

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	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	20
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	24
5	Catalogue	25
6	Acknowledgement	26
7	References	26

8 History	26
[2010/01/26 v1.0]	26
[2010/01/27 v1.1]	26
[2010/01/28 v1.2]	26
9 Index	26

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>TeX 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 TeX 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 [2010/01/28 v1.2 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}% |
122 \def\PdfRender@AtEndHook{}
123 \expandafter\def\expandafter\PdfRender@AtEnd\expandafter{%
124   \expandafter\PdfRender@AtEndHook
125   \PdfRender@AtEnd
126   \endinput
127 }

```

2.1 Look for pdfTeX, its mode and features

\PdfRender@newif

```

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

```

\ifPdfRender@Stack

```

141 \PdfRender@newif{Stack}

```

```

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

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

163 \PdfRender@RequirePackage{ifpdf}[2010/01/28]
164 \PdfRender@RequirePackage{infwarerr}[2007/09/09]
165 \PdfRender@RequirePackage{ltxcmds}[2010/01/28]

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

\pdfrender
188 \def\pdfrender#1{%

\textpdfrender
189 \long\def\textpdfrender#1#2{#2}%

190 }%

```

```

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

\textpdfrender
192 \newcommand{\textpdfrender}[2]{#2}%

193 }%
194 \expandafter\PdfRender@AtEnd
195 \fi%

```

2.2 Enable color support of L^AT_EX

```

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

```

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
210 \def\PdfRender@NormalColorHook{}

\PdfRender@ColorSetGroupHook
211 \def\PdfRender@ColorSetGroupHook{}

\PdfRender@TestBox
212 \def\PdfRender@TestBox#1{%
213 \setbox0=\color@hbox#1\color@endbox
214 }

\PdfRender@PatchNormalColor
215 \def\PdfRender@PatchNormalColor{%
216 \ltx@ifundefined{normalcolor}{%
217 \gdef\normalcolor{\PdfRender@NormalColorHook}%
218 }{%
219 \begingroup
220 \def\PdfRender@NormalColorHook{\let\PdfRender@temp=Y}%
221 \PdfRender@TestBox{%
222 \let\set@color\relax
223 \normalcolor
224 \ifx\PdfRender@temp Y%
225 \else

```



```

226         \ltx@GlobalAppendToMacro\normalcolor{%
227         \PdfRender@NormalColorHook
228     }%
229     \fi
230 }%
231 \endgroup
232 }%
233 \ifx\@nodocument\relax
234     \global\let\PdfRender@PatchNormalColor\relax
235 \fi
236 }%

```

\PdfRender@PatchColorSetGroup

```

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

```

\PdfRender@PatchColor

```

255 \def\PdfRender@PatchColor{%
256     \PdfRender@PatchNormalColor
257     \PdfRender@PatchColorSetGroup
258 }

259 \PdfRender@PatchColor
260 \ltx@ifundefined{AtBeginDocument}{}{%
261     \AtBeginDocument{\PdfRender@PatchColor}%
262 }

```

\AfterPackage is provided by package scrfile.

```

263 \ltx@ifundefined{AfterPackage}{%
264 }{%
265     \AfterPackage{color}{\PdfRender@PatchColor}%
266     \AfterPackage{xcolor}{\PdfRender@PatchColor}%
267     \AfterPackage{etoolbox}{%
268         \AfterEndPreamble{\PdfRender@PatchColor}%
269     }%
270 }%

```

\AfterEndPreamble is provided by package etoolbox.

```

271 \ltx@ifundefined{AfterEndPreamble}{%
272 }{%
273     \AfterEndPreamble{\PdfRender@PatchColor}%
274 }%

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

```

\PdfRender@texorpdfstring

```
276 \def\PdfRender@texorpdfstring{%
277   \ltx@ifundefined{texorpdfstring}\ltx@firstoftwo\texorpdfstring
278 }
```

\pdfrender

```
279 \ltx@ifundefined{DeclareRobustCommand}%
280 \ltx@firstoftwo\ltx@secondoftwo
281 {%
282   \def\pdfrender#1%
283 }{%
284   \newcommand{\pdfrender}{}%
285   \DeclareRobustCommand*\pdfrender[1]%
286 }%
287 {%
288   \PdfRender@texorpdfstring{%
289     \PdfRender@PatchNormalColor
290     \global\let\PdfRender@FillColor\ltx@empty
291     \global\let\PdfRender@StrokeColor\ltx@empty
292     \kvsetkeys{PDFRENDER}{#1}%
293     \PdfRender@SetColor
294   }{}%
295 }
```

\textpdfrender

```
296 \ltx@ifundefined{DeclareRobustCommand}%
297 \ltx@firstoftwo\ltx@secondoftwo
298 {%
299   \long\def\textpdfrender#1#2%
300 }{%
301   \newcommand{\textpdfrender}{}%
302   \DeclareRobustCommand{\textpdfrender}[2]%
303 }%
304 {%
305   \PdfRender@texorpdfstring{%
306     \begingroup
307       \pdfrender{#1}%
308       #2%
309     \endgroup
310   }{#2}%
311 }
```

\ifPdfRender@Values

```
312 \PdfRender@newif{Values}
```

\PdfRender@NewClassValues

```
313 \def\PdfRender@NewClassValues#1#2#3#4{%
314   \PdfRender@Valuestrue
315   \PdfRender@NewClass{#1}{#2}{#3}{#4}{}%
316 }
```

