

Das Paket `pst-pdf`*

Rolf Niepraschk[†] Hubert Gäßlein

2016/07/11

1 Einleitung

Das Paket `pst-pdf` vereinfacht die Verwendung von PSTricks-Grafiken und anderem PostScript-Code in PDF-Dokumenten. Ähnlich wie beim Erstellen des Literaturverzeichnisses mit `bibTeX` werden zusätzlich externe Programme aufgerufen. Sie dienen in diesem Fall dazu, eine PDF-Datei, die sämtliche Grafiken enthält, zu erzeugen. Ihr Inhalt wird im endgültigen Dokument statt des ursprünglichen PostScript-Codes eingefügt.

2 Anwendung

2.1 Paketoptionen

active Aktiviert den Extraktionsmodus (DVI-Ausgabe). Die explizite Angabe ist normalerweise unnötig (Standard im `LATEX`-Modus).

inactive Keine besonderen Aktionen; es werden nur die Pakete `pstricks` und `graphicx` geladen (Standard bei Verwendung von `VTEX`). Kann dazu benutzt werden, um das Dokument mit `LATEX` in eine DVI-Datei zu wandeln und dabei die automatische Verwendung des Extraktionsmodus' zu vermeiden.

pstricks Das Paket `pstricks` wird geladen (Standard).

nopstricks Das Paket `pstricks` wird nicht geladen. Wird später festgestellt, dass `pstricks` doch noch anderweitig geladen wurde, wird die Umgebung `pspicture` nachträglich in der Weise behandelt, als wäre die Option "pstricks" doch angegeben worden.

draft Im `pdfLATEX`-Modus werden aus der Containerdatei eingefügte Grafiken nur als Rahmen dargestellt.

final Im `pdfLATEX`-Modus werden aus der Containerdatei eingefügte Grafiken vollständig dargestellt (Standard).

tightpage Die Abmessung Grafiken in der Containerdatei entsprechen denen der zugehörigen `TEX`-Boxen (Standard).

*Dieses Dokument bezieht sich auf `pst-pdf` v1.2a vom 2016/07/11.

[†]Rolf.Niepraschk@gmx.de

notightpage Die Abmessung der zur Grafik gehörenden \TeX -Box ist manchmal nicht korrekt, da PostScript-Anweisungen auch außerhalb der Box zeichnen können. Die Option “notightpage” führt dazu, dass die Grafiken in der Containerdatei mindestens die Größe des gesamten Blattes einnehmen. Um die Grafiken im späteren pdf \LaTeX -Lauf verwenden zu können, muss die Containerdatei nachbearbeitet werden, so dass die Größe der Grafiken auf die der sichtbaren Bestandteile reduziert ist. Dazu kann z. B. das Programm `pdfcrop`¹ dienen. Die Anwendung dieses Verfahrens kann die Angabe der Option “trim” erübrigen (siehe Abschnitt ??).

displaymath Es werden zusätzlich die mathematischen Umgebungen `displaymath`, `eqnarray` und `$$` extrahiert und im pdf-Modus als Grafik eingefügt. So können zusätzliche PSTricks-Ergänzungen leicht dem Inhalt dieser Umgebungen zugefügt werden. (Frage: Wie verhalten sich die AMS \LaTeX -Umgebungen?)

<other> Alle anderen Optionen werden an das Paket `pstricks` weitergereicht.

2.2 Programmaufrufe

Die folgende Tabelle zeigt den Ablauf, der nötig ist, um ein PDF-Dokument mit PostScript-Grafiken zu erzeugen². Im Vergleich dazu ist der analoge Ablauf für Literaturverzeichnisse angegeben.

PostScript-Grafiken	Literaturverzeichnis
<code>pdflatex document.tex</code>	<code>pdflatex document.tex</code>
<i>Hilfsaufrufe</i>	
<code>latex document.tex</code>	
<code>dvips -o document-pics.ps document.dvi</code>	
<code>ps2pdf document-pics.ps</code>	<code>bibtex document.aux</code>
<code>pdflatex document.tex</code>	<code>pdflatex document.tex</code>

Bei der Erzeugung wird nur Code berücksichtigt, der sich innerhalb der Umgebungen `pspicture` oder `postscript` befindet. Ebenfalls werden PostScript-Grafiken, die als Parameter von `\includegraphics` angegeben wurden, der Containerdatei hinzugefügt. Der Name dieser Datei ist standardmäßig `\jobname-pics.pdf`. Er kann durch Undefinieren des Makros `\PDFcontainer` geändert werden.

