnum#1=#5 %
732 \xdef#2{#3 #4}%
733 \fi
734 \expandafter\PdfRender@@FilterOp\expandafter#1\expandafter#2%
735 \fi
736 }

737 \PdfRender@AtEnd
738 \end{package}

```

3 Test

3.1 Catcode checks for loading

```

739 \test1
740 \catcode'\{=1 %
741 \catcode'\}=2 %
742 \catcode'\#=6 %
743 \catcode'\@=11 %
744 \expandafter\ifx\csname count@\endcsname\relax
745 \countdef\count@=255 %
746 \fi
747 \expandafter\ifx\csname @gobble\endcsname\relax
748 \long\def\@gobble#1{%
749 \fi

```

```

750 \expandafter\ifx\csname @firstofone\endcsname\relax
751   \long\def\@firstofone#1{#1}%
752 \fi
753 \expandafter\ifx\csname loop\endcsname\relax
754   \expandafter\@firstofone
755 \else
756   \expandafter\@gobble
757 \fi
758 {%
759   \def\loop#1\repeat{%
760     \def\body{#1}%
761     \iterate
762   }%
763   \def\iterate{%
764     \body
765     \let\next\iterate
766   \else
767     \let\next\relax
768   \fi
769   \next
770 }%
771 \let\repeat=\fi
772 }%
773 \def\RestoreCatcodes{}
774 \count@=0 %
775 \loop
776   \edef\RestoreCatcodes{%
777     \RestoreCatcodes
778     \catcode\the\count@=\the\catcode\count@\relax
779   }%
780 \ifnum\count@<255 %
781   \advance\count@ 1 %
782 \repeat
783
784 \def\RangeCatcodeInvalid#1#2{%
785   \count@=#1\relax
786   \loop
787     \catcode\count@=15 %
788     \ifnum\count@<#2\relax
789       \advance\count@ 1 %
790     \repeat
791 }
792 \expandafter\ifx\csname LoadCommand\endcsname\relax
793   \def\LoadCommand{\input pdfrender.sty\relax}%
794 \fi
795 \def\Test{%
796   \RangeCatcodeInvalid{0}{47}%
797   \RangeCatcodeInvalid{58}{64}%
798   \RangeCatcodeInvalid{91}{96}%
799   \RangeCatcodeInvalid{123}{255}%
800   \catcode'\@=12 %
801   \catcode'\=0 %
802   \catcode'\{=1 %
803   \catcode'\}=2 %
804   \catcode'\#=6 %
805   \catcode'\[=12 %
806   \catcode'\]=12 %
807   \catcode'\%=14 %
808   \catcode'\ =10 %
809   \catcode13=5 %
810   \LoadCommand
811   \RestoreCatcodes

```

```

812 }
813 \Test
814 \csname @@end\endcsname
815 \end
816 </test1>

```

3.2 Simple test file

```

817 (*test2)
818 \NeedsTeXFormat{LaTeX2e}
819 \ProvidesFile{pdfrender-test2.tex}[2010/01/28]
820 \documentclass{article}
821 \usepackage{color}
822 \usepackage{pdfrender}[2010/01/28]
823 \begin{document}
824 Hello World
825 \newpage
826 Start
827 \textpdfrender{%
828   TextRenderingMode=1,%
829   LineWidth=.1,%
830   LineCapStyle=2,%
831   LineJoinStyle=1,%
832   MiterLimit=1.2,%
833   LineDashPattern=[2 2]0,%
834   RenderingIntent=Saturation,%
835 }{Hello\newpage World}
836 Stop
837 \par
838 \newlength{\LineWidth}
839 \setlength{\LineWidth}{.5pt}
840 Start
841 \textpdfrender{%
842   FillColor=yellow,%
843   StrokeColor=[cmyk]{1,.5,0,0},%
844   TextRenderingMode=FillStroke,%
845   LineWidth=.5\LineWidth,%
846   LineCapStyle=Round,%
847   LineJoinStyle=Bevel,%
848 }{Out-\par\newpage line}
849 Stop
850 \end{document}
851 </test2>

```

3.3 Further tests

Robustness and bookmarks.