\PdfRender@NewClass

```
317 \def\PdfRender@NewClass#1#2#3#4#5{%
318   \PdfRender@newif{Active#1}%
319   \expandafter\def\csname PdfRender@Default#1\endcsname{#2}%
320   \expandafter\let\csname PdfRender@Current#1\expandafter\endcsname
321     \csname PdfRender@Default#1\endcsname
```

```

322 \ifPdfRender@Stack
323   \expandafter\edef\csname PdfRender@Init#1\endcsname{%
324     \global\chardef
325     \expandafter\noexpand\csname PdfRender@Stack#1\endcsname=%
326     \noexpand\pdfcolorstackinit page direct{%
327       \noexpand#3%
328       \expandafter\noexpand\csname PdfRender@Default#1\endcsname
329     }\relax
330     \noexpand\@PackageInfo{pdfrender}{%
331       New color stack '#1' = \noexpand\number
332       \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
333     }%
334     \gdef\expandafter\noexpand\csname PdfRender@Init#1\endcsname{%}%
335   }%
336   \expandafter\edef\csname PdfRender@Set#1\endcsname{%
337     \expandafter\noexpand\csname PdfRender@Init#1\endcsname
338     \noexpand\pdfcolorstack
339     \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
340     push{%
341       #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
342     }%
343     \noexpand\aftergroup
344     \expandafter\noexpand\csname PdfRender@Reset#1\endcsname
345   }%
346   \expandafter\edef\csname PdfRender@Reset#1\endcsname{%
347     \expandafter\noexpand\csname PdfRender@Init#1\endcsname
348     \noexpand\pdfcolorstack
349     \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
350     pop\relax
351   }%
352 \else
353   \expandafter\edef\csname PdfRender@Set#1\endcsname{%
354     \noexpand\pdfliteral direct{%
355       #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
356     }%
357     \noexpand\aftergroup
358     \expandafter\noexpand\csname PdfRender@Reset#1\endcsname
359   }%
360   \expandafter\edef\csname PdfRender@Reset#1\endcsname{%
361     \noexpand\pdfliteral direct{%
362       #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
363     }%
364   }%
365 \fi
366 \expandafter\edef\csname PdfRender@Normal#1\endcsname{%
367   \let
368   \expandafter\noexpand\csname PdfRender@Current#1\endcsname
369   \expandafter\noexpand\csname PdfRender@Default#1\endcsname
370   \noexpand\pdfcolorstack
371 }%
372 \expandafter\ltx@GlobalAppendToMacro\expandafter\pdfcolorstack
373 \expandafter{%
374   \csname PdfRender@Normal#1\endcsname
375 }%
376 \ltx@GlobalAppendToMacro\pdfcolorstack
377 \pdfcolorstack
378 }%
379 \ifPdfRender@Values

```

```

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

\PdfRender@define@key
406 \ltx@ifundefined{define@key}{%
407 \def\PdfRender@define@key#1#2{%
408 \expandafter\def\csname KV@#1@#2\endcsname##1%
409 }%
410 }{%
411 \let\PdfRender@define@key\define@key
412 }

\PdfRender@Set
413 \def\PdfRender@Set#1{%
414 \csname ifPdfRender@Active#1\endcsname
415 \csname PdfRender@Set#1\expandafter\endcsname
416 \fi
417 }

\PdfRender@Reset
418 \def\PdfRender@Reset#1{%
419 \csname ifPdfRender@Active#1\endcsname
420 \csname PdfRender@Reset#1\expandafter\endcsname
421 \fi
422 }

\PdfRender@ErrorInvalidValue
423 \def\PdfRender@ErrorInvalidValue#1{%
424 \PackageError{pdfrender}{%
425 Ignoring parameter setting for ‘#1’\MessageBreak
426 because of invalid value %
427 ‘\csname PdfRender@Current#1\endcsname’%
428 }{@ehc
429 \expandafter\let\csname PdfRender@Current#1\endcsname\ltx@empty
430 }%

```

\PdfRender@SetValidate

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

\PdfRender@SetValidateValues

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

\PdfRender@OpValue

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

\PdfRender@OpName

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

2.4 Declare and setup parameters

```
465 \PdfRender@NewClassValues{TextRenderingMode}%
466                               {0}%
467                               {\PdfRender@OpValue{Tr}}{%
468   0=Fill,%
469   1=Stroke,%
470   2=FillStroke,%
471   3=Invisible,%
472   4=FillClip,%
473   5=StrokeClip,%
474   6=FillStrokeClip,%
475   7=Clip,%
476 }%
477 \PdfRender@NewClass{LineWidth}{1}{\PdfRender@OpValue{w}}{%
478   [0-9]+\string\.[0-9]*|\string\.[0-9]+%
```

```

479 }{}%
480 \ltx@ifundefined{dimexpr}{%
481   \def\PdfRender@dimexpr{%
482 }{}%
483   \let\PdfRender@dimexpr\dimexpr
484 }
485 \def\PdfRender@PostProcessLineWidth{%
486   \begingroup
487   \afterassignment\PdfRender@@PostProcessLineWidth
488   \dimen0=\PdfRender@dimexpr\PdfRender@CurrentLineWidth bp %
489   \PdfRender@let\PdfRender@relax\PdfRender@relax
490 }
491 \let\PdfRender@let\let
492 \let\PdfRender@relax\relax
493 \def\PdfRender@@PostProcessLineWidth#1\PdfRender@let{%
494   \ifx\#1\%
495     \endgroup
496   \else
497     \dimen0=.996264\dimen0 % 72/72.27
498     \edef\x{\endgroup
499       \def\noexpand\PdfRender@CurrentLineWidth{%
500         \strip@pt\dimen0%
501       }%
502     }%
503     \expandafter\x
504   \fi
505 }
506 \PdfRender@NewClassValues{LineCapStyle}{0}{\PdfRender@OpValue{J}}{%
507   0=Butt,%
508   1=Round,%
509   2=ProjectingSquare,%
510 }%
511 \PdfRender@NewClassValues{LineJoinStyle}{0}{\PdfRender@OpValue{j}}{%
512   0=Miter,%
513   1=Round,%
514   2=Bevel,%
515 }%
516 \PdfRender@NewClass{MiterLimit}{10}{\PdfRender@OpValue{M}}{%
517   [0-9]*[1-9][0-9]*\string\.[0-9]*|%
518   [0-9]*\string\.[0-9]*[1-9][0-9]*%
519 }{}%
520 \PdfRender@NewClass{Flatness}{0}{\PdfRender@OpValue{i}}{%
521   100(\string\.[0-9]?|[0-9][0-9](\string\.[0-9]*)?|\string\.[0-9]+%
522 }{}%
523 \PdfRender@NewClass{LineDashPattern}{[]0}{\PdfRender@OpValue{d}}{%
524   \string\[
525   ( ?([0-9]+\string\.[0-9]*|\string\.[0-9]+) ?)*%
526   \string\] ?%
527   ([0-9]+\string\.[0-9]*|\string\.[0-9]+)%
528 }{}%
529 \PdfRender@NewClassValues{RenderingIntent}%
530   {RelativeColorimetric}%
531   {\PdfRender@OpName{ri}}{%
532   AbsoluteColorimetric,%
533   RelativeColorimetric,%
534   Saturation,%
535   Perceptual,%
536 }%

