

The pdfrender package

Heiko Oberdiek*

<heiko.oberdiek at gmail.com>

2016/05/14 v1.3

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	4
2	Implementation	4
2.1	Look for pdfTeX, its mode and features	6
2.2	Enable color support of L ^A T _E X	8
2.3	Hook into \normalcolor	8
2.4	Declare and setup parameters	13
2.5	Fill and stroke color support	15
3	Test	19
3.1	Catcode checks for loading	19
3.2	Simple test file	21
3.3	Further tests	21
3.4	Compatibility with plain T _E X	23
4	Installation	23
4.1	Download	23
4.2	Bundle installation	24
4.3	Package installation	24
4.4	Refresh file name databases	24
4.5	Some details for the interested	25
5	Catalogue	25
6	Acknowledgement	26

*Please report any issues at <https://github.com/ho-tex/oberdiek/issues>

7	References	26
8	History	26
	[2010/01/26 v1.0]	26
	[2010/01/27 v1.1]	26
	[2010/01/28 v1.2]	27
	[2016/05/14 v1.3]	27
9	Index	27

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</i> <i>dimen</i>
LineCapStyle	0	Butt
	1	Round
	2	ProjectingSquare
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 T_EX *dimen* specification can be given. This includes explicit specifications with number and unit, e.g. `LineWidth=0.5pt`. Also L^AT_EX length registers may be used. If ε -T_EX'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). L^AT_EX'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 L^AT_EX 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 L^AT_EX 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 pdfT_EX 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\catcode61\catcode48\catcode32=10\relax%
3 \catcode13=5 % ^~M
4 \endlinechar=13 %
5 \catcode35=6 % #
6 \catcode39=12 % '
7 \catcode44=12 % ,
8 \catcode45=12 % -
9 \catcode46=12 % .
10 \catcode58=12 % :
11 \catcode64=11 % @
12 \catcode123=1 % {
13 \catcode125=2 % }
14 \expandafter\let\expandafter\x\csname ver@pdfrender.sty\endcsname
15 \ifx\x\relax % plain-TEX, first loading
16 \else
17 \def\empty{}%
18 \ifx\x\empty % LATEX, first loading,
19 % variable is initialized, but \ProvidesPackage not yet seen
20 \else
21 \expandafter\ifx\csname PackageInfo\endcsname\relax
22 \def\x#1#2{%
23 \immediate\write-1{Package #1 Info: #2.}%
24 }%
25 \else
26 \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
27 \fi
28 \x{pdfrender}{The package is already loaded}%
29 \aftergroup\endinput
30 \fi
31 \fi
```

```

32 \endgroup%
Package identification:
33 \begingroup\catcode61\catcode48\catcode32=10\relax%
34 \catcode13=5 % ^^M
35 \endlinechar=13 %
36 \catcode35=6 % #
37 \catcode39=12 % '
38 \catcode40=12 % (
39 \catcode41=12 % )
40 \catcode44=12 % ,
41 \catcode45=12 % -
42 \catcode46=12 % .
43 \catcode47=12 % /
44 \catcode58=12 % :
45 \catcode64=11 % @
46 \catcode91=12 % [
47 \catcode93=12 % ]
48 \catcode123=1 % {
49 \catcode125=2 % }
50 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
51 \def\x#1#2#3[#4]{\endgroup
52 \immediate\write-1{Package: #3 #4}%
53 \xdef#1{#4}%
54 }%
55 \else
56 \def\x#1#2[#3]{\endgroup
57 #2[#{#3}]%
58 \ifx#1\@undefined
59 \xdef#1{#3}%
60 \fi
61 \ifx#1\relax
62 \xdef#1{#3}%
63 \fi
64 }%
65 \fi
66 \expandafter\x\csname ver@pdfrender.sty\endcsname
67 \ProvidesPackage{pdfrender}%
68 [2016/05/14 v1.3 Access to some PDF graphics parameters (H0)]%
69 \begingroup\catcode61\catcode48\catcode32=10\relax%
70 \catcode13=5 % ^^M
71 \endlinechar=13 %
72 \catcode123=1 % {
73 \catcode125=2 % }
74 \catcode64=11 % @
75 \def\x{\endgroup
76 \expandafter\edef\csname PdfRender@AtEnd\endcsname{%
77 \endlinechar=\the\endlinechar\relax
78 \catcode13=\the\catcode13\relax
79 \catcode32=\the\catcode32\relax
80 \catcode35=\the\catcode35\relax
81 \catcode61=\the\catcode61\relax
82 \catcode64=\the\catcode64\relax
83 \catcode123=\the\catcode123\relax
84 \catcode125=\the\catcode125\relax
85 }%
86 }%
87 \x\catcode61\catcode48\catcode32=10\relax%
88 \catcode13=5 % ^^M

```

```

89 \endlinechar=13 %
90 \catcode35=6 % #
91 \catcode64=11 % @
92 \catcode123=1 % {
93 \catcode125=2 % }
94 \def\TMP@EnsureCode#1#2{%
95   \edef\PdfRender@AtEnd{%
96     \PdfRender@AtEnd
97     \catcode#1=\the\catcode#1\relax
98   }%
99   \catcode#1=#2\relax
100 }
101 \TMP@EnsureCode{10}{12}% ^^J
102 \TMP@EnsureCode{36}{3}% $
103 \TMP@EnsureCode{39}{12}% '
104 \TMP@EnsureCode{40}{12}% (
105 \TMP@EnsureCode{41}{12}% )
106 \TMP@EnsureCode{42}{12}% *
107 \TMP@EnsureCode{43}{12}% +
108 \TMP@EnsureCode{44}{12}% ,
109 \TMP@EnsureCode{45}{12}% -
110 \TMP@EnsureCode{46}{12}% .
111 \TMP@EnsureCode{47}{12}% /
112 \TMP@EnsureCode{58}{12}% :
113 \TMP@EnsureCode{59}{12}% ;
114 \TMP@EnsureCode{60}{12}% <
115 \TMP@EnsureCode{62}{12}% >
116 \TMP@EnsureCode{63}{12}% ?
117 \TMP@EnsureCode{91}{12}% [
118 \TMP@EnsureCode{93}{12}% ]
119 \TMP@EnsureCode{94}{7}% ^ (superscript)
120 \TMP@EnsureCode{96}{12}% '
121 \TMP@EnsureCode{124}{12}% |

Luatex compatibility
122 \ifx\pdfextension\undefined\else
123   \RequirePackage{luatex85}
124 \fi

125 \def\PdfRender@AtEndHook{}
126 \expandafter\def\expandafter\PdfRender@AtEnd\expandafter{%
127   \expandafter\PdfRender@AtEndHook
128   \PdfRender@AtEnd
129   \endinput
130 }

```

2.1 Look for pdfTeX, its mode and features

\PdfRender@newif

```

131 \def\PdfRender@newif#1{%
132   \expandafter\edef\csname PdfRender@#1true\endcsname{%
133     \let
134     \expandafter\noexpand\csname ifPdfRender@#1\endcsname
135     \noexpand\iftrue
136   }%
137   \expandafter\edef\csname PdfRender@#1false\endcsname{%
138     \let
139     \expandafter\noexpand\csname ifPdfRender@#1\endcsname
140     \noexpand\iffalse
141   }%

```