```

852 (*test3)
853 \NeedsTeXFormat{LaTeX2e}
854 \ProvidesFile{pdfrender-test3.tex}[2010/01/28]
855 \documentclass{article}
856 \usepackage{pdfrender}[2010/01/28]
857 \usepackage{hyperref}
858 \usepackage{bookmark}
859 \begin{document}
860 \tableofcontents
861 \section{%
862   \textpdfrender{%
863     TextRenderingMode=1,%
864     LineCapStyle=2,%
865     LineJoinStyle=1,%
866     MiterLimit=1.2,%
867     LineDashPattern=[2 2]0,%

```

```

868     RenderingIntent=Saturation,%
869   }{Hello World}%
870 }
871 \end{document}
872 </test3>

    Color algorithm if \pdfmatch is not available.

873 <*test4>
874 \NeedsTeXFormat{LaTeX2e}
875 \ProvidesFile{pdfrender-test4.tex}[2010/01/28]
876 \documentclass[12pt]{article}
877 \usepackage{pdfrender}[2010/01/28]
878 \usepackage{color}
879 \usepackage{qstest}
880 \IncludeTests{*}
881 \LogTests{log}{*}{*}
882 \makeatletter
883 \newcommand*{\CheckColor}[1]{%
884   \Expect{#1}*{\current@color}%
885 }
886 \makeatother
887 \begin{document}
888   \begin{qstest}{color}{color}%
889     \CheckColor{0 g 0 G}%
890     \Huge\bfseries
891     \noindent
892     \textpdfrender{%
893       TextRenderingMode=2,%
894       LineWidth=.5,%
895       FillColor=yellow,%
896       StrokeColor=blue,%
897     }{%
898       \CheckColor{0 0 1 0 k 0 0 1 RG}%
899       Blue(Yellow)\%
900       \textpdfrender{%
901         FillColor=green,%
902       }{%
903         \CheckColor{0 1 0 rg 0 0 1 RG}%
904         Blue(Green)%
905       }\%
906       \CheckColor{0 0 1 0 k 0 0 1 RG}%
907       Blue(Yellow)\%
908       \textpdfrender{%
909         StrokeColor=red,%
910       }{%
911         \CheckColor{0 0 1 0 k 1 0 0 RG}%
912         Red(Yellow)%
913       }\%
914       \CheckColor{0 0 1 0 k 0 0 1 RG}%
915       Blue(Yellow) %
916     }%
917   \end{qstest}%
918   \begin{qstest}{colorlast}{colorlast}%
919     \makeatletter
920     \def\Test#1#2#3{%
921       \begingroup
922         \def\current@color{#1}%
923         \textpdfrender{#2}{%
924           \CheckColor{#3}%
925         }%
926       \endgroup
927     }%
928     \Test{1 g 0 0 1 RG 0 0 1 0 k 0.5 G}%

```

```

929      {StrokeColor=green}%
930      {0 0 1 0 k 0 1 0 RG}%
931      \Test{1 g 0 0 1 RG 0 0 1 0 k 0.5 G}%
932      {FillColor=red}%
933      {1 0 0 rg 0.5 G}%
934    \end{qstest}%
935 \end{document}
936 </test4>

```

3.4 Compatibility with plain-TeX

```

937 <*test5>
938 \pdfoutput=1 %
939 \hsize=6.5in
940 \vsize=8.9in
941 \pdfpagewidth=\hsize
942 \pdfpageheight=\vsize
943 \parfillskip=0pt plus 1fil\relax
944 \input pdfrender.sty\relax
945 \catcode'\{=1 %
946 \catcode'\}=2 %
947 \let\OrgMakeFootLine\makefootline
948 \def\makefootline{%
949   \begingroup\normalcolor\OrgMakeFootLine\endgroup
950 }
951 \font\font=ec-lmr10 scaled 3000\relax
952 \font
953 Before %
954 \textpdfrender{%
955   TextRenderingMode=1,%
956   LineWidth=.1,%
957 }{Hello\par\vfill\penalty-10000 World} %
958 After %
959 \par
960 \vfill
961 \penalty-10000 %
962 \csname @@end\endcsname\end
963 </test5>

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

[CTAN:macros/latex/contrib/oberdiek/pdfrender.dtx](#) The source file.