```

2.5 Fill and stroke color support

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

\PdfRender@ColorAvailable
579 \def\PdfRender@ColorAvailable{%
580   \@ifundefined{set@color}{%
581     \@PackageError{pdfrender}{%
582       Ignoring color options, because neither\MessageBreak
583       package ‘color’ nor package ‘xcolor’ is loaded%
584     }\@ehc
585   \global\let\PdfRender@ColorAvailable\ltx@gobble
586 }{%
587   \global\let\PdfRender@ColorAvailable\ltx@firstofone
588 }%
589 \PdfRender@ColorAvailable
590 }
```

\PdfRender@TryColor

```
591 \def\PdfRender@TryColor{%  
592   \@ifnextchar[\color\PdfRender@@TryColor  
593 }
```

\PdfRender@@TryColor

```
594 \def\PdfRender@@TryColor#1\ltx@empty{%  
595   \expandafter\color\expandafter{\PdfRender@Color}%  
596 }
```

\PdfRender@SetColor

```
597 \def\PdfRender@SetColor{%  
598   \chardef\PdfRender@NeedsCurrentColor=0 %  
599   \ifx\PdfRender@FillColor\ltx@empty  
600     \ifx\PdfRender@StrokeColor\ltx@empty  
601       \else  
602         \edef\PdfRender@CurrentColor{%  
603           \noexpand\PdfRender@FillColor\ltx@space\PdfRender@StrokeColor  
604         }%  
605         \chardef\PdfRender@NeedsCurrentColor=1 %  
606       \fi  
607     \else  
608       \ifx\PdfRender@StrokeColor\ltx@empty  
609         \edef\PdfRender@CurrentColor{%  
610           \PdfRender@FillColor\ltx@space\noexpand\PdfRender@StrokeColor  
611         }%  
612         \chardef\PdfRender@NeedsCurrentColor=2 %  
613       \else  
614         \edef\current@color{%  
615           \PdfRender@FillColor\ltx@space\PdfRender@StrokeColor  
616         }%  
617         \set@color  
618       \fi  
619     \fi  
620     \ifnum\PdfRender@NeedsCurrentColor=1 %  
621       \PdfRender@GetFillColor  
622       \ifx\PdfRender@FillColor\ltx@empty  
623         \@PackageWarning{pdfrender}{%  
624           Cannot extract current fill color%  
625         }%  
626       \else  
627         \edef\current@color{\PdfRender@CurrentColor}%  
628         \set@color  
629       \fi  
630     \else  
631       \ifnum\PdfRender@NeedsCurrentColor=2 %  
632         \PdfRender@GetStrokeColor  
633         \ifx\PdfRender@StrokeColor\ltx@empty  
634           \@PackageWarning{pdfrender}{%  
635             Cannot extract current stroke color%  
636           }%  
637         \else  
638           \edef\current@color{\PdfRender@CurrentColor}%  
639           \set@color  
640         \fi  
641       \fi  
642     \fi  
643 }
```


\PdfRender@PatternFillColor

```
644 \edef\PdfRender@PatternFillColor{ % space
645   (%)
646   [0-9\string\.] + g|%
647   [0-9\string\.] + [0-9\string\.] + [0-9\string\.] + rg|%
648   [0-9\string\.] + [0-9\string\.] + %
649   [0-9\string\.] + [0-9\string\.] + k%
650   ) % space
651   (.*)$%
652 }
```

\PdfRender@PatternStrokeColor

```
653 \edef\PdfRender@PatternStrokeColor{ % space
654   (%)
655   [0-9\string\.] + G|%
656   [0-9\string\.] + [0-9\string\.] + [0-9\string\.] + RG|%
657   [0-9\string\.] + [0-9\string\.] + %
658   [0-9\string\.] + [0-9\string\.] + K%
659   ) % space
660   (.*)$%
661 }
```

\PdfRender@MatchPattern

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

\PdfRender@GetFillColor

```
677 \def\PdfRender@GetFillColor{%
678   \global\let\PdfRender@FillColor\ltx@empty
679   \begingroup
680     \if\PdfRender@Match
681       \let\PdfRender@Pattern\PdfRender@PatternFillColor
682       \edef\PdfRender@String{\ltx@space\current@color\ltx@space}%
683       \PdfRender@MatchPattern\PdfRender@FillColor
684     \else
685       \edef\current@color{\current@color\ltx@space}%
686       \let\PdfRender@OP\relax
687       \PdfRender@FindOp{g}0%
688       \PdfRender@FindOp{G}1%
689       \PdfRender@FindOp{rg}0%
690       \PdfRender@FindOp{RG}1%
691       \PdfRender@FindOp{k}0%
692       \PdfRender@FindOp{K}1%
693       \PdfRender@FilterOp 0\PdfRender@FillColor
694     \fi
```

