The pdfrender package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2010/01/26 v1.0

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 **1**
1.1 Macros . . . 2
1.2 Parameters . . . 2
1.2.1 Details . . . 3
1.3 Color stack . . . 3

2 Implementation **3**
2.1 Look for pdfTeX, its mode and features . . . 5
2.2 Macros for pdfTeX in PDF mode . . . 6
2.3 Declare and setup parameters . . . 9
2.4 Fill and stroke color support . . . 10

3 Test **13**
3.1 Catcode checks for loading . . . 13
3.2 Simple test file . . . 14

4 Installation **15**
4.1 Download . . . 15
4.2 Bundle installation . . . 15
4.3 Package installation . . . 15
4.4 Refresh file name databases . . . 16
4.5 Some details for the interested . . . 16

5 Acknowledgement **16**

6 References **17**

7 History **17**
[2010/01/26 v1.0] . . . 17

8 Index **17**

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 LaTeX's color interface.

At different places LaTeX 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 LaTeX3 should generalize LaTeX 2$_\varepsilon$'s color interface.

1.1 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).

1.2 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	*positive number, unit is bp*	*TEX dimen*
LineCapStyle	0	`Butt`
	1	`Round`
	2	`ProjectingSquare`
LineJoinStyle	0	`Miter`
	1	`Round`
	2	`Bevel`
MiterLimit	*positive number*	
Flatness	*number between 0 and 100*	
LineDashPattern	*numbers in square brackets, followed by number, units are bp*	

Parameter	Value	Alias
RenderingIntent	`AbsoluteColorimetric`	
	`RelativeColorimetric`	
	`Saturation`	
	`Perceptual`	
FillColor		*color specification*
StrokeColor		*color specification*

1.2.1 Details

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

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

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

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

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

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

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

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

1.3 Color stack

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

2 Implementation

```
1 ⟨*package⟩
```

Reload check, especially if the package is not used with LATEX.

```
2 \begingroup
3   \catcode44 12 % ,
4   \catcode45 12 % -
5   \catcode46 12 % .
```

```
  \catcode58 12 % :
  \catcode64 11 % @
  \catcode123 1 % {
  \catcode125 2 % }
  \expandafter\let\expandafter\x\csname ver@pdfrender.sty\endcsname
  \ifx\x\relax % plain-TeX, first loading
  \else
    \def\empty{}%
    \ifx\x\empty % LaTeX, first loading,
      % variable is initialized, but \ProvidesPackage not yet seen
    \else
      \catcode35 6 % #
      \expandafter\ifx\csname PackageInfo\endcsname\relax
        \def\x#1#2{%
          \immediate\write-1{Package #1 Info: #2.}%
        }%
      \else
        \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
      \fi
      \x{pdfrender}{The package is already loaded}%
      \aftergroup\endinput
    \fi
  \fi
\endgroup
```