```

142 \csname PdfRender@#1false\endcsname
143 }

\ifPdfRender@Stack
144 \PdfRender@newif{Stack}

\ifPdfRender@Match
145 \PdfRender@newif{Match}

\PdfRender@RequirePackage
146 \begingroup\expandafter\expandafter\expandafter\endgroup
147 \expandafter\ifx\csname RequirePackage\endcsname\relax
148 \def\PdfRender@RequirePackage#1[#2]{%
149 \expandafter\def\expandafter\PdfRender@AtEndHook\expandafter{%
150 \PdfRender@AtEndHook
151 \ltx@ifpackagelater{#1}{#2}{-}{%
152 \@PackageWarningNoLine{pdfrender}{%
153 You have requested version\MessageBreak
154 '#2' of package '#1',\MessageBreak
155 but only version\MessageBreak
156 '\csname ver@#1.\ltx@pkgextension\endcsname'\MessageBreak
157 is available%
158 }%
159 }%
160 }%
161 \input #1.sty\relax
162 }%
163 \else
164 \let\PdfRender@RequirePackage\RequirePackage
165 \fi

166 \PdfRender@RequirePackage{ifpdf}[2010/01/28]
167 \PdfRender@RequirePackage{infwarerr}[2007/09/09]
168 \PdfRender@RequirePackage{ltxcmds}[2010/01/28]

169 \ifpdf
170 \ltx@ifundefined{pdfcolorstackinit}{%
171 \@PackageWarning{pdfrender}{%
172 Missing \string\pdfcolorstackinit
173 }%
174 }{%
175 \PdfRender@Stacktrue
176 }%
177 \ltx@ifundefined{pdfmatch}{%
178 \@PackageInfoNoLine{pdfrender}{%
179 \string\pdfmatch\ltx@space not found. %
180 Therefore the values\MessageBreak
181 of some parameters are not validated%
182 }%
183 }{%
184 \PdfRender@Matchtrue
185 }%
186 \else
187 \@PackageWarning{pdfrender}{%
188 Missing pdfTeX in PDF mode%
189 }%
190 \ltx@ifundefined{newcommand}{%

```

```

\pdfrender
191 \def\pdfrender#1{%

\textpdfrender
192 \long\def\textpdfrender#1#2{#2}%
193 }%

\pdfrender
194 \newcommand*\pdfrender[1]{}%

\textpdfrender
195 \newcommand{\textpdfrender}[2]{#2}%
196 }%
197 \expandafter\PdfRender@AtEnd
198 \fi%

```

2.2 Enable color support of L^AT_EX

```

199 \ltx@ifpackageloaded{color}{}%
200 \def\color@setgroup{\begingroup\set@color}%
201 \let\color@begingroup\begingroup
202 \def\color@endgroup{\endgraf\endgroup}%
203 \def\color@hbox{\hbox\bgroup\color@begingroup}%
204 \def\color@vbox{\vbox\bgroup\color@begingroup}%
205 \def\color@endbox{\color@endgroup\egroup}%
206 \ltx@ifundefined{bgroup}{%
207 \let\bgroup={\let\egroup=}
208 }{}%
209 \ltx@ifundefined{endgraf}{%
210 \let\endgraf=\par
211 }{}%
212 }

```

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
213 \def\PdfRender@NormalColorHook{}

\PdfRender@ColorSetGroupHook
214 \def\PdfRender@ColorSetGroupHook{}

\PdfRender@TestBox
215 \def\PdfRender@TestBox#1{%
216 \setbox0=\color@hbox#1\color@endbox
217 }

\PdfRender@PatchNormalColor
218 \def\PdfRender@PatchNormalColor{%
219 \ltx@ifundefined{normalcolor}{%
220 \gdef\normalcolor{\PdfRender@NormalColorHook}%

```

```

221 }{%
222   \begingroup
223   \def\PdfRender@NormalColorHook{\let\PdfRender@temp=Y}%
224   \PdfRender@TestBox{%
225     \let\set@color\relax
226     \normalcolor
227     \ifx\PdfRender@temp Y%
228     \else
229       \ltx@GlobalAppendToMacro\normalcolor{%
230         \PdfRender@NormalColorHook
231       }%
232     \fi
233   }%
234   \endgroup
235 }%
236 \ifx\@nocument\relax
237   \global\let\PdfRender@PatchNormalColor\relax
238 \fi
239 }%

```

\PdfRender@PatchColorSetGroup

```

240 \def\PdfRender@PatchColorSetGroup{%
241   \begingroup
242   \def\PdfRender@ColorSetGroupHook{\let\PdfRender@temp=Y}%
243   \PdfRender@TestBox{%
244     \let\set@color\relax
245     \color@setgroup\color@endgroup
246     \ifx\PdfRender@temp Y%
247     \else
248       \ltx@GlobalAppendToMacro\color@setgroup{%
249         \PdfRender@ColorSetGroupHook
250       }%
251     \fi
252   }%
253   \endgroup
254   \ifx\@nocument\relax
255     \global\let\PdfRender@PatchColorSetGroup\relax
256   \fi
257 }%

```

\PdfRender@PatchColor

```

258 \def\PdfRender@PatchColor{%
259   \PdfRender@PatchNormalColor
260   \PdfRender@PatchColorSetGroup
261 }

262 \PdfRender@PatchColor
263 \ltx@ifundefined{AtBeginDocument}{-}{%
264   \AtBeginDocument{\PdfRender@PatchColor}%
265 }

```

\AfterPackage is provided by package scrfile.

```

266 \ltx@ifundefined{AfterPackage}{-}{%
267 }{%
268   \AfterPackage{color}{\PdfRender@PatchColor}%
269   \AfterPackage{xcolor}{\PdfRender@PatchColor}%
270   \AfterPackage{etoolbox}{%
271     \AfterEndPreamble{\PdfRender@PatchColor}%
272   }%

```

```

273 }%
\AfterEndPreamble is provided by package etoolbox.
274 \ltx@ifundefined{AfterEndPreamble}{%
275 }{%
276   \AfterEndPreamble{\PdfRender@PatchColor}%
277 }%
278 \PdfRender@RequirePackage{kvsetkeys}[2010/01/28]

\PdfRender@texorpdfstring
279 \def\PdfRender@texorpdfstring{%
280   \ltx@ifundefined{texorpdfstring}\ltx@firstoftwo\texorpdfstring
281 }

\pdfrender
282 \ltx@ifundefined{DeclareRobustCommand}%
283 \ltx@firstoftwo\ltx@secondoftwo
284 {%
285   \def\pdfrender#1%
286 }{%
287   \newcommand{\pdfrender}{}%
288   \DeclareRobustCommand*\pdfrender[1]%
289 }%
290 {%
291   \PdfRender@texorpdfstring{%
292     \PdfRender@PatchNormalColor
293     \global\let\PdfRender@FillColor\ltx@empty
294     \global\let\PdfRender@StrokeColor\ltx@empty
295     \kvsetkeys{PDFRENDER}{#1}%
296     \PdfRender@SetColor
297   }{}%
298 }

\textpdfrender
299 \ltx@ifundefined{DeclareRobustCommand}%
300 \ltx@firstoftwo\ltx@secondoftwo
301 {%
302   \long\def\textpdfrender#1#2%
303 }{%
304   \newcommand{\textpdfrender}{}%
305   \DeclareRobustCommand{\textpdfrender}[2]%
306 }%
307 {%
308   \PdfRender@texorpdfstring{%
309     \begingroup
310       \pdfrender{#1}%
311       #2%
312     \endgroup
313   }{#2}%
314 }

\ifPdfRender@Values
315 \PdfRender@newif{Values}

\PdfRender@NewClassValues
316 \def\PdfRender@NewClassValues#1#2#3#4{%
317   \PdfRender@Valuestrue
318   \PdfRender@NewClass{#1}{#2}{#3}{#4}{}%
319 }