2.3 Nutzeranweisungen

pspicture `\begin{pspicture}[\langle keys \rangle] (\langle x0,x1 \rangle) (\langle y0,y1 \rangle) ... \end{pspicture}`
Die `pspicture`-Umgebung steht zur Verfügung, wenn nicht die Option “nopstricks” angegeben wurde. Sie wird so wie in PSTricks üblich verwendet. Im pdf \LaTeX -Modus wird ihr Inhalt nur dann dargestellt, wenn vorher die Containerdatei erzeugt wurde.

postscript `\begin{postscript}[\langle keys \rangle] ... \end{postscript}`

¹CTAN: support/pdfcrop/

²Die Shell-Skripte `ps4pdf` bzw. `ps4pdf.bat` führen alle angegebenen Programmaufrufe automatisch aus.

Die `postscript`-Umgebung kann beliebigen Code mit Ausnahme von Gleitumgebungen aufnehmen. Im `pdfLATEX`-Modus wird ihr Inhalt ebenfalls der Containerdatei entnommen. Ist diese Datei nicht vorhanden, wird – anders als bei der `pspicture`-Umgebung – der später benötigte Platz möglicherweise nicht korrekt frei gehalten.

<code>\includegraphics</code>	<code>\includegraphics[⟨keys⟩]{⟨filename⟩}</code> Wie in <code>graphics/graphicx</code> definiert zu verwenden. Zusätzlich ist es nun möglich, auch im <code>pdfL^AT_EX</code> -Modus EPS-Dateien als Argument anzugeben und ihren Inhalt darzustellen. Er wird dazu ebenfalls der Containerdatei entnommen.
<code>\includegraphicsx</code>	<code>\includegraphicsx[⟨keys⟩](⟨pfxadd⟩)<⟨ovpfgd⟩>[⟨ovpbgd⟩]{⟨filename⟩}</code> Wie im Paket <code>psfragx</code> definiert zu verwenden.
<code>\savepicture</code>	<code>\savepicture{⟨name⟩}</code> Die zuletzt ausgegebene Grafik (Ergebnisse der Umgebungen <code>pspicture</code> , <code>postscript</code> und der <code>\includegraphics</code> -Anweisungen mit PostScript-Dateien) wird unter dem als Parameter übergebenen Namen gespeichert.
<code>\usepicture</code>	<code>\usepicture[⟨keys⟩]{⟨name⟩}</code> Die zuvor mit <code>\savepicture</code> gespeicherte Grafik wird ausgegeben. Der optionale Parameter entpricht dem bei der Anweisung <code>\includegraphics</code> möglichen.
<code>pst-pdf-defs</code>	<code>\begin{pst-pdf-defs} ... \end{pst-pdf-defs}</code> Sollen eigene Makros oder Umgebungen definiert werden, die das Zeichen <code>&</code> (andere?) im Ersetzungstext enthalten, so müssen diese Definitionen von der Umgebung <code>pst-pdf-defs</code> umschlossen werden.

2.4 Command options

Das Verhalten der Anweisungen `\includegraphics`, `\usepicture` und der Umgebung `postscript` kann mit den folgenden optionalen Parametern beeinflusst werden (key-value-Syntax):

frame=⟨true|false⟩ Es wird – ähnlich wie bei der Anweisung `\fbox` – ein Rahmen um die Grafik gezeichnet. Die durch Rotation geänderte Gesamtgröße wird dabei berücksichtigt. Das Zeichnen geschieht im `pdfLATEX`-Modus; vorher beim Erzeugen der Containerdatei wird dieser Parameter ignoriert. Standard: false.

innerframe=⟨true|false⟩ Wie “**frame**” jedoch wird der Rahmen nur um die Grafik selbst, nicht aber um die resultierende Box gezeichnet.

ignore=⟨true|false⟩ Bei “**true**” wird die Grafik nicht ausgegeben. Bei Angabe von `\savepicture{⟨name⟩}` kann sie später jedoch an anderer Stelle mit `\usepicture` verwendet werden. Standard: false.

showname=⟨true|false⟩ Gibt in kleiner Schrift den tatsächlich verwendeten Dateinamen unter der Grafik aus. Standard: false.

namefont=⟨font commands⟩ Beeinflusst die Schriftart, die bei “**showname=true**” benutzt wird. Standard: `\ttfamily\tiny`

Alle Parameter können auch global per `\setkeys{Gin}{⟨key=value⟩}` gesetzt werden.