```

695 \endgroup
696 }

\PdfRender@GetStrokeColor

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

717 \ifPdfRender@Match
718 \expandafter\PdfRender@AtEnd
719 \fi%

\PdfRender@FindOp

720 \def\PdfRender@FindOp#1#2{%
721 \def\PdfRender@temp##1 #1 ##2\@nil{%
722 ##1%
723 \ifx\##2\%
724 \expandafter\@gobble
725 \else
726 \PdfRender@OP{#1}#2%
727 \expandafter\@firstofone
728 \fi
729 {%
730 \PdfRender@temp##2\@nil
731 }%
732 }%
733 \edef\current@color{%
734 \@firstofone{\expandafter\PdfRender@temp\current@color} #1 \@nil
735 }%
736 }

\PdfRender@FilterOp

737 \def\PdfRender@FilterOp#1#2{%
738 \expandafter\PdfRender@@FilterOp\expandafter#1\expandafter#2%
739 \current@color\PdfRender@OP{#1}#2%
740 }

\PdfRender@@FilterOp

741 \def\PdfRender@@FilterOp#1#2#3\PdfRender@OP#4#5{%
742 \ifx\##5\%
743 \else

```

```

744 \ifnum#1=#5 %
745 \xdef#2{#3 #4}%
746 \fi
747 \expandafter\PdfRender@@FilterOp\expandafter#1\expandafter#2%
748 \fi
749 }

750 \PdfRender@AtEnd%
751 \end{package}

```

3 Test

3.1 Catcode checks for loading

```

752 \test1\
753 \catcode'\{=1 %
754 \catcode'\}=2 %
755 \catcode'\#=6 %
756 \catcode'\@=11 %
757 \expandafter\ifx\csname count@\endcsname\relax
758 \countdef\count@=255 %
759 \fi
760 \expandafter\ifx\csname @gobble\endcsname\relax
761 \long\def@gobble#1{}%
762 \fi
763 \expandafter\ifx\csname @firstofone\endcsname\relax
764 \long\def@firstofone#1{#1}%
765 \fi
766 \expandafter\ifx\csname loop\endcsname\relax
767 \expandafter@firstofone
768 \else
769 \expandafter@gobble
770 \fi
771 {%
772 \def\loop#1\repeat{%
773 \def\body{#1}%
774 \iterate
775 }%
776 \def\iterate{%
777 \body
778 \let\next\iterate
779 \else
780 \let\next\relax
781 \fi
782 \next
783 }%
784 \let\repeat=\fi
785 }%
786 \def\RestoreCatcodes{}
787 \count@=0 %
788 \loop
789 \edef\RestoreCatcodes{%
790 \RestoreCatcodes
791 \catcode\the\count@=\the\catcode\count@\relax
792 }%
793 \ifnum\count@<255 %
794 \advance\count@ 1 %

```

```

795 \repeat
796
797 \def\RangeCatcodeInvalid#1#2{%
798   \count@=#1\relax
799   \loop
800     \catcode\count@=15 %
801   \ifnum\count@<#2\relax
802     \advance\count@ 1 %
803   \repeat
804 }
805 \def\RangeCatcodeCheck#1#2#3{%
806   \count@=#1\relax
807   \loop
808     \ifnum#3=\catcode\count@
809   \else
810     \errmessage{%
811       Character \the\count@\space
812       with wrong catcode \the\catcode\count@\space
813       instead of \number#3%
814     }%
815   \fi
816   \ifnum\count@<#2\relax
817     \advance\count@ 1 %
818   \repeat
819 }
820 \def\space{ }
821 \expandafter\ifx\csname LoadCommand\endcsname\relax
822   \def\LoadCommand{\input pdfrender.sty\relax}%
823 \fi
824 \def\Test{%
825   \RangeCatcodeInvalid{0}{47}%
826   \RangeCatcodeInvalid{58}{64}%
827   \RangeCatcodeInvalid{91}{96}%
828   \RangeCatcodeInvalid{123}{255}%
829   \catcode'\@=12 %
830   \catcode'\=0 %
831   \catcode'\%=14 %
832   \LoadCommand
833   \RangeCatcodeCheck{0}{36}{15}%
834   \RangeCatcodeCheck{37}{37}{14}%
835   \RangeCatcodeCheck{38}{47}{15}%
836   \RangeCatcodeCheck{48}{57}{12}%
837   \RangeCatcodeCheck{58}{63}{15}%
838   \RangeCatcodeCheck{64}{64}{12}%
839   \RangeCatcodeCheck{65}{90}{11}%
840   \RangeCatcodeCheck{91}{91}{15}%
841   \RangeCatcodeCheck{92}{92}{0}%
842   \RangeCatcodeCheck{93}{96}{15}%
843   \RangeCatcodeCheck{97}{122}{11}%
844   \RangeCatcodeCheck{123}{255}{15}%
845   \RestoreCatcodes
846 }
847 \Test
848 \csname @@end\endcsname
849 \end
850 /test1}

```

3.2 Simple test file

```

851 (*test2)
852 \NeedsTeXFormat{LaTeX2e}
853 \ProvidesFile{pdfrender-test2.tex}[2010/01/28]
854 \documentclass{article}
855 \usepackage{color}
856 \usepackage{pdfrender}[2010/01/28]
857 \begin{document}
858 Hello World
859 \newpage
860 Start
861 \textpdfrender{%
862   TextRenderingMode=1,%
863   LineWidth=.1,%
864   LineCapStyle=2,%
865   LineJoinStyle=1,%
866   MiterLimit=1.2,%
867   LineDashPattern=[2 2]0,%
868   RenderingIntent=Saturation,%
869 }{Hello\newpage World}
870 Stop
871 \par
872 \newlength{\LineWidth}
873 \setlength{\LineWidth}{.5pt}
874 Start
875 \textpdfrender{%
876   FillColor=yellow,%
877   StrokeColor=[cmYk]{1,.5,0,0},%
878   TextRenderingMode=FillStroke,%
879   LineWidth=.5\LineWidth,%
880   LineCapStyle=Round,%
881   LineJoinStyle=Bevel,%
882 }{Out-\par\newpage line}
883 Stop
884 \end{document}
885 /test2)

```

3.3 Further tests

Robustness and bookmarks.