```

\PdfRender@NewClass

```
320 \def\PdfRender@NewClass#1#2#3#4#5{%
321   \PdfRender@newif{Active#1}%
322   \expandafter\def\csname PdfRender@Default#1\endcsname{#2}%
323   \expandafter\let\csname PdfRender@Current#1\expandafter\endcsname
324     \csname PdfRender@Default#1\endcsname
325   \ifPdfRender@Stack
326     \expandafter\edef\csname PdfRender@Init#1\endcsname{%
327       \global\chardef
328       \expandafter\noexpand\csname PdfRender@Stack#1\endcsname=
329         \noexpand\pdfcolorstackinit page direct{%
330           \noexpand#3%
331           \expandafter\noexpand\csname PdfRender@Default#1\endcsname
332         }\relax
333       \noexpand\@PackageInfo{pdfrender}{%
334         New color stack ‘#1’ = \noexpand\number
335         \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
336       }%
337       \gdef\expandafter\noexpand\csname PdfRender@Init#1\endcsname{%
338     }%
339     \expandafter\edef\csname PdfRender@Set#1\endcsname{%
340       \expandafter\noexpand\csname PdfRender@Init#1\endcsname
341       \noexpand\pdfcolorstack
342       \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
343       push{%
344         #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
345       }%
346       \noexpand\aftergroup
347       \expandafter\noexpand\csname PdfRender@Reset#1\endcsname
348     }%
349     \expandafter\edef\csname PdfRender@Reset#1\endcsname{%
350       \expandafter\noexpand\csname PdfRender@Init#1\endcsname
351       \noexpand\pdfcolorstack
352       \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
353       pop\relax
354     }%
355   \else
356     \expandafter\edef\csname PdfRender@Set#1\endcsname{%
357       \noexpand\pdfliteral direct{%
358         #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
359       }%
360       \noexpand\aftergroup
361       \expandafter\noexpand\csname PdfRender@Reset#1\endcsname
362     }%
363     \expandafter\edef\csname PdfRender@Reset#1\endcsname{%
364       \noexpand\pdfliteral direct{%
365         #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
366       }%
367     }%
368   \fi
369   \expandafter\edef\csname PdfRender@Normal#1\endcsname{%
370     \let
371     \expandafter\noexpand\csname PdfRender@Current#1\endcsname
372     \expandafter\noexpand\csname PdfRender@Default#1\endcsname
373     \noexpand\PdfRender@Set{#1}%
374   }%
375   \expandafter\ltx@GlobalAppendToMacro\expandafter\PdfRender@NormalColorHook
376   \expandafter{%
```

```

377 \csname PdfRender@Normal#1\endcsname
378 }%
379 \ltx@GlobalAppendToMacro\PdfRender@ColorSetGroupHook{%
380 \PdfRender@Set{#1}%
381 }%
382 \ifPdfRender@Values
383 \kv@parse@normalized{#4}{%
384 \expandafter\let\csname PdfRender@#1@\kv@key\endcsname\kv@key
385 \ifx\kv@value\relax
386 \else
387 \expandafter\let\csname PdfRender@#1@\kv@value\endcsname\kv@key
388 \fi
389 \ltx@gobbletwo
390 }%
391 \PdfRender@define@key{PDFRENDER}{#1}{%
392 \global\csname PdfRender@Active#1true\endcsname
393 \def\PdfRender@Current{##1}%
394 \PdfRender@SetValidateValues{#1}%
395 }%
396 \PdfRender@Valuesfalse
397 \else
398 \PdfRender@define@key{PDFRENDER}{#1}{%
399 \global\csname PdfRender@Active#1true\endcsname
400 \expandafter\def\csname PdfRender@Current#1\endcsname{##1}%
401 \ltx@ifundefined{PdfRender@PostProcess#1}{%
402 }{%
403 \csname PdfRender@PostProcess#1\endcsname
404 }%
405 \PdfRender@SetValidate{#1}{#4}{#5}%
406 }%
407 \fi
408 }%

```

\PdfRender@define@key

```

409 \ltx@ifundefined{define@key}{%
410 \def\PdfRender@define@key#1#2{%
411 \expandafter\def\csname KV@#1@#2\endcsname##1%
412 }%
413 }{%
414 \let\PdfRender@define@key\define@key
415 }

```

\PdfRender@Set

```

416 \def\PdfRender@Set#1{%
417 \csname ifPdfRender@Active#1\endcsname
418 \csname PdfRender@Set#1\expandafter\endcsname
419 \fi
420 }

```

\PdfRender@Reset

```

421 \def\PdfRender@Reset#1{%
422 \csname ifPdfRender@Active#1\endcsname
423 \csname PdfRender@Reset#1\expandafter\endcsname
424 \fi
425 }

```

\PdfRender@ErrorInvalidValue

```

426 \def\PdfRender@ErrorInvalidValue#1{%

```

```

427 \PackageError{pdfrender}{%
428   Ignoring parameter setting for '#1'\MessageBreak
429   because of invalid value %
430   '\csname PdfRender@Current#1\endcsname'%
431 }{@ehc
432 \expandafter\let\csname PdfRender@Current#1\endcsname\ltx@empty
433 }%

```

\PdfRender@SetValidate

```

434 \ifPdfRender@Match
435   \def\PdfRender@SetValidate#1#2#3{%
436     \ifnum\pdfmatch{^(#2)$}\csname PdfRender@Current#1\endcsname}=1 %
437     \csname PdfRender@Set#1\expandafter\endcsname
438   \else
439     \PdfRender@ErrorInvalidValue{#1}%
440   \fi
441 }%
442 \else
443   \def\PdfRender@SetValidate#1#2#3{%
444     \expandafter\let\expandafter\PdfRender@Current
445     \csname PdfRender@Current#1\endcsname
446     #3%
447     \ifx\PdfRender@Current\@empty
448       \PdfRender@ErrorInvalidValue{#1}%
449     \else
450       \csname PdfRender@Set#1\expandafter\endcsname
451     \fi
452   }%
453 \fi

```

\PdfRender@SetValidateValues

```

454 \def\PdfRender@SetValidateValues#1{%
455   \ltx@ifundefined{PdfRender@#1@PdfRender@Current}{%
456     \expandafter\let\csname PdfRender@Current#1\endcsname
457       \PdfRender@Current
458     \PdfRender@ErrorInvalidValue{#1}%
459   }{%
460     \expandafter\edef\csname PdfRender@Current#1\endcsname{%
461       \csname PdfRender@#1@PdfRender@Current\endcsname
462     }%
463     \csname PdfRender@Set#1\endcsname
464   }%
465 }

```

\PdfRender@OpValue

```

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

```

\PdfRender@OpName

```

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

```

2.4 Declare and setup parameters

```

468 \PdfRender@NewClassValues{TextRenderingMode}%
469   {0}%
470   {\PdfRender@OpValue{Tr}}{%
471   0=Fill,%
472   1=Stroke,%
473   2=FillStroke,%