3 Implementation

```
1 <*package>
```

3.1 Package options

```
2 \newcommand*\ppf@TeX@mode{-1}
3 \newcommand*\ppf@draft{false}
4 \newif\if@ppf@PST@used\@ppf@PST@usedtrue
5 \newif\if@ppf@tightpage \@ppf@tightpagetrue
6 \DeclareOption{active}{\OptionNotUsed}
7 \DeclareOption{inactive}{\def\ppf@TeX@mode{9}}
8 \DeclareOption{ignore}{\def\ppf@TeX@mode{999}}
9 \DeclareOption{pstricks}{\@ppf@PST@usedtrue}
10 \DeclareOption{nopstricks}{\@ppf@PST@usedfalse}
11 \DeclareOption{displaymath}{%
12   \PassOptionsToPackage\CurrentOption{preview}}
13 \DeclareOption{draft}{\def\ppf@draft{true}}
14 \DeclareOption{final}{\def\ppf@draft{false}}%
15   \PassOptionsToPackage\CurrentOption{graphicx}}
16 \DeclareOption{notightpage}{\@ppf@tightpagefalse}%
17 \DeclareOption{tightpage}{\@ppf@tightpagetrue}%
18 \DeclareOption*{%
19   \PassOptionsToPackage\CurrentOption{pstricks}}
20 \ProcessOptions\relax
21 \ifnum\ppf@TeX@mode=999\relax\expandafter\endinput\fi
```

3.2 Compilertests

Es wird getestet, welcher \TeX compiler in welchem Modus läuft (siehe ‘graphics.cfg’ von \textTeX /T \TeX Live). Entsprechend dem Ergebnis bekommen die Umgebungen `pspicture` und `postscript` unterschiedliche Funktionalität. Der Test wird nur ausgeführt, wenn nicht die Paketoptionen `active` oder `inactive` angegeben wurden.

```
22 \ifnum\ppf@TeX@mode=-1\relax
23   \RequirePackage{ifpdf,ifxetex,ifvtex}%
24   \ifpdf
25     ⇒ pdf $\text{\TeX}$  or Lua $\text{\TeX}$  are running in PDF mode
26     \def\ppf@TeX@mode{1}%
27     \RequirePackage{luatex85}%
28   \else
29     \ifvtex
30       ⇒ V $\text{\TeX}$ 
31       \def\ppf@TeX@mode{9}%
32     \else
33       \ifxetex
34         ⇒ Xe $\text{\TeX}$ 
35         \def\ppf@TeX@mode{9}%
36       \else
37         ⇒ DVI mode
38         \def\ppf@TeX@mode{0}%
39       \fi
```

```

36     \fi
37   \fi
38 \fi

39 \newcommand*\PDFcontainer{}
40 \edef\PDFcontainer{\jobname-pics.pdf}
41 \newcounter{pspicture}
42 \newcommand*\ppf@other@extensions[1]{}
43 \newcommand*\usepicture[2][{}]{
44 \newcommand*\savepicture[1]{}

```

pst-pdf-defs

```

45 \newenvironment*{pst-pdf-defs}{%
46   \endgroup
47 %   ??? \@currenvline
48 }{%
49   \begingroup
50   \def\@currenvir{pst-pdf-defs}%
51 }

52 \RequirePackage{graphicx}%
53 \let\ppf@Gininclude@graphics\Gininclude@graphics
54 \let\ppf@Gin@extensions\Gin@extensions
55 \let\ppf@Gin@ii\Gin@ii

56 \newif\ifppf@pdftex@graphic
57 \newif\ifGin@frame\Gin@framefalse
58 \newif\ifGin@innerframe\Gin@innerframefalse
59 \newif\ifGin@showname\Gin@shownamefalse
60 \newif\ifGin@ignore\Gin@ignorefalse

```

\ifpr@outer wird eigentlich im Paket preview definiert. Wir müssen es aber bereits hier zusätzlich tun, da sonst T_EX u. U. beim Parsen der \ifcase-Struktur “außer Tritt” kommt.

```

61 \newif\ifpr@outer

```

\ppf@is@pdfTeX@graphic Parameter #1 ist der Name einer Grafikdatei mit oder ohne Endung, Parameter #2 enthält die gültigen Dateierweiterungen im pdf-Modus, Parameter #3 enthält die gültigen Dateierweiterungen im dvi-Modus. Ist es möglich, die Grafik im pdf-Modus zu verarbeiten, werden die Anweisungen in #4 ausgeführt, sonst die in #5.

```

62 \newcommand*\ppf@is@pdfTeX@graphic[5]{%
63   \@ppf@pdftex@graphicfalse%
64   \begingroup
65   \edef\pdfTeXext{#2}%

```

Statt Einladen einer identifizierten Grafik nur Test der Grafikendung.

```

66   \def\Gin@setfile##1##2##3{%
67     \edef\@tempb{##2}%
68     \@for\@tempa:=\pdfTeXext\do{%
69       \ifx\@tempa\@tempb\global\@ppf@pdftex@graphictrue\fi}}%

```

Es müssen Dateitypen beider Moden gefunden werden, um die Fehlermeldung “File ‘#1’ not found” zu vermeiden.

```

70   \edef\Gin@extensions{#2,#3}%