```

886 (*test3)
887 \NeedsTeXFormat{LaTeX2e}
888 \ProvidesFile{pdfrender-test3.tex}[2010/01/28]
889 \documentclass{article}
890 \usepackage{pdfrender}[2010/01/28]
891 \usepackage{hyperref}
892 \usepackage{bookmark}
893 \begin{document}
894 \tableofcontents
895 \section{%
896   \textpdfrender{%
897     TextRenderingMode=1,%
898     LineCapStyle=2,%
899     LineJoinStyle=1,%
900     MiterLimit=1.2,%
901     LineDashPattern=[2 2]0,%
902     RenderingIntent=Saturation,%
903   }{Hello World}%
904 }
905 \end{document}

```

```

906 </test3>
      Color algorithm if \pdfmatch is not available.
907 <*test4>
908 \NeedsTeXFormat{LaTeX2e}
909 \ProvidesFile{pdfrender-test4.tex}[2010/01/28]
910 \documentclass[12pt]{article}
911 \usepackage{pdfrender}[2010/01/28]
912 \usepackage{color}
913 \usepackage{qstest}
914 \IncludeTests{*}
915 \LogTests{log}{*}{*}
916 \makeatletter
917 \newcommand*{\CheckColor}[1]{%
918   \Expect{#1}*{\current@color}%
919 }
920 \makeatother
921 \begin{document}
922   \begin{qstest}{color}{color}%
923     \CheckColor{0 g 0 G}%
924     \Huge\bfseries
925     \noindent
926     \textpdfrender{%
927       TextRenderingMode=2,%
928       LineWidth=.5,%
929       FillColor=yellow,%
930       StrokeColor=blue,%
931     }{%
932       \CheckColor{0 0 1 0 k 0 0 1 RG}%
933       Blue(Yellow)\%
934       \textpdfrender{%
935         FillColor=green,%
936       }{%
937         \CheckColor{0 1 0 rg 0 0 1 RG}%
938         Blue(Green)%
939       }\%
940       \CheckColor{0 0 1 0 k 0 0 1 RG}%
941       Blue(Yellow)\%
942       \textpdfrender{%
943         StrokeColor=red,%
944       }{%
945         \CheckColor{0 0 1 0 k 1 0 0 RG}%
946         Red(Yellow)%
947       }\%
948       \CheckColor{0 0 1 0 k 0 0 1 RG}%
949       Blue(Yellow) %
950     }%
951   \end{qstest}%
952   \begin{qstest}{colorlast}{colorlast}%
953     \makeatletter
954     \def\Test#1#2#3{%
955       \begingroup
956         \def\current@color{#1}%
957         \textpdfrender{#2}{%
958           \CheckColor{#3}%
959         }%
960       \endgroup
961     }%
962     \Test{1 g 0 0 1 RG 0 0 1 0 k 0.5 G}%

```

```

963      {StrokeColor=green}%
964      {0 0 1 0 k 0 1 0 RG}%
965      \Test{1 g 0 0 1 RG 0 0 1 0 k 0.5 G}%
966      {FillColor=red}%
967      {1 0 0 rg 0.5 G}%
968      \end{qstest}%
969 \end{document}
970 \end{test4}

```

3.4 Compatibility with plain T_EX

```

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

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

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

[CTAN:macros/latex/contrib/oberdiek/pdfrender.pdf](http://ctan.org/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](http://ctan.org/install/macros/latex/contrib/oberdiek.tds.zip)

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

¹ftp://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 `TEX`:

```
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):

```
pdfrender.sty          → tex/generic/oberdiek/pdfrender.sty
pdfrender.pdf          → doc/latex/oberdiek/pdfrender.pdf
test/pdfrender-test1.tex → doc/latex/oberdiek/test/pdfrender-test1.tex
test/pdfrender-test2.tex → doc/latex/oberdiek/test/pdfrender-test2.tex
test/pdfrender-test3.tex → doc/latex/oberdiek/test/pdfrender-test3.tex
test/pdfrender-test4.tex → doc/latex/oberdiek/test/pdfrender-test4.tex
test/pdfrender-test5.tex → doc/latex/oberdiek/test/pdfrender-test5.tex
pdfrender.dtx          → source/latex/oberdiek/pdfrender.dtx
```

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 `TEX` distribution (`teTEX`, `mikTEX`, ...) relies on file name databases, you must refresh these. For example, `teTEX` 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 L^AT_EX. The .dtx chooses its action depending on the format:

plain T_EX: Run docstrip and extract the files.

L^AT_EX: Generate the documentation.

If you insist on using L^AT_EX for docstrip (really, docstrip does not need L^AT_EX), 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 Catalogue

The following XML file can be used as source for the [T_EX 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`.