Package identification:

```
\begingroup
  \catcode35 6 % #
  \catcode40 12 % (
  \catcode41 12 % )
  \catcode44 12 % ,
  \catcode45 12 % -
  \catcode46 12 % .
  \catcode47 12 % /
  \catcode58 12 % :
  \catcode64 11 % @
  \catcode91 12 % [
  \catcode93 12 % ]
  \catcode123 1 % {
  \catcode125 2 % }
  \expandafter\ifx\csname ProvidesPackage\endcsname\relax
    \def\x#1#2#3[#4]{\endgroup
      \immediate\write-1{Package: #3 #4}%
      \xdef#1{#4}%
    }%
  \else
    \def\x#1#2[#3]{\endgroup
      #2[{#3}]%
      \ifx#1\@undefined
        \xdef#1{#3}%
      \fi
      \ifx#1\relax
        \xdef#1{#3}%
      \fi
    }%
  \fi
\expandafter\x\csname ver@pdfrender.sty\endcsname
\ProvidesPackage{pdfrender}%
  [2010/01/26 v1.0 Access to some PDF graphics parameters (HO)]
\begingroup
  \catcode123 1 % {
  \catcode125 2 % }
  \def\x{\endgroup
```

```
    \expandafter\edef\csname PdfRender@AtEnd\endcsname{%
      \catcode35 \the\catcode35\relax
      \catcode64 \the\catcode64\relax
      \catcode123 \the\catcode123\relax
      \catcode125 \the\catcode125\relax
    }%
  }%
\x
\catcode35 6 % #
\catcode64 11 % @
\catcode123 1 % {
\catcode125 2 % }
\def\TMP@EnsureCode#1#2{%
  \edef\PdfRender@AtEnd{%
    \PdfRender@AtEnd
    \catcode#1 \the\catcode#1\relax
  }%
  \catcode#1 #2\relax
}
\TMP@EnsureCode{10}{12}% ^^J
\TMP@EnsureCode{36}{3}% $
\TMP@EnsureCode{39}{12}% '
\TMP@EnsureCode{40}{12}% (
\TMP@EnsureCode{41}{12}% )
\TMP@EnsureCode{42}{12}% *
\TMP@EnsureCode{43}{12}% +
\TMP@EnsureCode{44}{12}% ,
\TMP@EnsureCode{45}{12}% -
\TMP@EnsureCode{46}{12}% .
\TMP@EnsureCode{47}{12}% /
\TMP@EnsureCode{58}{12}% :
\TMP@EnsureCode{59}{12}% ;
\TMP@EnsureCode{60}{12}% <
\TMP@EnsureCode{61}{12}% =
\TMP@EnsureCode{62}{12}% >
\TMP@EnsureCode{63}{12}% ?
\TMP@EnsureCode{91}{12}% [
\TMP@EnsureCode{93}{12}% ]
\TMP@EnsureCode{94}{7}% ^ (superscript)
\TMP@EnsureCode{96}{12}% `
\TMP@EnsureCode{124}{12}% |
\g@addto@macro\PdfRender@AtEnd{\endinput}
```

2.1 Look for pdfTEX, its mode and features

\ifPdfRender@Stack

```
\newif\ifPdfRender@Stack
```

\ifPdfRender@Match

```
\newif\ifPdfRender@Match

\RequirePackage{ifpdf}[2009/04/10]
\RequirePackage{infwarerr}[2007/09/09]
\RequirePackage{ltxcmds}[2009/12/12]

\ifpdf
  \ltx@IfUndefined{pdfcolorstackinit}{%
    \@PackageWarning{pdfrender}{%
      Missing \string\pdfcolorstackinit
    }%
  }{%
    \PdfRender@Stacktrue
  }%
```

```
  \ltx@IfUndefined{pdfmatch}{%
    \@PackageInfoNoLine{pdfrender}{%
      \string\pdfmatch\ltx@space not found. %
      Therefore the values\MessageBreak
      of some parameters are not validated%
    }%
  }{%
    \PdfRender@Matchtrue
  }%
\else
  \@PackageWarning{pdfrender}{%
    Missing pdfTeX in PDF mode%
  }%
```

`\pdfrender`

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

`\textpdfrender`

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

  \expandafter\PdfRender@AtEnd
\fi
```

2.2 Macros for pdfTEX in PDF mode

```
\RequirePackage{color}
\RequirePackage{keyval}
\RequirePackage{kvsetkeys}[2009/07/19]
```

`\pdfrender`

```
\newcommand*{\pdfrender}[1]{%
  \global\let\PdfRender@FillColor\ltx@empty
  \global\let\PdfRender@StrokeColor\ltx@empty
  \setkeys{PDFRENDER}{#1}%
  \PdfRender@SetColor
}
```

`\textpdfrender`

```
\newcommand{\textpdfrender}[2]{%
  \begingroup
    \pdfrender{#1}%
    #2%
  \endgroup
}
```

`\ifPdfRender@Values`

```
\newif\ifPdfRender@Values
```

`\PdfRender@NewClassValues`

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

`\PdfRender@NewClass`

```
\def\PdfRender@NewClass#1#2#3#4#5{%
  \expandafter\newif\csname ifPdfRender@Active#1\endcsname
  \expandafter\def\csname PdfRender@Default#1\endcsname{#2}%
  \expandafter\let\csname PdfRender@Current#1\expandafter\endcsname
      \csname PdfRender@Default#1\endcsname
  \ifPdfRender@Stack
```