```

Testaufruf. Dabei Ausgabe vollständig verhindern.

```

71 \pr@outerfalse\ppf@Ginclude@graphics{#1}%
72 \endgroup
73 \if@ppf@pdftex@graphic#4\else#5\fi
74 }

75 \ifcase\ppf@TeX@mode\relax

```

3.3 Extraction mode (dvi output)

Die Umgebung `pspicture` behält die Definition aus `pstricks.tex`. Ausschließlich der Code der Umgebungen `pspicture` und `postscript` sowie `\includegraphics` mit PS-Dateien bewirken Einträge in die DVI-Datei. Der restliche Code des Dokuments wird bei der Ausgabe der DVI-Datei ignoriert. Nach Wandlung der DVI-Datei über PostScript (“dvips”) nach PDF (Datei `\PDFcontainer`) nimmt jede Grafik genau eine Seite der pdf-Datei ein. Der \TeX -Compiler mit DVI-Ausgabe sowie die Paketoption “active” erzwingen diesen Modus.

```

76 \PackageInfo{pst-pdf}{%
77   MODE: \ppf@TeX@mode\space (dvi -- extraction mode)}
78 \nofiles
79 \let\makeindex\@empty \let\makeglossary\@empty
80 \AtBeginDocument{\overfullrule=\z@}%
81 \if@ppf@PST@used\RequirePackage{pstricks}\fi
82 \RequirePackage[active,dvips,tightpage]{preview}[2005/01/29]%
83 \newcommand*\ppf@PreviewBbAdjust{}
84 \newcommand*\ppf@RestoreBbAdjust{%
85   \let\PreviewBbAdjust\ppf@PreviewBbAdjust}%

```

Es werden auch die im pdf \LaTeX -Modus erlaubten Endungen von Grafikdateien benötigt.

```

86 \begingroup
87 \let\AtBeginDocument\@gobble \let\PackageWarningNoLine\@gobbletwo
88 \chardef\pdftexversion=121 %
89 \newcount\pdfoutput
90 \pdfoutput=1 %
91 \input{pdftex.def}%
92 \edef\x{\endgroup\def\noexpand\ppf@other@extensions{\Gin@extensions}
93 }%
94 \x

```

Für die im PDF-Modus möglichen Grafikformate dürfen keine speziellen Regeln definiert sein (z. B. wegen ‘dvips’-Erweiterungen). Für sie wird die universelle EPS-Regel verwendet, damit sie zumindest gefunden werden.

```

95 \AtBeginDocument{%
96   \ifpackageloaded{keyval}{%
97     \def\KV@errx#1{\PackageInfo{keyval}{#1}}%
98   }{}%
99   \@ifpackageloaded{xkeyval}{%
100     \def\XKV@err#1{\PackageInfo{xkeyval}{#1}}%
101   }{}%

```

In diesem Modus sollten undefinierte keys keinen Fehler bewirken.

```

102 \@for\@tempa:=\ppf@other@extensions\do{%
103   \expandafter\let\csname Gin@rule@\@tempa\endcsname\relax}%
104 \DeclareGraphicsRule{*}{eps}{*}{}%

```

In diesem Modus keine Funktion.

```

105 \define@key{Gin}{innerframe}[true]{}%
106 \define@key{Gin}{frame}[true]{}%
107 \define@key{Gin}{ignore}[true]{}%
108 \define@key{Gin}{showname}[true]{}%
109 \define@key{Gin}{namefont}{}%
110 \@ifundefined{GPT@page}{\define@key{Gin}{page}{}{}}{}

111 \if@ppf@tightpage\else
112   \def\PreviewBbAdjust{%
113     -600pt -600pt 600pt 600pt}%
114   \AtEndDocument{%
115     \PackageWarningNoLine{pst-pdf}{Picture container needs cropping.}}%
116 \fi

```

postscript Die Umgebung postscript wertet die trim-Option in derselben Weise wie \includegraphics aus (Angaben ohne Maßeinheit werden als bp interpretiert).

```

117 \newenvironment{postscript}[1][]{%
118   {%
119     \global\let\ppf@PreviewBbAdjust\PreviewBbAdjust
120     \if@ppf@tightpage
121       \begingroup
122         \setkeys{Gin}{#1}%
123         \xdef\PreviewBbAdjust{%
124           -\Gin@vllx bp -\Gin@villy bp \Gin@vurx bp \Gin@vury bp}%
125       \endgroup
126     \fi
127     \ignorespaces
128   }%
129   {\aftergroup\ppf@RestoreBbAdjust}%

130 \PreviewEnvironment{postscript}%
131 \AtBeginDocument{%
132   \@ifundefined{PSTricksLoaded}{}%
133   {%

```

pspicture Originaldefinition preview bekannt machen.

```

134 \PreviewEnvironment{pspicture}%