```
998 (*catalogue)
999 <?xml version='1.0' encoding='us-ascii'?>
1000 <!DOCTYPE entry SYSTEM 'catalogue.dtd'>
1001 <entry datestamp='$Date$' modifier='$Author$' id='pdfrender'>
1002   <name>pdfrender</name>
1003   <caption>Control rendering parameters.</caption>
1004   <authorref id='auth:oberdiek' />
1005   <copyright owner='Heiko Oberdiek' year='2010' />
1006   <license type='lppl1.3' />
1007   <version number='1.2' />
1008   <description>
1009     The package provides interfaces for the user to control PDF
1010     parameters, such as line width or text rendering mode. The
1011     control operations work in a manner very similar to that of the
1012     <xref refid='color'>color</xref> package.
1013   </p>
1014   The package is part of the <xref refid='oberdiek'>oberdiek</xref> bundle.
1015 </description>
1016 <documentation details='Package documentation'
1017   href='ctan:/macros/latex/contrib/oberdiek/pdfrender.pdf' />
1018 <ctan file='true' path='/macros/latex/contrib/oberdiek/pdfrender.dtx' />
1019 <miktex location='oberdiek' />
1020 <texlive location='oberdiek' />
1021 <install path='/macros/latex/contrib/oberdiek/oberdiek.tds.zip' />
1022 </entry>
1023 </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 $\text{T}_{\text{E}}\text{X}$ and even `iniTEX` added.

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	\%	831
\#		755

<code>\.</code>	478, 517, 518, 521, 525, 527, 646, 647, 648, 649, 655, 656, 657, 658	<code>\csname</code>	14, 21, 50, 66, 76, 129, 131, 134, 136, 139, 144, 153, 319, 320, 321, 323, 325, 328, 332, 334, 336, 337, 339, 341, 344, 346, 347, 349, 353, 355, 358, 360, 362, 366, 368, 369, 374, 381, 384, 389, 396, 397, 400, 408, 414, 415, 419, 420, 427, 429, 433, 434, 442, 447, 453, 457, 458, 460, 757, 760, 763, 766, 821, 848, 996
<code>\@</code>	756, 829	<code>\current@color</code>	614, 627, 638, 682, 685, 702, 705, 733, 734, 739, 918, 956
<code>\@PackageError</code>	581	D	
<code>\@PackageInfo</code>	330	<code>\DeclareRobustCommand</code>	285, 302
<code>\@PackageInfoNoLine</code>	175	<code>\define@key</code>	411
<code>\@PackageWarning</code>	168, 184, 548, 569, 623, 634	<code>\dimen</code>	488, 497, 500
<code>\@PackageWarningNoLine</code>	149	<code>\dimexpr</code>	483
<code>\@ehc</code>	428, 584	<code>\documentclass</code>	854, 889, 910
<code>\@empty</code>	444	E	
<code>\@firstofone</code>	727, 734, 764, 767	<code>\empty</code>	17, 18
<code>\@gobble</code>	724, 761, 769	<code>\end</code>	849, 884, 905, 951, 968, 969, 996
<code>\@ifnextchar</code>	592	<code>\endcsname</code>	14, 21, 50, 66, 76, 129, 131, 134, 136, 139, 144, 153, 319, 320, 321, 323, 325, 328, 332, 334, 336, 337, 339, 341, 344, 346, 347, 349, 353, 355, 358, 360, 362, 366, 368, 369, 374, 381, 384, 389, 396, 397, 400, 408, 414, 415, 419, 420, 427, 429, 433, 434, 442, 447, 453, 457, 458, 460, 757, 760, 763, 766, 821, 848, 996
<code>\@ifundefined</code>	580	<code>\endgraf</code>	199, 207
<code>\@nil</code>	721, 730, 734	<code>\endinput</code>	29, 126
<code>\@nodocument</code>	233, 251	<code>\endlinechar</code>	4, 35, 71, 77, 89
<code>\@undefined</code>	58	<code>\errmessage</code>	810
<code>\[</code>	524	<code>\Expect</code>	918
<code>\[</code>	494, 723, 742, 830, 933, 939, 941, 947	F	
<code>\{</code>	753, 979	<code>\f</code>	985, 986
<code>\}</code>	754, 980	<code>\font</code>	985
<code>\]</code>	526	G	
A			
<code>\advance</code>	794, 802, 817	<code>\gdef</code>	217, 334
<code>\afterassignment</code>	487	H	
<code>\AfterEndPreamble</code>	268, 273	<code>\hbox</code>	200
<code>\aftergroup</code>	29, 343, 357	<code>\hsize</code>	973, 975
<code>\AfterPackage</code>	265, 266, 267	<code>\Huge</code>	924
<code>\AtBeginDocument</code>	261	I	
B			
<code>\begin</code>	857, 893, 921, 922, 952	<code>\iffalse</code>	137
<code>\bfseries</code>	924	<code>\ifnum</code>	433, 620, 631, 663, 744, 793, 801, 808, 816
<code>\body</code>	773, 777	<code>\ifpdf</code>	166
C			
<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, 753, 754, 755, 756, 791, 800, 808, 812, 829, 830, 831, 979, 980	D	
<code>\chardef</code>	324, 598, 605, 612	<code>\DeclareRobustCommand</code>	285, 302
<code>\CheckColor</code>	917, 923, 932, 937, 940, 945, 948, 958	<code>\define@key</code>	411
<code>\color</code>	592, 595	<code>\dimen</code>	488, 497, 500
<code>\color@begingroup</code>	198, 200, 201	<code>\dimexpr</code>	483
<code>\color@endbox</code>	202, 213	<code>\documentclass</code>	854, 889, 910
<code>\color@endgroup</code>	199, 202, 242	E	
<code>\color@hbox</code>	200, 213	<code>\empty</code>	17, 18
<code>\color@setgroup</code>	197, 242, 245	<code>\end</code>	849, 884, 905, 951, 968, 969, 996
<code>\color@vbox</code>	201	<code>\endcsname</code>	14, 21, 50, 66, 76, 129, 131, 134, 136, 139, 144, 153, 319, 320, 321, 323, 325, 328, 332, 334, 336, 337, 339, 341, 344, 346, 347, 349, 353, 355, 358, 360, 362, 366, 368, 369, 374, 381, 384, 389, 396, 397, 400, 408, 414, 415, 419, 420, 427, 429, 433, 434, 442, 447, 453, 457, 458, 460, 757, 760, 763, 766, 821, 848, 996
<code>\count@</code>	758, 787, 791, 793, 794, 798, 800, 801, 802, 806, 808, 811, 812, 816, 817	<code>\endgraf</code>	199, 207
<code>\countdef</code>	758	<code>\endinput</code>	29, 126
D			
<code>\csname</code>	14, 21, 50, 66, 76, 129, 131, 134, 136, 139, 144, 153, 319, 320, 321, 323, 325, 328, 332, 334, 336, 337, 339, 341, 344, 346, 347, 349, 353, 355, 358, 360, 362, 366, 368, 369, 374, 381, 384, 389, 396, 397, 400, 408, 414, 415, 419, 420, 427, 429, 433, 434, 442, 447, 453, 457, 458, 460, 757, 760, 763, 766, 821, 848, 996	<code>\endlinechar</code>	4, 35, 71, 77, 89
<code>\current@color</code>	614, 627, 638, 682, 685, 702, 705, 733, 734, 739, 918, 956	<code>\errmessage</code>	810
E			
<code>\empty</code>	17, 18	<code>\Expect</code>	918
<code>\end</code>	849, 884, 905, 951, 968, 969, 996	F	
<code>\endcsname</code>	14, 21, 50, 66, 76, 129, 131, 134, 136, 139, 144, 153, 319, 320, 321, 323, 325, 328, 332, 334, 336, 337, 339, 341, 344, 346, 347, 349, 353, 355, 358, 360, 362, 366, 368, 369, 374, 381, 384, 389, 396, 397, 400, 408, 414, 415, 419, 420, 427, 429, 433, 434, 442, 447, 453, 457, 458, 460, 757, 760, 763, 766, 821, 848, 996	G	
<code>\endgraf</code>	199, 207	<code>\gdef</code>	217, 334
<code>\endinput</code>	29, 126	H	
<code>\endlinechar</code>	4, 35, 71, 77, 89	<code>\hbox</code>	200
<code>\errmessage</code>	810	<code>\hsize</code>	973, 975
<code>\Expect</code>	918	<code>\Huge</code>	924
F			
<code>\f</code>	985, 986	I	
<code>\font</code>	985	<code>\iffalse</code>	137
G			
<code>\gdef</code>	217, 334	<code>\ifnum</code>	433, 620, 631, 663, 744, 793, 801, 808, 816
H			
<code>\hbox</code>	200	<code>\ifpdf</code>	166
<code>\hsize</code>	973, 975	C	
<code>\Huge</code>	924		
I			
<code>\iffalse</code>	137	D	
<code>\ifnum</code>	433, 620, 631, 663, 744, 793, 801, 808, 816		
<code>\ifpdf</code>	166	E	