```

```

474 3=Invisible,%
475 4=FillClip,%
476 5=StrokeClip,%
477 6=FillStrokeClip,%
478 7=Clip,%
479 }%
480 \PdfRender@NewClass{LineWidth}{1}{\PdfRender@OpValue{w}}{%
481   [0-9]+\string\.[0-9]*|\string\.[0-9]+%
482 }{%
483 \let\IfUndefined{dimexpr}{%
484   \def\PdfRender@dimexpr{%
485 }{%
486   \let\PdfRender@dimexpr\dimexpr
487 }
488 \def\PdfRender@PostProcessLineWidth{%
489   \begin{group}
490     \afterassignment\PdfRender@@PostProcessLineWidth
491     \dimen0=\PdfRender@dimexpr\PdfRender@CurrentLineWidth bp %
492     \PdfRender@let\PdfRender@relax\PdfRender@relax
493   }
494   \let\PdfRender@let\let
495   \let\PdfRender@relax\relax
496   \def\PdfRender@@PostProcessLineWidth#1\PdfRender@let{%
497     \ifx\#1\%
498       \endgroup
499     \else
500       \dimen0=.996264\dimen0 % 72/72.27
501       \edef\x{\endgroup
502         \def\noexpand\PdfRender@CurrentLineWidth{%
503           \strip@pt\dimen0%
504         }%
505       }%
506       \expandafter\x
507     \fi
508   }
509   \PdfRender@NewClassValues{LineCapStyle}{0}{\PdfRender@OpValue{J}}{%
510     0=Butt,%
511     1=Round,%
512     2=ProjectingSquare,%
513   }%
514   \PdfRender@NewClassValues{LineJoinStyle}{0}{\PdfRender@OpValue{j}}{%
515     0=Miter,%
516     1=Round,%
517     2=Bevel,%
518   }%
519   \PdfRender@NewClass{MiterLimit}{10}{\PdfRender@OpValue{M}}{%
520     [0-9]*[1-9][0-9]*\string\.[0-9]*|
521     [0-9]*\string\.[0-9]*[1-9][0-9]*%
522   }{%
523   \PdfRender@NewClass{Flatness}{0}{\PdfRender@OpValue{i}}{%
524     100(\string\.[0-9]*|\string\.[0-9]*|\string\.[0-9]+%
525   }{%
526   \PdfRender@NewClass{LineDashPattern}{[]}{\PdfRender@OpValue{d}}{%
527     \string[%
528     ( ?([0-9]+\string\.[0-9]*|\string\.[0-9]+) ?)*%
529     \string\] ?%
530     ([0-9]+\string\.[0-9]*|\string\.[0-9]+)%
531   }{%

```

```

532 \PdfRender@NewClassValues{RenderingIntent}%
533           {RelativeColorimetric}%
534           {\PdfRender@OpName{ri}}}%
535 AbsoluteColorimetric,%
536 RelativeColorimetric,%
537 Saturation,%
538 Perceptual,%
539 }%

```

2.5 Fill and stroke color support

```

540 \PdfRender@define@key{PDFRENDER}{FillColor}{%
541   \begingroup
542     \def\PdfRender@Color{#1}%
543     \ifx\PdfRender@Color\ltx@empty
544       \global\let\PdfRender@FillColor\ltx@empty
545     \else
546       \PdfRender@ColorAvailable{%
547         \PdfRender@TestBox{%
548           \expandafter\PdfRender@TryColor\PdfRender@Color\ltx@empty
549           \PdfRender@GetFillColor
550           \ifx\PdfRender@FillColor\ltx@empty
551             \@PackageWarning{pdfrender}{%
552               Cannot extract fill color\MessageBreak
553               from value ‘#1’%
554             }%
555           \fi
556         }%
557       }%
558     \fi
559   \endgroup
560 }
561 \PdfRender@define@key{PDFRENDER}{StrokeColor}{%
562   \begingroup
563     \def\PdfRender@Color{#1}%
564     \ifx\PdfRender@Color\ltx@empty
565       \global\let\PdfRender@StrokeColor\ltx@empty
566     \else
567       \PdfRender@ColorAvailable{%
568         \PdfRender@TestBox{%
569           \expandafter\PdfRender@TryColor\PdfRender@Color\ltx@empty
570           \PdfRender@GetStrokeColor
571           \ifx\PdfRender@StrokeColor\ltx@empty
572             \@PackageWarning{pdfrender}{%
573               Cannot extract stroke color\MessageBreak
574               from value ‘#1’%
575             }%
576           \fi
577         }%
578       }%
579     \fi
580   \endgroup
581 }

```

\PdfRender@ColorAvailable

```

582 \def\PdfRender@ColorAvailable{%
583   \@ifundefined{set@color}{%
584     \@PackageError{pdfrender}{%
585       Ignoring color options, because neither\MessageBreak

```

```

586     package 'color' nor package 'xcolor' is loaded%
587   }\@ehc
588   \global\let\PdfRender@ColorAvailable\ltx@gobble
589 }{%
590   \global\let\PdfRender@ColorAvailable\ltx@firstofone
591 }%
592 \PdfRender@ColorAvailable
593 }

\PdfRender@TryColor

594 \def\PdfRender@TryColor{%
595   \@ifnextchar[\color\PdfRender@@TryColor
596 }

\PdfRender@@TryColor

597 \def\PdfRender@@TryColor#1\ltx@empty{%
598   \expandafter\color\expandafter{\PdfRender@Color}%
599 }

\PdfRender@SetColor

600 \def\PdfRender@SetColor{%
601   \chardef\PdfRender@NeedsCurrentColor=0 %
602   \ifx\PdfRender@FillColor\ltx@empty
603     \ifx\PdfRender@StrokeColor\ltx@empty
604       \else
605         \edef\PdfRender@CurrentColor{%
606           \noexpand\PdfRender@FillColor\ltx@space\PdfRender@StrokeColor
607         }%
608         \chardef\PdfRender@NeedsCurrentColor=1 %
609       \fi
610     \else
611       \ifx\PdfRender@StrokeColor\ltx@empty
612         \edef\PdfRender@CurrentColor{%
613           \PdfRender@FillColor\ltx@space\noexpand\PdfRender@StrokeColor
614         }%
615         \chardef\PdfRender@NeedsCurrentColor=2 %
616       \else
617         \edef\current@color{%
618           \PdfRender@FillColor\ltx@space\PdfRender@StrokeColor
619         }%
620         \set@color
621       \fi
622     \fi
623   \ifnum\PdfRender@NeedsCurrentColor=1 %
624     \PdfRender@GetFillColor
625     \ifx\PdfRender@FillColor\ltx@empty
626       \@PackageWarning{pdfrender}{%
627         Cannot extract current fill color%
628       }%
629     \else
630       \edef\current@color{\PdfRender@CurrentColor}%
631       \set@color
632     \fi
633   \else
634     \ifnum\PdfRender@NeedsCurrentColor=2 %
635       \PdfRender@GetStrokeColor
636       \ifx\PdfRender@StrokeColor\ltx@empty
637         \@PackageWarning{pdfrender}{%

```

```

638         Cannot extract current stroke color%
639     }%
640     \else
641         \edef\current@color{\PdfRender@CurrentColor}%
642         \set@color
643     \fi
644 \fi
645 \fi
646 }

```

\PdfRender@PatternFillColor

```

647 \edef\PdfRender@PatternFillColor{ % space
648     (%
649     [0-9\string\.] + g|%
650     [0-9\string\.] + [0-9\string\.] + [0-9\string\.] + rg|%
651     [0-9\string\.] + [0-9\string\.] + %
652     [0-9\string\.] + [0-9\string\.] + k%
653     ) % space
654     (.*)$%
655 }