```

psmatrix Originaldefinition preview bekannt machen.

```

135 \@ifundefined{psmatrix}{}%
136 {%
137   \PreviewEnvironment{psmatrix}%
138   \newcommand*\ppf@set@mode{}%
139   \newcommand*\ppf@test@mmode{%
140     \ifmmode
141       \ifinner
142         \let\ppf@set@mode=$%
143       \else
144         \def\ppf@set@mode{$$}%
145       \fi
146     \else
147       \let\ppf@set@mode=\@empty
148     \fi

```

```

149     }%
150     \let\ppf@psmatrix=\psmatrix
151     \expandafter\let\expandafter\ppf@pr@psmatrix%
152         \expandafter=\csname pr@\string\psmatrix\endcsname
153     \let\ppf@endpsmatrix=\endpsmatrix
154     \def\psmatrix{\ppf@test@mode\ppf@psmatrix}
155     \expandafter\def\csname pr@\string\psmatrix\endcsname{%
156         \ppf@set@mode\ppf@pr@psmatrix}%
157     \def\endpsmatrix{\ppf@endpsmatrix\ppf@set@mode}%
158     }%

```

Internes Makro `\pst@object` bekanntmachen, um manchen PSTricks-Code außerhalb von `pspicture`-Umgebungen ebenfalls verwenden zu können. Derzeit sind Aufrufe der folgenden Art möglich:

```

\pst@object {<m>}<*>[<o>]{<o>}{<o>}<(>)>(<o>)>(<o>)>
(m = notwendig, * = optional, o = optional)

```

Mehr als drei optionale Argumente am Ende des Aufrufs, wie beispielsweise bei `\psline` denkbar, sind noch nicht möglich.

```

159     \PreviewMacro[{}*[]%
160     ?\bgroup{#{#1}{#{1}}}{}%
161     ?\bgroup{#{#1}{#{1}}}{}%
162     ?({#{(1)}{({#1})}}){}%
163     ?({#{(1)}{({#1})}}){}%
164     ?({#{(1)}{({#1})}}){}%
165     ]{\pst@object}}

```

Mehrfaches testweises Setzen von Tabelleninhalten durch “`tabularx`” verhindern.

```

166     \@ifundefined{tabularx}{}{%
167     \newcolumntype{X}{c}%
168     \expandafter\let\expandafter\tabularx\csname tabular*\endcsname
169     \expandafter\let\expandafter\endtabularx\csname endtabular*\endcsname
170     }%

```

Unterstützung von `\includegraphicx` aus dem Paket `psfragx`.

```

171     \@ifundefined{pfx@includegraphicx}{}{%
172     \PreviewMacro[{}{}]{\pfx@includegraphicx}}%
173     }%

```

`\Gscale@@box` Skalieren verhindern.

```

174     \def\Gscale@@box#1#2#3{%
175     \toks@{\mbox}%
176     }

```

`\Ginclude@graphics` Alle Grafiken mit bekanntem Format (z. B. EPS-Dateien) werden normal verarbeitet, was in diesem Modus bedeutet, dass sie der Preview-Funktionalität unterliegen. Andere Grafiken (z. B. PDF-Dateien) werden ignoriert.

```

177     \def\Ginclude@graphics#1{%
178     \ifpr@outer

```

Im allgemeinen Fall sollen pdf_{TeX}-Grafiken bevorzugt werden (Einfügen erst im pdf_{TeX}-Modus). Ist nur eine DVIPS-Graphik vorhanden, dann wirkt wieder die Originaldefinition und Registrierung beim `preview`-Paket muss erfolgen.

```

179     \ppf@is@pdfTeX@graphic{#1}{\ppf@other@extensions}{\Gin@extensions}%

```


Dummy-Box, um Division durch Null bei Skalierung/Rotation zu vermeiden. Wird ansonsten ignoriert.

```

180     {\rule{10pt}{10pt}}%
181     {\ppf@Ginclude@graphics{#1}}%
182     \else
    Innerhalb von PS-Umgebungen (pspicture usw.) muss sich \includegraphics
    wie die Originaldefinition verhalten (nur die DVIPS-Graphik-Typen sind gültig).
183     \ppf@Ginclude@graphics{#1}%
184     \fi
185 }%

186 \PreviewMacro[{}]{\ppf@Ginclude@graphics}%
187 \let\pdfliteral\@gobble%
188 \or

```

3.4 pdfL^AT_EX mode (pdf output)