<code>\ifPdfRender@Match</code>	<code>\next</code>
..... 142 , 431 , 680 , 700 , 717	778 , 780 , 782
<code>\ifPdfRender@Stack</code>	<code>\noindent</code>
141 , 322	925
<code>\ifPdfRender@Values</code>	<code>\normalcolor</code>
312 , 379	217 , 223 , 226 , 983
<code>\iftrue</code>	<code>\number</code>
132	331 , 813
<code>\ifx</code>	O
15 , 18 , 21 , 50 , 58 ,	<code>\OrgMakeFootLine</code>
61 , 144 , 224 , 233 , 243 , 251 , 382 ,	981 , 983
444 , 494 , 540 , 547 , 561 , 568 ,	P
599 , 600 , 608 , 622 , 633 , 670 ,	<code>\PackageError</code>
723 , 742 , 757 , 760 , 763 , 766 , 821	424
<code>\immediate</code>	<code>\PackageInfo</code>
23 , 52	26
<code>\IncludeTests</code>	<code>\par</code>
914	207 , 871 , 882 , 991 , 993
<code>\input</code>	<code>\parfillskip</code>
158 , 822 , 978	977
<code>\iterate</code>	<code>\pdfcolorstack</code>
774 , 776 , 778	338 , 348
K	<code>\pdfcolorstackinit</code>
<code>\kv@key</code>	169 , 326
381 , 384	<code>\pdflastmatch</code>
<code>\kv@parse@normalized</code>	665 , 668
380	<code>\pdfliteral</code>
<code>\kv@value</code>	354 , 361
382 , 384	<code>\pdfmatch</code>
<code>\kvsetkeys</code>	176 , 433 , 663
292	<code>\pdfoutput</code>
L	972
<code>\LineWidth</code>	<code>\pdfpageheight</code>
872 , 873 , 879	976
<code>\LoadCommand</code>	<code>\pdfpagewidth</code>
822 , 832	975
<code>\LogTests</code>	<code>\pdfrender</code>
915	2 , 188 , 191 , 279 , 307
<code>\loop</code>	<code>\PdfRender@@FilterOp</code>
772 , 788 , 799 , 807	738 , 741
<code>\ltx@empty</code>	<code>\PdfRender@@PostProcessLineWidth</code>
290 ,
291 , 429 , 540 , 541 , 545 , 547 ,	487 , 493
561 , 562 , 566 , 568 , 594 , 599 ,	<code>\PdfRender@@TryColor</code>
600 , 608 , 622 , 633 , 670 , 678 , 698	592 , 594
<code>\ltx@firstofone</code>	<code>\PdfRender@AtEnd</code>
587	... 95 , 96 , 123 , 125 , 194 , 718 , 750
<code>\ltx@firstoftwo</code>	<code>\PdfRender@AtEndHook</code> 122 , 124 , 146 , 147
277 , 280 , 297	<code>\PdfRender@Color</code>
<code>\ltx@GlobalAppendToMacro</code>	539 , 540 , 545 , 560 , 561 , 566 , 595
.....	<code>\PdfRender@ColorAvailable</code>
226 , 245 , 372 , 376
<code>\ltx@gobble</code>	543 , 564 , 579
585	<code>\PdfRender@ColorSetGroupHook</code> ...
<code>\ltx@gobbletwo</code>
386	211 , 239 , 246 , 376
<code>\ltx@ifpackagelater</code>	<code>\PdfRender@Current</code>
148
<code>\ltx@ifpackageloaded</code>	390 , 441 , 444 , 452 , 454 , 458
196	<code>\PdfRender@CurrentColor</code>
<code>\ltx@ifUndefined</code>
... 167 , 174 , 187 , 260 , 263 , 271 ,	602 , 609 , 627 , 638
277 , 279 , 296 , 398 , 406 , 452 , 480	<code>\PdfRender@CurrentLineWidth</code> 488 , 499
<code>\ltx@ifundefined</code>	<code>\PdfRender@define@key</code>
203 , 206 , 216
<code>\ltx@pkgextension</code>	388 , 395 , 406 , 537 , 558
153	<code>\PdfRender@dimexpr</code>
<code>\ltx@secondoftwo</code>	481 , 483 , 488
280 , 297	<code>\PdfRender@ErrorInvalidValue</code> ...
<code>\ltx@space</code>
176 , 463 , 464 ,	423 , 436 , 445 , 455
603 , 610 , 615 , 682 , 685 , 702 , 705	<code>\PdfRender@FillColor</code>
M
<code>\makeatletter</code>	290 , 541 , 547 , 599 ,
916 , 953	603 , 610 , 615 , 622 , 678 , 683 , 693
<code>\makeatother</code>	<code>\PdfRender@FilterOp</code> ...
920	693 , 713 , 737
<code>\makefootline</code>	<code>\PdfRender@FindOp</code>
981 , 982	...
<code>\MessageBreak</code>	687 , 688 , 689 , 690 , 691 , 692 ,
150 , 151 ,	707 , 708 , 709 , 710 , 711 , 712 , 720
152 , 153 , 177 , 425 , 549 , 570 , 582	<code>\PdfRender@GetFillColor</code> 546 , 621 , 677
N	<code>\PdfRender@GetStrokeColor</code>
<code>\NeedsTeXFormat</code>
852 , 887 , 908	567 , 632 , 697
<code>\newcommand</code> ...	<code>\PdfRender@let</code>
191 , 192 , 284 , 301 , 917	489 , 491 , 493
<code>\newlength</code>	<code>\PdfRender@MatchPattern</code> 662 , 683 , 703
872	<code>\PdfRender@Matchtrue</code>
<code>\newpage</code>	181
859 , 869 , 882	