[CTAN:macros/latex/contrib/oberdiek/pdfrender.pdf](#) Documentation.

Bundle. All the packages of the bundle ‘oberdiek’ are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

[CTAN:install/macros/latex/contrib/oberdiek.tds.zip](#)

TDS refers to the standard “A Directory Structure for TeX Files” ([CTAN:tds/tds.pdf](#)). Directories with `texmf` in their name are usually organized this way.

¹[ftp://ftp.ctan.org/tex-archive/](http://ftp.ctan.org/tex-archive/)

4.2 Bundle installation

Unpacking. Unpack the `oberdiek.tds.zip` in the TDS tree (also known as `texmf` tree) of your choice. Example (linux):

```
unzip oberdiek.tds.zip -d ~/texmf
```

Script installation. Check the directory `TDS:scripts/oberdiek/` for scripts that need further installation steps. Package `attachfile2` comes with the Perl script `pdfatfi.pl` that should be installed in such a way that it can be called as `pdfatfi`. Example (linux):

```
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
```

4.3 Package installation

Unpacking. The `.dtx` file is a self-extracting `docstrip` archive. The files are extracted by running the `.dtx` through plain- $\mathrm{T}_{\mathrm{E}}\mathrm{X}$:

```
tex pdfrender.dtx
```

TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

<code>pdfrender.sty</code>	\rightarrow <code>tex/generic/oberdiek/pdfrender.sty</code>
<code>pdfrender.pdf</code>	\rightarrow <code>doc/latex/oberdiek/pdfrender.pdf</code>
<code>test/pdfrender-test1.tex</code>	\rightarrow <code>doc/latex/oberdiek/test/pdfrender-test1.tex</code>
<code>test/pdfrender-test2.tex</code>	\rightarrow <code>doc/latex/oberdiek/test/pdfrender-test2.tex</code>
<code>test/pdfrender-test3.tex</code>	\rightarrow <code>doc/latex/oberdiek/test/pdfrender-test3.tex</code>
<code>test/pdfrender-test4.tex</code>	\rightarrow <code>doc/latex/oberdiek/test/pdfrender-test4.tex</code>
<code>test/pdfrender-test5.tex</code>	\rightarrow <code>doc/latex/oberdiek/test/pdfrender-test5.tex</code>
<code>pdfrender.dtx</code>	\rightarrow <code>source/latex/oberdiek/pdfrender.dtx</code>

If you have a `docstrip.cfg` that configures and enables `docstrip`'s TDS installing feature, then some files can already be in the right place, see the documentation of `docstrip`.

4.4 Refresh file name databases

If your $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ distribution (`te $\mathrm{T}_{\mathrm{E}}\mathrm{X}$` , `mik $\mathrm{T}_{\mathrm{E}}\mathrm{X}$` , ...) relies on file name databases, you must refresh these. For example, `te $\mathrm{T}_{\mathrm{E}}\mathrm{X}$` users run `texhash` or `mktextlsr`.

4.5 Some details for the interested

Attached source. The PDF documentation on CTAN also includes the `.dtx` source file. It can be extracted by AcrobatReader 6 or higher. Another option is `pdftk`, e.g. unpack the file into the current directory:

```
pdftk pdfrender.pdf unpack_files output .
```

Unpacking with $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$. The `.dtx` chooses its action depending on the format:

plain- $\mathrm{T}_{\mathrm{E}}\mathrm{X}$: Run `docstrip` and extract the files.

$\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$: Generate the documentation.

If you insist on using $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ for `docstrip` (really, `docstrip` does not need $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{pdfrender.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

Generating the documentation. You can use both the `.dtx` or the `.drv` to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdfL^AT_EX:

```
pdflatex pdfrender.dtx
makeindex -s gind.ist pdfrender.idx
pdflatex pdfrender.dtx
makeindex -s gind.ist pdfrender.idx
pdflatex pdfrender.dtx
```

5 Acknowledgement

Friedrich Vosberg asked in the newsgroup `de.comp.text.tex` for the font outline feature [2].

Gaius Pupus proposed the basic method using `\pdfliteral` in this thread [3].