Ist die Datei \PDFcontainer (default: \jobname-pics.pdf) vorhanden, so wird der Inhalt der Umgebungen pspicture und postscript ignoriert. Stattdessen wird die zugehörige Grafik aus der Datei \PDFcontainer eingebunden.

```

189 \PackageInfo{pst-pdf}{MODE: \ppf@TeX@mode\space (pdfTeX mode)}%
    Verhindert pdfTEXs Warnung Non-PDF special ignored!.
190 \if@ppf@PST@used
191     \let\ppf@temp\AtBeginDvi\let\AtBeginDvi\@gobble
192     \RequirePackage{pstricks}\let\AtBeginDvi\ppf@temp
193 \fi

194 \@temptokena{%
195     \let\Gin@PS@file@header\@gobble\let\Gin@PS@literal@header\@gobble
196     \let\Gin@PS@raw\@gobble\let\Gin@PS@restored\@gobble
197     \@ifundefined{PSTricksLoaded}{-}{%

```

Für PSTricks < 2.0 nötig.

```

198     \PSTricksOff
199     \@ifundefined{c@lor@to@ps}{\def\c@lor@to@ps#1 #2\@{}}{}}%

```

PostScript-Ausgabe jetzt verhindern und später noch einmal.

```

200 \the\@temptokena
201 \expandafter\AtBeginDocument\expandafter
202     {\the\@temptokena\@temptokena{}}%
203 \@ifundefined{PSTricksLoaded}{-}{%

```

Zum Parsen der Argumente von PSTricks' \pst@object laden wir preview im active-Modus, restaurieren aber die standardmäßigen Definitionen von \output und \shipout. \pr@startbox und \pr@endbox dienen hier nur dazu, um \pst@object wirkungslos zu machen und stattdessen die zugehörige Grafik aus der Containerdatei einzuladen. Derzeit werden nur maximal 3 optionale Parameter in runden Klammern am Ende von \pst@object unterstützt, was für viele, aber nicht für alle Fälle ausreichend ist.

```

204 \newtoks\ppf@output
205 \ppf@output\expandafter{\the\output}%
206 \let\ppf@nofiles=\nofiles \let\nofiles=\relax
207 \let\ppf@shipout=\shipout

```

```

208 \RequirePackage[active]{preview}[2005/01/29]%
209 \let\shipout=\ppf@shipout \let\ppf@shipout=\relax
210 \let\nofiles=\ppf@nofiles \let\ppf@nofiles=\relax
211 \output\expandafter{\the\ppf@output} \ppf@output{}%

\pr@startbox, \pr@endbox: Gegenüber Originaldefinition vereinfacht.

212 \long\def\pr@startbox#1#2{%
213   \ifpr@outer
214     \toks@{#2}%
215     \edef\pr@cleanup{\the\toks@}%
216     \setbox\@tempboxa\vbox\bgroup
217     \everydisplay{}%
218     \pr@outerfalse%
219     \expandafter\@firstofone
220   \else
221     \expandafter\@gobble
222   \fi{#1}}%
223 \def\pr@endbox{%
224   \egroup
225   \setbox\@tempboxa\box\voidb@x
226   \ppf@@getpicture
227   \pr@cleanup}%

```

(Siehe auch identische Definition im DVI-Modus.)

```

228 \AtBeginDocument{%
229   \ifundefined{pst@object}{}%
230   {%
231     \PreviewMacro[{}*[]%
232       ?\bgroup{#1}{#1}}{}%
233       ?\bgroup{#1}{#1}}{}%
234       ?({#1})(#1)}{}%
235       ?({#1})(#1)}{}%
236       ?({#1})(#1)}{}%
237     }}\pst@object}}%
238   }%
239 }%

```

Es werden auch die im DVI-Modus erlaubten Endungen von Grafikdateien benötigt.

```

240 \begingroup
241   \input{dvips.def}%
242   \edef\x{\endgroup\def\noexpand\ppf@other@extensions{\Gin@extensions}}%
243   \x

```

Dummy-Definition für die im DVI-Modus gültigen Dateitypen.

```

244 \DeclareGraphicsRule{*}{eps}{*}{}%
245 \define@key{Gin}{innerframe}[true]{%
246   \lowercase{\Gin@boolkey{#1}}{innerframe}}%
247 \define@key{Gin}{frame}[true]{%
248   \lowercase{\Gin@boolkey{#1}}{frame}}%
249 \define@key{Gin}{ignore}[true]{%
250   \lowercase{\Gin@boolkey{#1}}{ignore}}%
251 \define@key{Gin}{frame@}{%

```

(Nur intern zu benutzen!)

```

252 \edef\@tempa{\toks@{\noexpand\frame{\the\toks@}}}%
253 \ifcase#1\relax
254 \ifGin@innerframe\else\let\@tempa\relax\fi
255 \or
256 \ifGin@frame\else\let\@tempa\relax\fi
257 \fi
258 \@tempa
259 }%
260 \define@key{Gin}{showname}[true]{%
261 \lowercase{\Gin@boolkey{#1}}{showname}}%
262 \define@key{Gin}{namefont}{%
263 \begingroup
264 \temptokena\expandafter{\ppf@namefont#1}%
265 \edef\x{\endgroup\def\noexpand\ppf@namefont{\the\@temptokena}}%
266 \x
267 }%
268 \newcommand*\ppf@filename{}%
269 \newcommand*\ppf@namefont{\tiny\ttfamily}%
270 \newcommand*\ppf@Gin@keys{}%
271 \let\ppf@Gin@setfile\Gin@setfile