\PdfRender@NeedsCurrentColor . . .	\ProvidesPackage 19, 67
. 598, 605, 612, 620, 631	
\PdfRender@NewClass	R
. 315, 317, 477, 516, 520, 523	\RangeCatcodeCheck
\PdfRender@NewClassValues 805, 833, 834, 835, 836, 837,
. 313, 465, 506, 511, 529	838, 839, 840, 841, 842, 843, 844
\PdfRender@newif 128, 141, 142, 312, 318	\RangeCatcodeInvalid
\PdfRender@NormalColorHook 797, 825, 826, 827, 828
. 210, 217, 220, 227, 372	\repeat 772, 784, 795, 803, 818
\PdfRender@OP 686, 706, 726, 739, 741	\RequirePackage 161
\PdfRender@OpName 464, 531	\RestoreCatcodes 786, 789, 790, 845
\PdfRender@OpValue 463,	
467, 477, 506, 511, 516, 520, 523	S
\PdfRender@PatchColor	\section 895
. 255, 259, 261, 265, 266, 268, 273	\set@color 197, 222, 241, 617, 628, 639
\PdfRender@PatchColorSetGroup . .	\setbox 213
. 237, 257	\setlength 873
\PdfRender@PatchNormalColor	\space 811, 812, 820
. 215, 256, 289	\strip@prefix 665, 668
\PdfRender@Pattern 663, 681, 701	\strip@pt 500
\PdfRender@PatternFillColor 644, 681	
\PdfRender@PatternStrokeColor . .	T
. 653, 701	\tableofcontents 894
\PdfRender@PostProcessLineWidth . 485	\Test 824, 847, 954, 962, 965
\PdfRender@relax 489, 492	\texorpdfstring 277
\PdfRender@RequirePackage	\textpdfrender 2, 189, 192, 296, 861,
. 143, 163, 164, 165, 275	875, 896, 926, 934, 942, 957, 988
\PdfRender@Reset 418	\the 77, 78, 79,
\PdfRender@Set 370, 377, 413	80, 81, 82, 83, 84, 97, 791, 811, 812
\PdfRender@SetColor 293, 597	\TMP@EnsureCode 94, 101, 102,
\PdfRender@SetValidate 402, 431	103, 104, 105, 106, 107, 108,
\PdfRender@SetValidateValues 391, 451	109, 110, 111, 112, 113, 114,
\PdfRender@Stacktrue 172	115, 116, 117, 118, 119, 120, 121
\PdfRender@String	
. 663, 667, 670, 682, 702	U
\PdfRender@StrokeColor	\usepackage 855,
. 291, 562, 568, 600, 603,	856, 890, 891, 892, 911, 912, 913
608, 610, 615, 633, 698, 703, 713	
\PdfRender@temp	V
. 220, 224, 239, 243, 721, 730, 734	\vbox 201
\PdfRender@TestBox	\vfill 991, 994
. 212, 221, 240, 544, 565	\vsize 974, 976
\PdfRender@texorpdfstring	
. 276, 288, 305	W
\PdfRender@TryColor 545, 566, 591	\write 23, 52
\PdfRender@Valuesfalse 393	
\PdfRender@Valustrue 314	X
\penalty 991, 995	\x 14, 15, 18, 22,
\ProvidesFile 853, 888, 909	26, 28, 51, 56, 66, 75, 87, 498, 503