```
    \expandafter\edef\csname PdfRender@Init#1\endcsname{%
      \global\chardef
      \expandafter\noexpand\csname PdfRender@Stack#1\endcsname=%
          \noexpand\pdfcolorstackinit page direct{%
            \noexpand#3%
            \expandafter\noexpand\csname PdfRender@Default#1\endcsname
          }\relax
      \noexpand\@PackageInfo{pdfrender}{%
        New color stack `#1' = \noexpand\number
        \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
      }%
      \gdef\expandafter\noexpand\csname PdfRender@Init#1\endcsname{}%
    }%
    \expandafter\edef\csname PdfRender@Set#1\endcsname{%
      \expandafter\noexpand\csname PdfRender@Init#1\endcsname
      \noexpand\pdfcolorstack
      \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
      push{%
        #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
      }%
      \noexpand\aftergroup
      \expandafter\noexpand\csname PdfRender@Reset#1\endcsname
    }%
    \expandafter\edef\csname PdfRender@Reset#1\endcsname{%
      \expandafter\noexpand\csname PdfRender@Init#1\endcsname
      \noexpand\pdfcolorstack
      \expandafter\noexpand\csname PdfRender@Stack#1\endcsname
      pop\relax
    }%
  \else
    \expandafter\edef\csname PdfRender@Set#1\endcsname{%
      \noexpand\pdfliteral direct{%
        #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
      }%
      \noexpand\aftergroup
      \expandafter\noexpand\csname PdfRender@Reset#1\endcsname
    }%
    \expandafter\edef\csname PdfRender@Reset#1\endcsname{%
      \noexpand\pdfliteral direct{%
        #3{\expandafter\noexpand\csname PdfRender@Current#1\endcsname}%
      }%
    }%
  \fi
  \expandafter\edef\csname PdfRender@Normal#1\endcsname{%
    \let
    \expandafter\noexpand\csname PdfRender@Current#1\endcsname
    \expandafter\noexpand\csname PdfRender@Default#1\endcsname
    \noexpand\PdfRender@Set{#1}%
  }%
  \expandafter\g@addto@macro\expandafter\normalcolor
  \expandafter{%
    \csname PdfRender@Normal#1\endcsname
  }%
  \g@addto@macro\color@setgroup{%
    \PdfRender@Set{#1}%
  }%
  \ifPdfRender@Values
    \kv@parse@normalized{#4}{%
      \expandafter\let\csname PdfRender@#1@\kv@key\endcsname\kv@key
      \ifx\kv@value\relax
      \else
        \expandafter\let\csname PdfRender@#1@\kv@value\endcsname\kv@key
```

```
      \fi
      \ltx@gobbletwo
    }%
    \define@key{PDFRENDER}{#1}[1]{%
      \global\csname PdfRender@Active#1true\endcsname
      \def\PdfRender@Current{##1}%
      \PdfRender@SetValidateValues{#1}%
    }%
    \PdfRender@Valuesfalse
  \else
    \define@key{PDFRENDER}{#1}[1]{%
      \global\csname PdfRender@Active#1true\endcsname
      \expandafter\def\csname PdfRender@Current#1\endcsname{##1}%
      \ltx@IfUndefined{PdfRender@PostProcess#1}{%
      }{%
        \csname PdfRender@PostProcess#1\endcsname
      }%
      \PdfRender@SetValidate{#1}{#4}{#5}%
    }%
  \fi
}%
```

\PdfRender@Set

```
\def\PdfRender@Set#1{%
  \csname ifPdfRender@Active#1\endcsname
    \csname PdfRender@Set#1\expandafter\endcsname
  \fi
}
```

\PdfRender@Reset

```
\def\PdfRender@Reset#1{%
  \csname ifPdfRender@Active#1\endcsname
    \csname PdfRender@Reset#1\expandafter\endcsname
  \fi
}
```

\PdfRender@ErrorInvalidValue

```
\def\PdfRender@ErrorInvalidValue#1{%
  \PackageError{pdfrender}{%
    Ignoring parameter setting for ‘#1’\MessageBreak
    because of invalid value %
    ‘\csname PdfRender@Current#1\endcsname’%
  }\@ehc
  \expandafter\let\csname PdfRender@Current#1\endcsname\ltx@empty
}%
```

\PdfRender@SetValidate