```

\PdfRender@PatternStrokeColor

```

656 \edef\PdfRender@PatternStrokeColor{ % space
657     (%
658     [0-9\string\.] + G|%
659     [0-9\string\.] + [0-9\string\.] + [0-9\string\.] + RG|%
660     [0-9\string\.] + [0-9\string\.] + %
661     [0-9\string\.] + [0-9\string\.] + K%
662     ) % space
663     (.*)$%
664 }

```

\PdfRender@MatchPattern

```

665 \def\PdfRender@MatchPattern#1{%
666     \ifnum\pdfmatch{\PdfRender@Pattern}{\PdfRender@String}=1 %
667         \xdef#1{%
668             \expandafter\strip@prefix\pdflastmatch 1%
669         }%
670         \edef\PdfRender@String{%
671             \expandafter\strip@prefix\pdflastmatch 2%
672         }%
673         \ifx\PdfRender@String\ltx@empty
674             \else
675                 \expandafter\expandafter\expandafter\PdfRender@MatchPattern
676                 \expandafter\expandafter\expandafter#1%
677             \fi
678         \fi
679 }

```

\PdfRender@GetFillColor

```

680 \def\PdfRender@GetFillColor{%
681     \global\let\PdfRender@FillColor\ltx@empty
682     \begingroup
683         \if\PdfRender@Match
684             \let\PdfRender@Pattern\PdfRender@PatternFillColor
685             \edef\PdfRender@String{\ltx@space\current@color\ltx@space}%
686             \PdfRender@MatchPattern\PdfRender@FillColor
687         \else

```

```

688     \edef\current@color{\current@color\ltx@space}%
689     \let\PdfRender@OP\relax
690     \PdfRender@FindOp{g}0%
691     \PdfRender@FindOp{G}1%
692     \PdfRender@FindOp{rg}0%
693     \PdfRender@FindOp{RG}1%
694     \PdfRender@FindOp{k}0%
695     \PdfRender@FindOp{K}1%
696     \PdfRender@FilterOp 0\PdfRender@FillColor
697     \fi
698 \endgroup
699 }

```

\PdfRender@GetStrokeColor

```

700 \def\PdfRender@GetStrokeColor{%
701   \global\let\PdfRender@StrokeColor\ltx@empty
702   \begingroup
703     \ifPdfRender@Match
704       \let\PdfRender@Pattern\PdfRender@PatternStrokeColor
705       \edef\PdfRender@String{\ltx@space\current@color\ltx@space}%
706       \PdfRender@MatchPattern\PdfRender@StrokeColor
707     \else
708       \edef\current@color{\current@color\ltx@space}%
709       \let\PdfRender@OP\relax
710       \PdfRender@FindOp{g}0%
711       \PdfRender@FindOp{G}1%
712       \PdfRender@FindOp{rg}0%
713       \PdfRender@FindOp{RG}1%
714       \PdfRender@FindOp{k}0%
715       \PdfRender@FindOp{K}1%
716       \PdfRender@FilterOp 1\PdfRender@StrokeColor
717     \fi
718   \endgroup
719 }

720 \ifPdfRender@Match
721   \expandafter\PdfRender@AtEnd
722 \fi%

```

\PdfRender@FindOp

```

723 \def\PdfRender@FindOp#1#2{%
724   \def\PdfRender@temp##1 #1 ##2\@nil{%
725     ##1%
726     \ifx\##2\%
727       \expandafter\@gobble
728     \else
729       \PdfRender@OP{#1}#2%
730       \expandafter\@firstofone
731     \fi
732     {%
733       \PdfRender@temp##2\@nil
734     }%
735   }%
736   \edef\current@color{%
737     \@firstofone{\expandafter\PdfRender@temp\current@color} #1 \@nil
738   }%
739 }

```

\PdfRender@FilterOp

```
740 \def\PdfRender@FilterOp#1#2{%
741   \expandafter\PdfRender@@FilterOp\expandafter#1\expandafter#2%
742   \current@color\PdfRender@OP{}}}%
743 }
```

\PdfRender@@FilterOp

```
744 \def\PdfRender@@FilterOp#1#2#3\PdfRender@OP#4#5{%
745   \ifx\#4#5\\%
746   \else
747     \ifnum#1=#5 %
748       \xdef#2{#3 #4}%
749     \fi
750     \expandafter\PdfRender@@FilterOp\expandafter#1\expandafter#2%
751   \fi
752 }

753 \PdfRender@AtEnd%
754 \end{package}
```

3 Test

3.1 Catcode checks for loading

```
755 (*test1)

756 \catcode'\{=1 %
757 \catcode'\}=2 %
758 \catcode'\#=6 %
759 \catcode'\@=11 %
760 \expandafter\ifx\csname count@\endcsname\relax
761   \countdef\count@=255 %
762 \fi
763 \expandafter\ifx\csname @gobble\endcsname\relax
764   \long\def\@gobble#1{}%
765 \fi
766 \expandafter\ifx\csname @firstofone\endcsname\relax
767   \long\def\@firstofone#1{#1}%
768 \fi
769 \expandafter\ifx\csname loop\endcsname\relax
770   \expandafter\@firstofone
771 \else
772   \expandafter\@gobble
773 \fi
774 {%
775   \def\loop#1\repeat{%
776     \def\body{#1}%
777     \iterate
778   }%
779   \def\iterate{%
780     \body
781     \let\next\iterate
782   \else
783     \let\next\relax
784   \fi
785   \next
786 }%
787 \let\repeat=\fi
```

```

788 }%
789 \def\RestoreCatcodes{}
790 \count@=0 %
791 \loop
792   \edef\RestoreCatcodes{%
793     \RestoreCatcodes
794     \catcode\the\count@=\the\catcode\count@\relax
795   }%
796 \ifnum\count@<255 %
797   \advance\count@ 1 %
798 \repeat
799
800 \def\RangeCatcodeInvalid#1#2{%
801   \count@=#1\relax
802   \loop
803     \catcode\count@=15 %
804   \ifnum\count@<#2\relax
805     \advance\count@ 1 %
806   \repeat
807 }
808 \def\RangeCatcodeCheck#1#2#3{%
809   \count@=#1\relax
810   \loop
811     \ifnum#3=\catcode\count@
812   \else
813     \errmessage{%
814       Character \the\count@\space
815       with wrong catcode \the\catcode\count@\space
816       instead of \number#3%
817     }%
818   \fi
819   \ifnum\count@<#2\relax
820     \advance\count@ 1 %
821   \repeat
822 }
823 \def\space{ }
824 \expandafter\ifx\csname LoadCommand\endcsname\relax
825   \def\LoadCommand{\input pdfrender.sty\relax}%
826 \fi
827 \def\Test{%
828   \RangeCatcodeInvalid{0}{47}%
829   \RangeCatcodeInvalid{58}{64}%
830   \RangeCatcodeInvalid{91}{96}%
831   \RangeCatcodeInvalid{123}{255}%
832   \catcode'\@=12 %
833   \catcode'\=0 %
834   \catcode'\%=14 %
835   \LoadCommand
836   \RangeCatcodeCheck{0}{36}{15}%
837   \RangeCatcodeCheck{37}{37}{14}%
838   \RangeCatcodeCheck{38}{47}{15}%
839   \RangeCatcodeCheck{48}{57}{12}%
840   \RangeCatcodeCheck{58}{63}{15}%
841   \RangeCatcodeCheck{64}{64}{12}%
842   \RangeCatcodeCheck{65}{90}{11}%
843   \RangeCatcodeCheck{91}{91}{15}%
844   \RangeCatcodeCheck{92}{92}{0}%
845   \RangeCatcodeCheck{93}{96}{15}%