Rolf Niepraschk added color support [4].

6 References

- [1] Adobe Systems Incorporated. *PDF Reference – Adobe Portable Document format – Version 1.7*. 6th ed. 2006. URL: http://www.adobe.com/devnet/acrobat/pdfs/pdf_reference_1-7.pdf.
- [2] Friedrich Vosberg, *Text in Buchstabenumrissen*, `de.comp.text.tex`, 2010-01-22. URL: <http://groups.google.com/group/de.comp.text.tex/msg/f442310ac8b2d506>.
- [3] Gaius Pupus, *Re: Text in Buchstabenumrissen*, `de.comp.text.tex`, 2010-01-23. URL: <http://groups.google.com/group/de.comp.text.tex/msg/95d890d77ac47eb1>.
- [4] Rolf Niepraschk, *Re: Text in Buchstabenumrissen*, `de.comp.text.tex`, 2010-01-24. URL: <http://groups.google.com/group/de.comp.text.tex/msg/4eb61a5879db54db>.

7 History

[2010/01/26 v1.0]

- The first version.

[2010/01/27 v1.1]

- Macros `\pdfrender` and `\textpdfrender` are made robust.
- Color extraction rewritten for the case that `\pdfmatch` is not available. This fixes wrong color assignments in case of nesting.
- Color extraction of case `\pdfmatch` is fixed for the case that the color string contains several fill or several stroke operations.

[2010/01/28 v1.2]

- Dependency from package `color` is removed.
- Compatibility for plain- \TeX and even $\text{ini}\text{\TeX}$ added.

8 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.

Symbols	
\#	742, 804
\%	807
\.	465, 504, 505, 508, 512, 514, 633, 634, 635, 636, 642, 643, 644, 645
\@	743, 800
\@PackageError	568
\@PackageInfo	317
\@PackageInfoNoLine	161
\@PackageWarning	154, 170, 535, 556, 610, 621
\@PackageWarningNoLine	135
\@ehc	415, 571
\@empty	431
\@firstofone	714, 721, 751, 754
\@gobble	711, 748, 756
\@ifnextchar	579
\@ifundefined	567
\@nil	708, 717, 721
\@nocument	219, 237
\@undefined	52
\[511, 805
\]	481, 710, 729, 801, 899, 905, 907, 913
\{	740, 802, 945
\}	741, 803, 946
\]	513, 806
_	808
A	
\advance	781, 789
\afterassignment	474
\AfterEndPreamble	255, 260
\aftergroup	26, 330, 344
\AfterPackage	252, 253, 254
\AtBeginDocument	248
B	
\begin	245, 823, 859, 887, 888, 918
\bfseries	890
\body	760, 764
C	
\catcode	3, 4, 5, 6, 7, 8, 9, 17, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 64, 65, 68, 69, 70, 71, 75, 76, 77, 78, 82, 84, 740, 741, 742, 743, 778, 787, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 945, 946
\chardef	311, 585, 592, 599
\CheckColor	883, 889, 898, 903, 906, 911, 914, 924
\color	579, 582
\color@begingroup	184, 186, 187
\color@endbox	188, 199
\color@endgroup	185, 188, 228
\color@hbox	186, 199
\color@setgroup	183, 228, 231
\color@vbox	187
\count@	745, 774, 778, 780, 781, 785, 787, 788, 789
\countdef	745
\csname	10, 18, 44, 60, 67, 115, 117, 120, 122, 125, 130, 139, 306, 307, 308, 310, 312, 315, 319, 321, 323, 324, 326, 328, 331, 333, 334, 336, 340, 342, 345, 347, 349, 353, 355, 356, 361, 368, 371, 376, 383, 384, 387, 395, 401, 402, 406, 407, 414, 416, 420, 421, 429, 434, 440, 444, 445, 447, 744, 747, 750, 753, 792, 814, 962
\current@color	601, 614, 625, 669, 672, 689, 692, 720, 721, 726, 884, 922
D	
\DeclareRobustCommand	272, 289
\define@key	398
\dimen	475, 484, 487
\dimexpr	470
\documentclass	820, 855, 876
E	
\empty	13, 14
\end	815, 850, 871, 917, 934, 935, 962
\endcsname	10, 18, 44, 60, 67, 115, 117, 120, 122, 125, 130, 139, 306, 307, 308, 310, 312, 315, 319, 321, 323, 324, 326, 328, 331, 333, 334, 336, 340, 342, 345, 347, 349, 353, 355, 356, 361, 368, 371, 376, 383, 384, 387, 395, 401, 402, 406, 407, 414, 416, 420, 421, 429, 434, 440, 444, 445, 447, 744, 747, 750, 753, 792, 814, 962
\endgraf	185, 193
\endinput	26, 112
\Expect	884
F	
\f	951, 952
\font	951
G	
\gdef	203, 321
H	
\hbox	186
\hsize	939, 941
\Huge	890