```
\ifPdfRender@Match
  \def\PdfRender@SetValidate#1#2#3{%
    \ifnum\pdfmatch{^(#2)$}{\csname PdfRender@Current#1\endcsname}=1 %
      \csname PdfRender@Set#1\expandafter\endcsname
    \else
      \PdfRender@ErrorInvalidValue{#1}%
    \fi
  }%
\else
  \def\PdfRender@SetValidate#1#2#3{%
    \expandafter\let\expandafter\PdfRender@Current
    \csname PdfRender@Current#1\endcsname
    #3%
    \ifx\PdfRender@Current\@empty
      \PdfRender@ErrorInvalidValue{#1}%
```

```
    \else
      \csname PdfRender@Set#1\expandafter\endcsname
    \fi
  }%
\fi
```

\PdfRender@SetValidateValues

```
\def\PdfRender@SetValidateValues#1{%
  \ltx@IfUndefined{PdfRender@#1@\PdfRender@Current}{%
    \expandafter\let\csname PdfRender@Current#1\endcsname
                    \PdfRender@Current
    \PdfRender@ErrorInvalidValue{#1}%
  }{%
    \expandafter\edef\csname PdfRender@Current#1\endcsname{%
      \csname PdfRender@#1@\PdfRender@Current\endcsname
    }%
    \csname PdfRender@Set#1\endcsname
  }%
}
```

\PdfRender@OpValue

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

\PdfRender@OpName

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

2.3 Declare and setup parameters

```
\PdfRender@NewClassValues{TextRenderingMode}%
                         {0}%
                         {\PdfRender@OpValue{Tr}}{%
  0=Fill,%
  1=Stroke,%
  2=FillStroke,%
  3=Invisible,%
  4=FillClip,%
  5=StrokeClip,%
  6=FillStrokeClip,%
  7=Clip,%
}%
\PdfRender@NewClass{LineWidth}{1}{\PdfRender@OpValue{w}}{%
  [0-9]+\string\.?[0-9]*|\string\.[0-9]+%
}{}%
\ltx@IfUndefined{dimexpr}{%
  \def\PdfRender@dimexpr{}%
}{%
  \let\PdfRender@dimexpr\dimexpr
}
\def\PdfRender@PostProcessLineWidth{%
  \begingroup
  \afterassignment\PdfRender@@PostProcessLineWidth
  \dimen@=\PdfRender@dimexpr\PdfRender@CurrentLineWidth bp %
  \PdfRender@let\PdfRender@relax\PdfRender@relax
}
\let\PdfRender@let\let
\let\PdfRender@relax\relax
\def\PdfRender@@PostProcessLineWidth#1\PdfRender@let{%
  \ifx\\#1\\%
    \endgroup
  \else
    \dimen@=.996264\dimen@ % 72/72.27
    \edef\x{\endgroup
```

```
      \def\noexpand\PdfRender@CurrentLineWidth{%
        \strip@pt\dimen@
      }%
    }%
    \expandafter\x
  \fi
}
\PdfRender@NewClassValues{LineCapStyle}{0}{\PdfRender@OpValue{J}}{%
  0=Butt,%
  1=Round,%
  2=ProjectingSquare,%
}%
\PdfRender@NewClassValues{LineJoinStyle}{0}{\PdfRender@OpValue{j}}{%
  0=Miter,%
  1=Round,%
  2=Bevel,%
}%
\PdfRender@NewClass{MiterLimit}{10}{\PdfRender@OpValue{M}}{%
  [0-9]*[1-9][0-9]*\string\.?[0-9]*|%
  [0-9]*\string\.?[0-9]*[1-9][0-9]*%
}{}%
\PdfRender@NewClass{Flatness}{0}{\PdfRender@OpValue{i}}{%
  100(\string\.0*)?|[0-9][0-9](\string\.[0-9]*)?|\string\.[0-9]+%
}{}%
\PdfRender@NewClass{LineDashPattern}{[]0}{\PdfRender@OpValue{d}}{%
  \string\[%
  ( ?([0-9]+\string\.?[0-9]*|\string\.[0-9]+) ?)*%
  \string\] ?%
  ([0-9]+\string\.?[0-9]*|\string\.[0-9]+)%
}{}%
\PdfRender@NewClassValues{RenderingIntent}%
                       {RelativeColorimetric}%
                       {\PdfRender@OpName{ri}}{%
  AbsoluteColorimetric,%
  RelativeColorimetric,%
  Saturation,%
  Perceptual,%
}%
```

2.4 Fill and stroke color support