```

```

846 \RangeCatcodeCheck{97}{122}{11}%
847 \RangeCatcodeCheck{123}{255}{15}%
848 \RestoreCatcodes
849 }
850 \Test
851 \csname @@end\endcsname
852 \end
853 \</test1>

```

3.2 Simple test file

```

854 \<test2>
855 \NeedsTeXFormat{LaTeX2e}
856 \ProvidesFile{pdfrender-test2.tex}[2016/05/14]
857 \documentclass{article}
858 \usepackage{color}
859 \usepackage{pdfrender}[2016/05/14]
860 \begin{document}
861 Hello World
862 \newpage
863 Start
864 \textpdfrender{%
865   TextRenderingMode=1,%
866   LineWidth=.1,%
867   LineCapStyle=2,%
868   LineJoinStyle=1,%
869   MiterLimit=1.2,%
870   LineDashPattern=[2 2]0,%
871   RenderingIntent=Saturation,%
872 }{Hello\newpage World}
873 Stop
874 \par
875 \newlength{\LineWidth}
876 \setlength{\LineWidth}{.5pt}
877 Start
878 \textpdfrender{%
879   FillColor=yellow,%
880   StrokeColor=[cmyk]{1,.5,0,0},%
881   TextRenderingMode=FillStroke,%
882   LineWidth=.5\LineWidth,%
883   LineCapStyle=Round,%
884   LineJoinStyle=Bevel,%
885 }{Out-\par\newpage line}
886 Stop
887 \end{document}
888 \</test2>

```

3.3 Further tests

Robustness and bookmarks.

```

889 \<test3>
890 \NeedsTeXFormat{LaTeX2e}
891 \ProvidesFile{pdfrender-test3.tex}[2016/05/14]
892 \documentclass{article}
893 \usepackage{pdfrender}[2016/05/14]
894 \usepackage{hyperref}
895 \usepackage{bookmark}
896 \begin{document}
897 \tableofcontents

```

```

898 \section{%
899   \textpdfrenderer{%
900     TextRenderingMode=1,%
901     LineCapStyle=2,%
902     LineJoinStyle=1,%
903     MiterLimit=1.2,%
904     LineDashPattern=[2 2]0,%
905     RenderingIntent=Saturation,%
906   }{Hello World}%
907 }
908 \end{document}
909 /test3)

Color algorithm if \pdfmatch is not available.

910 (*test4)
911 \NeedsTeXFormat{LaTeX2e}
912 \ProvidesFile{pdfrenderer-test4.tex}[2016/05/14]
913 \documentclass[12pt]{article}
914 \usepackage{pdfrenderer}[2016/05/14]
915 \usepackage{color}
916 \usepackage{qstest}
917 \IncludeTests{*}
918 \LogTests{log}{*}{*}
919 \makeatletter
920 \newcommand*{\CheckColor}[1]{%
921   \Expect{#1}*{\current@color}%
922 }
923 \makeatother
924 \begin{document}
925   \begin{qstest}{color}{color}%
926     \CheckColor{0 g 0 G}%
927     \Huge\bfseries
928     \noindent
929     \textpdfrenderer{%
930       TextRenderingMode=2,%
931       LineWidth=.5,%
932       FillColor=yellow,%
933       StrokeColor=blue,%
934     }{%
935       \CheckColor{0 0 1 0 k 0 0 1 RG}%
936       Blue(Yellow)\%
937       \textpdfrenderer{%
938         FillColor=green,%
939       }{%
940         \CheckColor{0 1 0 rg 0 0 1 RG}%
941         Blue(Green)%
942       }\%
943       \CheckColor{0 0 1 0 k 0 0 1 RG}%
944       Blue(Yellow)\%
945       \textpdfrenderer{%
946         StrokeColor=red,%
947       }{%
948         \CheckColor{0 0 1 0 k 1 0 0 RG}%
949         Red(Yellow)%
950       }\%
951       \CheckColor{0 0 1 0 k 0 0 1 RG}%
952       Blue(Yellow) %
953     }%
954   \end{qstest}%

```

```

955 \begin{qstest}{colorlast}{colorlast}%
956 \makeatletter
957 \def\Test#1#2#3{%
958 \begingroup
959 \def\current@color{#1}%
960 \textpdfrender{#2}{%
961 \CheckColor{#3}%
962 }%
963 \endgroup
964 }%
965 \Test{1 g 0 0 1 RG 0 0 1 0 k 0.5 G}%
966 {StrokeColor=green}%
967 {0 0 1 0 k 0 1 0 RG}%
968 \Test{1 g 0 0 1 RG 0 0 1 0 k 0.5 G}%
969 {FillColor=red}%
970 {1 0 0 rg 0.5 G}%
971 \end{qstest}%
972 \end{document}
973 \end{test4}

```

3.4 Compatibility with plain T_EX

```

974 (*test5)
975 \input luatex85.sty
976 \pdfoutput=1 %
977 \hsize=6.5in
978 \vsize=8.9in
979 \pdfpagewidth=\hsize
980 \pdfpageheight=\vsize
981 \parfillskip=0pt plus 1fil\relax
982 \input pdfrender.sty\relax
983 \catcode'\{=1 %
984 \catcode'\}=2 %
985 \let\OrgMakeFootLine\makefootline
986 \def\makefootline{%
987 \begingroup\normalcolor\OrgMakeFootLine\endgroup
988 }
989 \font\f=ec-lmr10 scaled 3000\relax
990 \f
991 Before %
992 \textpdfrender{%
993 TextRenderingMode=1,%
994 LineWidth=.1,%
995 }{Hello\par\vfill\penalty-10000 World} %
996 After %
997 \par
998 \vfill
999 \penalty-10000 %
1000 \csname @@end\endcsname\end
1001 \end{test5}

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

¹<http://ctan.org/pkg/pdfrender>

[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 T_EX Files” ([CTAN:tds/tds.pdf](#)). Directories with `texmf` in their name are usually organized this way.

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 T_EX:

```
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>	→ <code>tex/generic/oberdiek/pdfrender.sty</code>
<code>pdfrender.pdf</code>	→ <code>doc/latex/oberdiek/pdfrender.pdf</code>
<code>test/pdfrender-test1.tex</code>	→ <code>doc/latex/oberdiek/test/pdfrender-test1.tex</code>
<code>test/pdfrender-test2.tex</code>	→ <code>doc/latex/oberdiek/test/pdfrender-test2.tex</code>
<code>test/pdfrender-test3.tex</code>	→ <code>doc/latex/oberdiek/test/pdfrender-test3.tex</code>
<code>test/pdfrender-test4.tex</code>	→ <code>doc/latex/oberdiek/test/pdfrender-test4.tex</code>
<code>test/pdfrender-test5.tex</code>	→ <code>doc/latex/oberdiek/test/pdfrender-test5.tex</code>
<code>pdfrender.dtx</code>	→ <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 T_EX distribution (teT_EX, miK_TE_X, ...) relies on file name databases, you must refresh these. For example, teT_EX 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 \LaTeX . The `.dtx` chooses its action depending on the format:

plain \TeX : Run `docstrip` and extract the files.