```

`\Gin@setfile` Realen Dateinamen und ggf. Seitenzahl zur späteren Verwendung merken.

```

272 \def\Gin@setfile#1#2#3{\ppf@Gin@setfile{#1}{#2}{#3}%
273 \xdef\ppf@filename{%
274 #3\ifx\GPT@page\empty\else(\GPT@page)\fi}}%

```

`\Gin@ii` Auswertung der Optionen “frame”, “ignore” usw. sowie weiterer Spezialfälle.

```

275 \def\Gin@ii[#1]#2{%
276 \begingroup

```

Der Wert `\ifGin@innerframe` muss bereits vor Zeichnen des inneren Rahmens bekannt sein. Die Werte für `\ifGin@showname` und `\ppf@namefont` müssen auch nach Darstellung der Grafik verfügbar sein. Daher durch eine Gruppe geschützt vorher Auswertung der Optionen.

```

277 \@temptokena{#1}\def\ppf@tempb{#2}%

```

Leerer Dateiname beim Aufruf von `\usepicture` aus.

```

278 \ifx\ppf@tempb\empty\else
279 \ppf@is@pdfTeX@graphic{#2}{\Gin@extensions}{\ppf@other@extensions}%

```

Grafiken aus Containerdatei sind bereits skaliert usw. Nicht noch einmal, daher optionalen Parameter ignorieren.

```

280 {%
281 \setkeys{Gin}{#1}%
282 \ifx\ppf@tempb\PDFcontainer
283 \@temptokena{page=\GPT@page}%
284 \fi
285 }%
286 {%
287 \refstepcounter{pspicture}%
288 \@temptokena{page=\the\c@pspicture}\def\ppf@tempb{\PDFcontainer}%
289 }%
290 \fi
291 \ifGin@ignore\else

```

“frame@@=0” = innerer Rahmen, “frame@@=1” = äußerer Rahmen.

```

292     \edef\@tempa{\noexpand\ppf@Gin@ii[frame@@=0,\the\@temptokena,
293         frame@@=1]{\ppf@tempb}}}%
294     \@tempa
295     \ifGin@showname
296         \ppf@namefont
297         \raisebox{-\ht\strutbox}[Opt][Opt]{\llap{\ppf@filename}}%
298         \gdef\ppf@filename{}%
299     \fi
300 \fi
301 \endgroup
302 }%

303 \IfFileExists{\PDFcontainer}%
304 {%

```

\ppf@container@max Die Anzahl der in der Containerdatei enthaltenen Seiten.

```

305     \pdfximage{\PDFcontainer}%
306     \edef\ppf@container@max{\the\pdflastximagepages}%

307     \AtEndDocument{%
308         \ifnum\c@pspicture>\z@

```

Warnung ist nur sinnvoll, wenn überhaupt Grafiken benötigt wurden.

```

309         \ifnum\c@pspicture=\ppf@container@max\else
310             \PackageWarningNoLine{pst-pdf}{%
311                 ‘\PDFcontainer’ contains \ppf@container@max\space pages
312                 \MessageBreak but \the\c@pspicture\space pages are requested:
313                 \MessageBreak File ‘\PDFcontainer’ is no more valid!
314                 \MessageBreak Recreate it
315             }%
316         \fi
317     \fi
318 }%
319 }%
320 {%
321     \def\ppf@container@max{0}%
322     \AtEndDocument{%
323         \ifnum\c@pspicture>\z@
324             \filename@parse{\PDFcontainer}%
325             \PackageWarningNoLine{pst-pdf}{%
326                 File ‘\PDFcontainer’ not found.\MessageBreak
327                 Use the following commands to create it:\MessageBreak
328                 -----
329                 \MessageBreak
330                 latex \jobname.tex\MessageBreak
331                 dvips -o \filename@base.ps \jobname.dvi\MessageBreak
332                 ps2pdf \filename@base.ps\MessageBreak
333                 -----
334             }%
335         \fi
336     }%
337 }%

```

`\ppf@isnum` Ist Parameter #1 numerisch, werden Anweisungen in #2 sonst die in #3 ausgeführt (siehe `bibtopic.sty`).

```
338 \newcommand\ppf@isnum[1]{%
339   \if!\ifnum9<1#1!\else_\fi\expandafter\@firstoftwo
340   \else\expandafter\@secondoftwo\fi}%
```

`psmatrix` Beide Umgebungen ignorieren ihren Inhalt und laden stattdessen die zugehörige Grafik aus der Containerdatei. Auf den Wert des dabei benutzten Zählers (`pspicture`) kann per `\label/\ref` zugegriffen werden.