```
\define@key{PDFRENDER}{FillColor}{%
  \begingroup
    \def\PdfRender@Color{#1}%
    \ifx\PdfRender@Color\ltx@empty
      \global\let\PdfRender@FillColor\ltx@empty
    \else
      \sbox0{%
        \expandafter\PdfRender@TryColor\PdfRender@Color\ltx@empty
        \PdfRender@GetFillColor
        \ifx\PdfRender@FillColor\ltx@empty
          \@PackageWarning{pdfrender}{%
            Cannot extract fill color\MessageBreak
            from value ‘#1’%
          }%
        \fi
      }%
    \fi
  \endgroup
}
\define@key{PDFRENDER}{StrokeColor}{%
  \begingroup
    \def\PdfRender@Color{#1}%
```

```
    \ifx\PdfRender@Color\ltx@empty
      \global\let\PdfRender@StrokeColor\ltx@empty
    \else
      \sbox0{%
        \expandafter\PdfRender@TryColor\PdfRender@Color\ltx@empty
        \PdfRender@GetStrokeColor
        \ifx\PdfRender@StrokeColor\ltx@empty
          \@PackageWarning{pdfrender}{%
            Cannot extract stroke color\MessageBreak
            from value `#1'%
          }%
        \fi
      }%
    \fi
  \endgroup
}
```

\PdfRender@TryColor

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

\PdfRender@@TryColor

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

\PdfRender@SetColor

```
\def\PdfRender@SetColor{%
  \chardef\PdfRender@NeedsCurrentColor=0 %
  \ifx\PdfRender@FillColor\ltx@empty
    \ifx\PdfRender@StrokeColor\ltx@empty
    \else
      \edef\PdfRender@CurrentColor{%
        \noexpand\PdfRender@FillColor\ltx@space\PdfRender@StrokeColor
      }%
      \chardef\PdfRender@NeedsCurrentColor=1 %
    \fi
  \else
    \ifx\PdfRender@StrokeColor\ltx@empty
      \edef\PdfRender@CurrentColor{%
        \PdfRender@FillColor\ltx@space\noexpand\PdfRender@StrokeColor
      }%
      \chardef\PdfRender@NeedsCurrentColor=2 %
    \else
      \edef\current@color{%
        \PdfRender@FillColor\ltx@space\PdfRender@StrokeColor
      }%
      \set@color
    \fi
  \fi
  \ifnum\PdfRender@NeedsCurrentColor=1 %
    \PdfRender@GetFillColor
    \ifx\PdfRender@FillColor\ltx@empty
      \@PackageWarning{pdfrender}{%
        Cannot extract current fill color%
      }%
    \else
      \edef\current@color{\PdfRender@CurrentColor}%
      \set@color
    \fi
  \else
```

```
    \ifnum\PdfRender@NeedsCurrentColor=2 %
      \PdfRender@GetStrokeColor
      \ifx\PdfRender@StrokeColor\ltx@empty
        \@PackageWarning{pdfrender}{%
          Cannot extract current stroke color%
        }%
      \else
        \edef\current@color{\PdfRender@CurrentColor}%
        \set@color
      \fi
    \fi
  \fi
}
```

\PdfRender@GetFillColor

```
\def\PdfRender@GetFillColor{%
  \global\let\PdfRender@FillColor\ltx@empty
  \begingroup
    \ifPdfRender@Match
      \edef\current@color{\ltx@space\current@color\ltx@space}%
      \ifnum\pdfmatch{ %
        (%
          [0-9\string\.]+ g|%
          [0-9\string\.]+ [0-9\string\.]+ [0-9\string\.]+ rg|%
          [0-9\string\.]+ [0-9\string\.]+ %
          [0-9\string\.]+ [0-9\string\.]+ k%
        )%
      }{\current@color}=1 %
        \xdef\PdfRender@FillColor{%
          \expandafter\strip@prefix\pdflastmatch 1%
        }%
      \fi
    \else
% Using the method of Rolf Niepraschk.
      \edef\x{%
        \lowercase{%
          \xdef\noexpand\PdfRender@FillColor{%
            \current@color
          }%
        }%
      }%
      \x
    \fi
  \endgroup
}
```

\PdfRender@GetStrokeColor