\LaTeX : Generate the documentation.

If you insist on using \LaTeX for `docstrip` (really, `docstrip` does not need \LaTeX), 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 `pdf \LaTeX` :

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

5 Catalogue

The following XML file can be used as source for the [\$\TeX\$ Catalogue](#). The elements `caption` and `description` are imported from the original XML file from the Catalogue. The name of the XML file in the Catalogue is `pdfrender.xml`.

```
1002 (*catalogue)
1003 <?xml version='1.0' encoding='us-ascii'?>
1004 <!DOCTYPE entry SYSTEM 'catalogue.dtd'>
1005 <entry datestamp='$Date$' modifier='$Author$' id='pdfrender'>
1006   <name>pdfrender</name>
1007   <caption>Control rendering parameters.</caption>
1008   <authorref id='auth:oberdiek'/>
1009   <copyright owner='Heiko Oberdiek' year='2010'/>
1010   <license type='lppl1.3'/>
1011   <version number='1.3'/>
1012   <description>
1013     The package provides interfaces for the user to control PDF
1014     parameters, such as line width or text rendering mode. The
1015     control operations work in a manner very similar to that of the
1016     <xref refid='color'>color</xref> package.
1017   </p>
1018   The package is part of the <xref refid='oberdiek'>oberdiek</xref> bundle.
```

```

1019 </description>
1020 <documentation details='Package documentation'
1021     href='ctan:/macros/latex/contrib/oberdiek/pdfrender.pdf' />
1022 <ctan file='true' path='/macros/latex/contrib/oberdiek/pdfrender.dtx' />
1023 <miktex location='oberdiek' />
1024 <texlive location='oberdiek' />
1025 <install path='/macros/latex/contrib/oberdiek/oberdiek.tds.zip' />
1026 </entry>
1027 </catalogue>

```

6 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].

7 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>.

8 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 `iniTEX` added.

[2016/05/14 v1.3]

- Use package `luatex85` for compatibility with new `LuaTEX`.

9 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; plain numbers refer to the code lines where the entry is used.

Symbols		C	
<code>\#</code>	758	<code>\catcode</code>	2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 69, 70, 72, 73, 74, 78, 79, 80, 81, 82, 83, 84, 87, 88, 90, 91, 92, 93, 97, 99, 756, 757, 758, 759, 794, 803, 811, 815, 832, 833, 834, 983, 984
<code>\%</code>	834	<code>\chardef</code>	327, 601, 608, 615
<code>\.</code>	481, 520, 521, 524, 528, 530, 649, 650, 651, 652, 658, 659, 660, 661	<code>\CheckColor</code>	920, 926, 935, 940, 943, 948, 951, 961
<code>\@</code>	759, 832	<code>\color</code>	595, 598
<code>\@PackageError</code>	584	<code>\color@begingroup</code>	201, 203, 204
<code>\@PackageInfo</code>	333	<code>\color@endbox</code>	205, 216
<code>\@PackageInfoNoLine</code>	178	<code>\color@endgroup</code>	202, 205, 245
<code>\@PackageWarning</code>	171, 187, 551, 572, 626, 637	<code>\color@hbox</code>	203, 216
<code>\@PackageWarningNoLine</code>	152	<code>\color@setgroup</code>	200, 245, 248
<code>\@ehc</code>	431, 587	<code>\color@vbox</code>	204
<code>\@empty</code>	447	<code>\count@</code>	761, 790, 794, 796, 797, 801, 803, 804, 805, 809, 811, 814, 815, 819, 820
<code>\@firstofone</code>	730, 737, 767, 770	<code>\countdef</code>	761
<code>\@gobble</code>	727, 764, 772	<code>\csname</code>	14, 21, 50, 66, 76, 132, 134, 137, 139, 142, 147, 156, 322, 323, 324, 326, 328, 331, 335, 337, 339, 340, 342, 344, 347, 349, 350, 352, 356, 358, 361, 363, 365, 369, 371, 372, 377, 384, 387, 392, 399, 400, 403, 411, 417, 418, 422, 423, 430, 432, 436, 437, 445, 450, 456, 460, 461, 463, 760, 763, 766, 769, 824, 851, 1000
<code>\@ifnextchar</code>	595	<code>\current@color</code>	617, 630, 641, 685, 688, 705, 708, 736, 737, 742, 921, 959
<code>\@ifundefined</code>	583		
<code>\@nil</code>	724, 733, 737		
<code>\@nodocument</code>	236, 254		
<code>\@undefined</code>	58, 122		
<code>\[</code>	527		
<code>\]</code>	497, 726, 745, 833, 936, 942, 944, 950		
<code>\{</code>	756, 983		
<code>\}</code>	757, 984		
<code>\]</code>	529		
A		D	
<code>\advance</code>	797, 805, 820	<code>\DeclareRobustCommand</code>	288, 305
<code>\afterassignment</code>	490	<code>\define@key</code>	414
<code>\AfterEndPreamble</code>	271, 276	<code>\dimen</code>	491, 500, 503
<code>\aftergroup</code>	29, 346, 360		
<code>\AfterPackage</code>	268, 269, 270		
<code>\AtBeginDocument</code>	264		
B			
<code>\begin</code>	860, 896, 924, 925, 955		
<code>\bfseries</code>	927		
<code>\body</code>	776, 780		