I		
\iffalse	123	
\ifnum	420, 607, 618, 650, 731, 780, 788	
\ifpdf	152	
\ifPdfRender@Match		
.....	128, 418, 667, 687, 704	
\ifPdfRender@Stack	127, 309	
\ifPdfRender@Values	299, 366	
\iftrue	118	
\ifx	11, 14, 18, 44, 52,	
	55, 130, 210, 219, 229, 237, 369,	
	431, 481, 527, 534, 548, 555,	
	586, 587, 595, 609, 620, 657,	
	710, 729, 744, 747, 750, 753, 792	
\immediate	20, 46	
\IncludeTests	880	
\input	144, 793, 944	
\iterate	761, 763, 765	
K		
\kv@key	368, 371	
\kv@parse@normalized	367	
\kv@value	369, 371	
\kvsetkeys	279	
L		
\LineWidth	838, 839, 845	
\LoadCommand	793, 810	
\LogTests	881	
\loop	759, 775, 786	
\ltx@empty	277,	
	278, 416, 527, 528, 532, 534,	
	548, 549, 553, 555, 581, 586,	
	587, 595, 609, 620, 657, 665, 685	
\ltx@firstofone	574	
\ltx@firstoftwo	264, 267, 284	
\ltx@GlobalAppendToMacro		
.....	212, 231, 359, 363	
\ltx@gobble	572	
\ltx@gobbletwo	373	
\ltx@ifpackagelater	134	
\ltx@ifpackageloaded	182	
\ltx@ifUndefined		
.....	153, 160, 173, 247, 250, 258,	
	264, 266, 283, 385, 393, 439, 467	
\ltx@ifundefined	189, 192, 202	
\ltx@pkgextension	139	
\ltx@secondoftwo	267, 284	
\ltx@space	162, 450, 451,	
	590, 597, 602, 669, 672, 689, 692	
M		
\makeatletter	882, 919	
\makeatother	886	
\makefootline	947, 948	
\MessageBreak	136, 137,	
	138, 139, 163, 412, 536, 557, 569	
N		
\NeedsTeXFormat	818, 853, 874	
\newcommand	177, 178, 271, 288, 883	
\newlength	838	
\newpage	825, 835, 848	
\next	765, 767, 769	
\noindent	891	
\normalcolor	203, 209, 212, 949	
\number	318	
O		
\OrgMakeFootLine	947, 949	
P		
\PackageError	411	
\PackageInfo	23	
\par	193, 837, 848, 957, 959	
\parfillskip	943	
\pdfcolorstack	325, 335	
\pdfcolorstackinit	155, 313	
\pdflastmatch	652, 655	
\pdfliteral	341, 348	
\pdfmatch	162, 420, 650	
\pdfoutput	938	
\pdfpageheight	942	
\pdfpagewidth	941	
\pdfrender	2, 174, 177, 266, 294	
\PdfRender@@FilterOp	725, 728	
\PdfRender@@PostProcessLineWidth		
.....	474, 480	
\PdfRender@@TryColor	579, 581	
\PdfRender@AtEnd		
...	80, 81, 109, 111, 180, 705, 737	
\PdfRender@AtEndHook	108, 110, 132, 133	
\PdfRender@Color		
.....	526, 527, 532, 547, 548, 553, 582	
\PdfRender@ColorAvailable		
.....	530, 551, 566	
\PdfRender@ColorSetGroupHook		
.....	197, 225, 232, 363	
\PdfRender@Current		
.....	377, 428, 431, 439, 441, 445	
\PdfRender@CurrentColor		
.....	589, 596, 614, 625	
\PdfRender@CurrentLineWidth	475, 486	
\PdfRender@define@key		
.....	375, 382, 393, 524, 545	
\PdfRender@dimexpr	468, 470, 475	
\PdfRender@ErrorInvalidValue		
.....	410, 423, 432, 442	
\PdfRender@FillColor		
.....	277, 528, 534, 586,	
	590, 597, 602, 609, 665, 670, 680	
\PdfRender@FilterOp	680, 700, 724	
\PdfRender@FindOp		
.....	674, 675, 676, 677, 678, 679,	
	694, 695, 696, 697, 698, 699, 707	
\PdfRender@GetFillColor	533, 608, 664	
\PdfRender@GetStrokeColor		
.....	554, 619, 684	
\PdfRender@let	476, 478, 480	
\PdfRender@MatchPattern	649, 670, 690	
\PdfRender@Matchtrue	167	
\PdfRender@NeedsCurrentColor		
.....	585, 592, 599, 607, 618	
\PdfRender@NewClass		
.....	302, 304, 464, 503, 507, 510	