```
\def\PdfRender@GetStrokeColor{%
  \global\let\PdfRender@StrokeColor\ltx@empty
  \begingroup
    \ifPdfRender@Match
      \edef\current@color{\ltx@space\current@color\ltx@space}%
      \ifnum\pdfmatch{ %
        (%
          [0-9\string\.]+ G|%
          [0-9\string\.]+ [0-9\string\.]+ [0-9\string\.]+ RG|%
          [0-9\string\.]+ [0-9\string\.]+ %
          [0-9\string\.]+ [0-9\string\.]+ K%
        )%
      }{\current@color}=1 %
        \xdef\PdfRender@StrokeColor{%
          \expandafter\strip@prefix\pdflastmatch 1%
```

```
        }%
      \fi
    \else
% Using the method of Rolf Niepraschk.
      \edef\x{%
        \uppercase{%
          \xdef\noexpand\PdfRender@StrokeColor{%
            \current@color
          }%
        }%
      }%
      \x
    \fi
  \endgroup
}

\PdfRender@AtEnd
```
⟨/package⟩

3 Test

3.1 Catcode checks for loading

⟨∗test1⟩

```
\catcode`\{=1 %
\catcode`\}=2 %
\catcode`\#=6 %
\catcode`\@=11 %
\expandafter\ifx\csname count@\endcsname\relax
  \countdef\count@=255 %
\fi
\expandafter\ifx\csname @gobble\endcsname\relax
  \long\def\@gobble#1{}%
\fi
\expandafter\ifx\csname @firstofone\endcsname\relax
  \long\def\@firstofone#1{#1}%
\fi
\expandafter\ifx\csname loop\endcsname\relax
  \expandafter\@firstofone
\else
  \expandafter\@gobble
\fi
{%
  \def\loop#1\repeat{%
    \def\body{#1}%
    \iterate
  }%
  \def\iterate{%
    \body
      \let\next\iterate
    \else
      \let\next\relax
    \fi
    \next
  }%
  \let\repeat=\fi
}%
\def\RestoreCatcodes{}
\count@=0 %
\loop
  \edef\RestoreCatcodes{%
    \RestoreCatcodes
```

```
    \catcode\the\count@=\the\catcode\count@\relax
  }%
\ifnum\count@<255 %
  \advance\count@ 1 %
\repeat

\def\RangeCatcodeInvalid#1#2{%
  \count@=#1\relax
  \loop
    \catcode\count@=15 %
  \ifnum\count@<#2\relax
    \advance\count@ 1 %
  \repeat
}
\expandafter\ifx\csname LoadCommand\endcsname\relax
  \def\LoadCommand{\input pdfrender.sty\relax}%
\fi
\def\Test{%
  \RangeCatcodeInvalid{0}{47}%
  \RangeCatcodeInvalid{58}{64}%
  \RangeCatcodeInvalid{91}{96}%
  \RangeCatcodeInvalid{123}{255}%
  \catcode`\@=12 %
  \catcode`\\=0 %
  \catcode`\{=1 %
  \catcode`\}=2 %
  \catcode`\#=6 %
  \catcode`\[=12 %
  \catcode`\]=12 %
  \catcode`\%=14 %
  \catcode`\ =10 %
  \catcode13=5 %
  \LoadCommand
  \RestoreCatcodes
}
\Test
\csname @@end\endcsname
\end
```

⟨/test1⟩

3.2 Simple test file

```
⟨*test2⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesFile{pdfrender-test2.tex}[2010/01/26]
\documentclass{article}
\usepackage{pdfrender}[2010/01/26]
\begin{document}
Hello World
\newpage
Start
\textpdfrender{%
  TextRenderingMode=1,%
  LineWidth=.1,%
  LineCapStyle=2,%
  LineJoinStyle=1,%
  MiterLimit=1.2,%
  LineDashPattern=[2 2]0,%
  RenderingIntent=Saturation,%
}{Hello\newpage World}
Stop
\par
\newlength{\LineWidth}
```