<code>\dimexpr</code>	486	<code>\kvsetkeys</code>	295
<code>\documentclass</code>	857, 892, 913		
E			
<code>\empty</code>	17, 18		
<code>\end</code> ..	852, 887, 908, 954, 971, 972, 1000		
<code>\endcsname</code>	14, 21, 50, 66, 76, 132, 134, 137, 139, 142, 147, 156, 322, 323, 324, 326, 328, 331, 335, 337, 339, 340, 342, 344, 347, 349, 350, 352, 356, 358, 361, 363, 365, 369, 371, 372, 377, 384, 387, 392, 399, 400, 403, 411, 417, 418, 422, 423, 430, 432, 436, 437, 445, 450, 456, 460, 461, 463, 760, 763, 766, 769, 824, 851, 1000		
<code>\endgraf</code>	202, 210		
<code>\endingput</code>	29, 129		
<code>\endlinechar</code>	4, 35, 71, 77, 89		
<code>\errmessage</code>	813		
<code>\Expect</code>	921		
F			
<code>\f</code>	989, 990		
<code>\font</code>	989		
G			
<code>\gdef</code>	220, 337		
H			
<code>\hbox</code>	203		
<code>\hsize</code>	977, 979		
<code>\Huge</code>	927		
I			
<code>\iffalse</code>	140		
<code>\ifnum</code>	436, 623, 634, 666, 747, 796, 804, 811, 819		
<code>\ifpdf</code>	169		
<code>\ifPdfRender@Match</code>	145, 434, 683, 703, 720		
<code>\ifPdfRender@Stack</code>	144, 325		
<code>\ifPdfRender@Values</code>	315, 382		
<code>\iftrue</code>	135		
<code>\ifx</code>	15, 18, 21, 50, 58, 61, 122, 147, 227, 236, 246, 254, 385, 447, 497, 543, 550, 564, 571, 602, 603, 611, 625, 636, 673, 726, 745, 760, 763, 766, 769, 824		
<code>\immediate</code>	23, 52		
<code>\includeTests</code>	917		
<code>\input</code>	161, 825, 975, 982		
<code>\iterate</code>	777, 779, 781		
K			
<code>\kv@key</code>	384, 387		
<code>\kv@parse@normalized</code>	383		
<code>\kv@value</code>	385, 387		
L			
<code>\LineWidth</code>	875, 876, 882		
<code>\LoadCommand</code>	825, 835		
<code>\LogTests</code>	918		
<code>\loop</code>	775, 791, 802, 810		
<code>\ltx@empty</code>	293, 294, 432, 543, 544, 548, 550, 564, 565, 569, 571, 597, 602, 603, 611, 625, 636, 673, 681, 701		
<code>\ltx@firstofone</code>	590		
<code>\ltx@firstoftwo</code>	280, 283, 300		
<code>\ltx@GlobalAppendToMacro</code>	229, 248, 375, 379		
<code>\ltx@gobble</code>	588		
<code>\ltx@gobbletwo</code>	389		
<code>\ltx@ifpackagelater</code>	151		
<code>\ltx@ifpackageloaded</code>	199		
<code>\ltx@ifUndefined</code>	170, 177, 190, 263, 266, 274, 280, 282, 299, 401, 409, 455, 483		
<code>\ltx@ifundefined</code>	206, 209, 219		
<code>\ltx@pkgextension</code>	156		
<code>\ltx@secondoftwo</code>	283, 300		
<code>\ltx@space</code>	179, 466, 467, 606, 613, 618, 685, 688, 705, 708		
M			
<code>\makeatletter</code>	919, 956		
<code>\makeatother</code>	923		
<code>\makefootline</code>	985, 986		
<code>\MessageBreak</code>	153, 154, 155, 156, 180, 428, 552, 573, 585		
N			
<code>\NeedsTeXFormat</code>	855, 890, 911		
<code>\newcommand</code> ...	194, 195, 287, 304, 920		
<code>\newlength</code>	875		
<code>\newpage</code>	862, 872, 885		
<code>\next</code>	781, 783, 785		
<code>\noindent</code>	928		
<code>\normalcolor</code>	220, 226, 229, 987		
<code>\number</code>	334, 816		
O			
<code>\OrgMakeFootLine</code>	985, 987		
P			
<code>\PackageError</code>	427		
<code>\PackageInfo</code>	26		
<code>\par</code>	210, 874, 885, 995, 997		
<code>\parfillskip</code>	981		
<code>\pdfcolorstack</code>	341, 351		
<code>\pdfcolorstackinit</code>	172, 329		
<code>\pdfextension</code>	122		
<code>\pdflastmatch</code>	668, 671		
<code>\pdfliteral</code>	357, 364		
<code>\pdfmatch</code>	179, 436, 666		

\pdfoutput	976	\PdfRender@PatchNormalColor	218, 259, 292
\pdfpageheight	980	\PdfRender@Pattern	666, 684, 704
\pdfpagewidth	979	\PdfRender@PatternFillColor	647, 684
\pdfrender	2, 191, 194, 282, 310	\PdfRender@PatternStrokeColor	656, 704
\PdfRender@@FilterOp	741, 744	\PdfRender@PostProcessLineWidth	488
\PdfRender@@PostProcessLineWidth	490, 496	\PdfRender@relax	492, 495
\PdfRender@@TryColor	595, 597	\PdfRender@RequirePackage	146, 166, 167, 168, 278
\PdfRender@AtEnd	95, 96, 126, 128, 197, 721, 753	\PdfRender@Reset	421
\PdfRender@AtEndHook	125, 127, 149, 150	\PdfRender@Set	373, 380, 416
\PdfRender@Color	542, 543, 548, 563, 564, 569, 598	\PdfRender@SetColor	296, 600
\PdfRender@ColorAvailable	546, 567, 582	\PdfRender@SetValidate	405, 434
\PdfRender@ColorSetGroupHook	214, 242, 249, 379	\PdfRender@SetValidateValues	394, 454
\PdfRender@Current	393, 444, 447, 455, 457, 461	\PdfRender@Stacktrue	175
\PdfRender@CurrentColor	605, 612, 630, 641	\PdfRender@String	666, 670, 673, 685, 705
\PdfRender@CurrentLineWidth	491, 502	\PdfRender@StrokeColor	294, 565, 571, 603, 606, 611, 613, 618, 636, 701, 706, 716
\PdfRender@define@key	391, 398, 409, 540, 561	\PdfRender@temp	223, 227, 242, 246, 724, 733, 737
\PdfRender@dimexpr	484, 486, 491	\PdfRender@TestBox	215, 224, 243, 547, 568
\PdfRender@ErrorInvalidValue	426, 439, 448, 458	\PdfRender@texorpdfstring	279, 291, 308
\PdfRender@FillColor	293, 544, 550, 602, 606, 613, 618, 625, 681, 686, 696	\PdfRender@TryColor	548, 569, 594
\PdfRender@FilterOp	696, 716, 740	\PdfRender@Valuesfalse	396
\PdfRender@FindOp	690, 691, 692, 693, 694, 695, 710, 711, 712, 713, 714, 715, 723	\PdfRender@Valuetrue	317
\PdfRender@GetFillColor	549, 624, 680	\penalty	995, 999
\PdfRender@GetStrokeColor	570, 635, 700	\ProvidesFile	856, 891, 912
\PdfRender@let	492, 494, 496	\ProvidesPackage	19, 67
\PdfRender@MatchPattern	665, 686, 706	R	
\PdfRender@Matchtrue	184	\RangeCatcodeCheck	808, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847
\PdfRender@NeedsCurrentColor	601, 608, 615, 623, 634	\RangeCatcodeInvalid	800, 828, 829, 830, 831
\PdfRender@NewClass	318, 320, 480, 519, 523, 526	\repeat	775, 787, 798, 806, 821
\PdfRender@NewClassValues	316, 468, 509, 514, 532	\RequirePackage	123, 164
\PdfRender@newif	131, 144, 145, 315, 321	\RestoreCatcodes	789, 792, 793, 848
\PdfRender@NormalColorHook	213, 220, 223, 230, 375	S	
\PdfRender@OP	689, 709, 729, 742, 744	\section	898
\PdfRender@OpName	467, 534	\set@color	200, 225, 244, 620, 631, 642
\PdfRender@OpValue	466, 470, 480, 509, 514, 519, 523, 526	\setbox	216
\PdfRender@PatchColor	258, 262, 264, 268, 269, 271, 276	\setlength	876
\PdfRender@PatchColorSetGroup	240, 260	\space	814, 815, 823
		\strip@prefix	668, 671
		\strip@pt	503
		T	
		\tableofcontents	897
		\Test	827, 850, 957, 965, 968
		\texorpdfstring	280
		\textpdfrender	2, 192, 195, 299, 864, 878, 899, 929, 937, 945, 960, 992

<code>\the</code>	77, 78, 79, 80, 81, 82, 83, 84, 97, 794, 814, 815	V	
<code>\TMP@EnsureCode</code>	94, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121	<code>\vbox</code>	204
		<code>\vfill</code>	995, 998
		<code>\vsize</code>	978, 980
		W	
		<code>\write</code>	23, 52
		U	
<code>\usepackage</code>	858, 859, 893, 894, 895, 914, 915, 916	X	
		<code>\x</code>	14, 15, 18, 22, 26, 28, 51, 56, 66, 75, 87, 501, 506