\PdfRender@NewClassValues	\ProvidesPackage
..... 300, 452, 493, 498, 516	15, 61
\PdfRender@newif 114, 127, 128, 299, 305	
\PdfRender@NormalColorHook	R
..... 196, 203, 206, 213, 359	\RangeCatcodeInvalid
\PdfRender@OP . 673, 693, 713, 726, 728 784, 796, 797, 798, 799
\PdfRender@OpName	\repeat
451, 518	759, 771, 782, 790
\PdfRender@OpValue	\RequirePackage
450, 454, 464, 493, 498, 503, 507, 510	147
\PdfRender@PatchColor	\RestoreCatcodes .. 773, 776, 777, 811
241	
\PdfRender@PatchColorSetGroup ..	S
..... 223, 243	\section
\PdfRender@PatchNormalColor	861
201, 242, 276	\set@color 183, 208, 227, 604, 615, 626
\PdfRender@Pattern 650, 668, 688	\setbox
\PdfRender@PatternFillColor 631, 668	199
\PdfRender@PatternStrokeColor ..	\setlength
..... 640, 688	839
\PdfRender@PostProcessLineWidth . 472	\strip@prefix
\PdfRender@relax	652, 655
476, 479	\strip@pt
\PdfRender@RequirePackage	487
..... 129, 149, 150, 151, 262	T
\PdfRender@Reset	\tableofcontents
405	860
\PdfRender@Set	\Test
357, 364, 400	795, 813, 920, 928, 931
\PdfRender@SetColor	\texorpdfstring
280, 584	264
\PdfRender@SetValidate 389, 418	\textpdfrender 2, 175, 178, 283, 827,
\PdfRender@SetValidateValues 378, 438	841, 862, 892, 900, 908, 923, 954
\PdfRender@Stacktrue	\the
158	68, 69, 70, 71, 82, 778
\PdfRender@String	\TMP@EnsureCode
..... 650, 654, 657, 669, 689	. 79, 86, 87, 88, 89, 90, 91, 92,
\PdfRender@StrokeColor	93, 94, 95, 96, 97, 98, 99, 100,
..... 278, 549, 555, 587, 590,	101, 102, 103, 104, 105, 106, 107
595, 597, 602, 620, 685, 690, 700	U
\PdfRender@temp	\usepackage
. 206, 210, 225, 229, 708, 717, 721	821,
\PdfRender@TestBox	822, 856, 857, 858, 877, 878, 879
..... 198, 207, 226, 531, 552	V
\PdfRender@texorpdfstring	\vbox
..... 263, 275, 292	187
\PdfRender@TryColor ... 532, 553, 578	\vfill
\PdfRender@Valuesfalse	957, 960
380	\vsize
\PdfRender@Valustrue	940, 942
301	W
\penalty	\write
957, 961	20, 46
\ProvidesFile	X
819, 854, 875	\x
	10, 11, 14, 19,
	23, 25, 45, 50, 60, 66, 74, 485, 490