```
\setlength{\LineWidth}{.5pt}
Start
\textpdfrender{%
  FillColor=yellow,%
  StrokeColor=[cmyk]{1,.5,0,0},%
  TextRenderingMode=FillStroke,%
  LineWidth=.5\LineWidth,%
  LineCapStyle=Round,%
  LineJoinStyle=Bevel,%
}{Out-\par\newpage line}
Stop
\end{document}
```
⟨/test2⟩

4 Installation

4.1 Download

Package. This package is available on CTAN[1]:

CTAN:macros/latex/contrib/oberdiek/pdfrender.dtx The source file.

CTAN:macros/latex/contrib/oberdiek/pdfrender.pdf Documentation.

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

CTAN:install/macros/latex/contrib/oberdiek.tds.zip

TDS refers to the standard "A Directory Structure for TEX Files" (CTAN:tds/tds.pdf). Directories with `texmf` in their name are usually organized this way.

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
```

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

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/latex/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
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 `mktexlsr`.

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 pdfLATEX:

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

5 Acknowledgement

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

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

Rolf Niepraschk added color support [4].

6 References

[1] Adobe Systems Incorporated. *PDF Reference – Adobe Portable Document format – Version 1.7*. 6th ed. 2006. URL: http://www.adobe.com/devnet/acrobat/pdfs/pdf_reference_1-7.pdf.

[2] Friedrich Vosberg, *Text in Buchstabenumrissen*, de.comp.text.tex, 2010-01-22. URL: http://groups.google.com/group/de.comp.text.tex/msg/f442310ac8b2d506.

[3] Gaius Pupus, *Re: Text in Buchstabenumrissen*, de.comp.text.tex, 2010-01-23. URL: http://groups.google.com/group/de.comp.text.tex/msg/95d890d77ac47eb1.

[4] Rolf Niepraschk, *Re: Text in Buchstabenumrissen*, de.comp.text.tex, 2010-01-24. URL: http://groups.google.com/group/de.comp.text.tex/msg/4eb61a5879db54db.

7 History

[2010/01/26 v1.0]

- The first version.

8 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols

\# 528, 590
\% 593
\. 313, 352, 353, 356, 360, 362, 470, 471, 472, 473, 500, 501, 502, 503
\@ 529, 586
\@PackageInfo 172
\@PackageInfoNoLine 123
\@PackageWarning 116, 132, 382, 401, 442, 453
\@ehc 263
\@empty 279
\@firstofone 537, 540
\@gobble 534, 542
\@ifnextchar 411
\@undefined 52
\[. 359, 591
\\ 329, 587
\{ 526, 588
\} 527, 589
\] 361, 592

\␣ 594

A

\advance 567, 575
\afterassignment 322
\aftergroup 26, 185, 199

B

\begin 608
\body 546, 550

C

\catcode 3, 4, 5, 6, 7, 8, 9, 17, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 64, 65, 68, 69, 70, 71, 75, 76, 77, 78, 82, 84, 526, 527, 528, 529, 564, 573, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595
\chardef 166, 417, 424, 431
\color 411, 414
\color@setgroup 218
\count@ 531, 560, 564, 566, 567, 571, 573, 574, 575
\countdef 531
\csname 10, 18, 44, 60, 67, 160, 161, 162, 163, 165, 167, 170, 174, 176, 178, 179, 181, 183, 186, 188, 189, 191, 195, 197, 200, 202, 204, 208, 210, 211, 216, 223, 226, 231, 238, 239, 242, 249, 250, 254, 255, 262, 264, 268, 269, 277, 282, 288, 292, 293, 295, 530, 533, 536, 539, 578, 600
\current@color 433, 446, 457, 467, 475, 485, 497, 505, 515

D

\define@key 230, 237, 372, 391
\dimen@ 323, 332, 335
\dimexpr 318
\documentclass 606

E

\empty 13, 14
\end 601, 635
\endcsname 10, 18, 44, 60, 67, 160, 161, 162, 163, 165, 167, 170, 174, 176, 178, 179, 181, 183, 186, 188, 189, 191, 195, 197, 200, 202, 204, 208, 210, 211, 216, 223, 226, 231, 238, 239, 242, 249, 250, 254, 255, 262, 264, 268, 269, 277, 282, 288, 292, 293, 295, 530, 533, 536, 539, 578, 600
\endinput 26, 108

G

\g@addto@macro 108, 214, 218
\gdef 176

I

\ifnum 268, 439, 450, 468, 498, 566, 574
\ifpdf 114
\ifPdfRender@Match . 110, 266, 466, 496
\ifPdfRender@Stack 109, 164
\ifPdfRender@Values 154, 221
\ifx 11, 14, 18, 44, 52, 55, 224, 279, 329, 375, 381, 394, 400, 418, 419, 427, 441, 452, 530, 533, 536, 539, 578
\immediate 20, 46
\input 579
\iterate 547, 549, 551

K

\kv@key 223, 226
\kv@parse@normalized 222
\kv@value 224, 226

L

\LineWidth 623, 624, 630
\LoadCommand 579, 596
\loop 545, 561, 572
\lowercase 483
\ltx@empty 143, 144, 264, 375, 376, 379, 381, 394, 395, 398, 400, 413, 418, 419, 427, 441, 452, 464, 494
\ltx@gobbletwo 228
\ltx@IfUndefined 115, 122, 240, 287, 315
\ltx@space 124, 298, 299, 422, 429, 434, 467, 497

M

\MessageBreak 125, 260, 383, 402

N

\NeedsTeXFormat 604
\newcommand 135, 136, 142, 148
\newif 109, 110, 154, 160
\newlength 623
\newpage 610, 620, 633
\next 551, 553, 555
\normalcolor 214
\number 173

P

\PackageError 259
\PackageInfo 23
\par 622, 633
\pdfcolorstack 180, 190
\pdfcolorstackinit 117, 168
\pdflastmatch 477, 507
\pdfliteral 196, 203
\pdfmatch 124, 268, 468, 498
\pdfrender *2*, 135, 142, 150
\PdfRender@@PostProcessLineWidth 322, 328
\PdfRender@@TryColor 411, 413
\PdfRender@AtEnd 80, 81, 108, 137, 523
\PdfRender@Color 374, 375, 379, 393, 394, 398, 414
\PdfRender@Current 232, 276, 279, 287, 289, 293
\PdfRender@CurrentColor 421, 428, 446, 457
\PdfRender@CurrentLineWidth 323, 334
\PdfRender@dimexpr 316, 318, 323
\PdfRender@ErrorInvalidValue 258, 271, 280, 290
\PdfRender@FillColor 143, 376, 381, 418, 422, 429, 434, 441, 464, 476, 484
\PdfRender@GetFillColor 380, 440, 463
\PdfRender@GetStrokeColor 399, 451, 493
\PdfRender@let 324, 326, 328
\PdfRender@Matchtrue 129
\PdfRender@NeedsCurrentColor 417, 424, 431, 439, 450
\PdfRender@NewClass 157, 159, 312, 351, 355, 358
\PdfRender@NewClassValues 155, 300, 341, 346, 364
\PdfRender@OpName 299, 366
\PdfRender@OpValue 298, 302, 312, 341, 346, 351, 355, 358
\PdfRender@PostProcessLineWidth . 320
\PdfRender@relax 324, 327
\PdfRender@Reset 253
\PdfRender@Set 212, 219, 248
\PdfRender@SetColor 146, 416
\PdfRender@SetValidate 244, 266
\PdfRender@SetValidateValues 233, 286
\PdfRender@Stacktrue 120
\PdfRender@StrokeColor 144, 395, 400, 419, 422, 427, 429, 434, 452, 494, 506, 514
\PdfRender@TryColor . . . 379, 398, 410
\PdfRender@Valuesfalse 235
\PdfRender@Valuestrue 156

\ProvidesFile 605
\ProvidesPackage 15, 61

R

\RangeCatcodeInvalid 570, 582, 583, 584, 585
\repeat 545, 557, 568, 576
\RequirePackage 111, 112, 113, 139, 140, 141
\RestoreCatcodes . . 559, 562, 563, 597

S

\sbox 378, 397
\set@color 436, 447, 458
\setkeys 145
\setlength 624
\strip@prefix 477, 507
\strip@pt 335

T

\Test 581, 599
\textpdfrender . . *2*, 136, 148, 612, 626
\the 68, 69, 70, 71, 82, 564
\TMP@EnsureCode 79, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107

U

\uppercase 513
\usepackage 607

W

\write 20, 46

X

\x . 10, 11, 14, 19, 23, 25, 45, 50, 60, 66, 74, 333, 338, 482, 489, 512, 519