`postscript`

```
341 \newcommand*\ppf@set@mode{%
342 \newcommand*\ppf@test@mmode{%
343 \ifmmode
344   \ifinner
345     \let\ppf@set@mode=$%
346   \else
347     \def\ppf@set@mode{${}%
348   \fi
349 \else
350   \let\ppf@set@mode=\@empty
351 \fi
352 }

353 \RequirePackage{environ}%
354 \newenvironment{postscript}[1] []{%
355   \def\@tempa{postscript}%
356   \ifx\@tempa\@currenvir
357     \def\ppf@Gin@keys{#1}%
358   \else
359     \def\ppf@Gin@keys{}%
360   \fi
361   \ppf@@getpicture
362   \Collect@Body\@gobble}{}%
363 \AtBeginDocument{%
364   \@ifundefined{PSTricksLoaded}{-}{%
365     \def\pst@@@picture[#1](#2,#3)(#4,#5){\postscript}%
366     \def\endpspicture{\endpostscript\endgroup}%
367     \@ifundefined{psmatrix}{-}{%
368       \let\psmatrix=\postscript
369       \let\endpsmatrix=\endpostscript}%
370   }%
371   \@ifundefined{pfx@includegraphics}{-}{%
```

Die im pdfTeX-Modus unnütze Umdefinition von `\includegraphics` (Paket `psfrag`) führt zu zweifachem Einfügen des Ergebnisses, weshalb die Originaldefinition wiederhergestellt wird.

```
372   \let\includegraphics=\pfx@includegraphics
373   \def\pfx@includegraphics#1#2{\ppf@@getpicture}%
374 }%
375 }%
```

`\savepicture` Speichert die Nummer der aktuellen Grafik in einem Makro mit Namen `\ppf@@@#1`.

```

376 \def\savepicture#1{%
377 \expandafter\xdef\csname ppf@@#1\endcsname{\the\pdflastximage}}%

```

`\usepicture` Fügt Grafik mit symbolischem Namen #2 ein. Der Name muss vorher mit `\savepicture{<Name>}` vereinbart worden sein. Statt des Namens kann auch eine Zahl angegeben werden, die dann direkt eine Grafik aus der Containerdatei adressiert. Der optionale Parameter #1 entspricht dem bei `\includegraphics`.

```

378 \renewcommand*\usepicture[2][]{%
379 \ifundefined{ppf@@#2}%
380 {%
381 \ppf@isnum{#2}%
382 {\ppf@getpicture{#1}{#2}}%
383 {\@latexerror{picture ‘#2’ undefined}\@ehc}%
384 }%
385 {%
386 \begingroup
387 \def\Gin@include@graphics##1{%
388 \xdef\ppf@filename{#2}%
389 \setbox\z@\hbox{\pdfrefximage\@nameuse{ppf@@#2}}%
390 \Gin@nat@height\ht\z@ \Gin@nat@width\wd\z@
391 \def\Gin@llx{0} \let\Gin@lly\Gin@llx
392 \Gin@defaultbp\Gin@urx{\Gin@nat@width}%
393 \Gin@defaultbp\Gin@ury{\Gin@nat@height}%
394 \Gin@bboxtrue\Gin@viewport@code
395 \Gin@nat@height\Gin@ury bp%
396 \advance\Gin@nat@height-\Gin@lly bp%
397 \Gin@nat@width\Gin@urx bp%
398 \advance\Gin@nat@width-\Gin@llx bp%
399 \Gin@req@sizes
400 \ht\z@\Gin@req@height \wd\z@\Gin@req@width
401 \leavevmode\box\z@}%
402 \define@key{Gin}{type}{}%
403 \includegraphics[scale=1,#1]{}%
404 \endgroup
405 }}%

```

`\ppf@getpicture` Fügt die Seite (Grafik) mit Nummer #2 aus der Containerdatei ein. Parameter #1: Optionen wie bei `\includegraphics`.

```

406 \newcommand*\ppf@getpicture[2]{%
407 \@tempcnta=#2\relax%
408 \ifnum\@tempcnta>\ppf@container@max
409 \PackageWarningNoLine{pst-pdf}{%
410 pspicture No. \the\@tempcnta\space undefined}%
411 \else
412 \includegraphics[draft=\ppf@draft,#1,page=\the\@tempcnta]%
413 {\PDFcontainer}%
414 \fi
415 \gdef\ppf@Gin@keys{}}%

```

`\ppf@@getpicture` Fügt die nächste Seite (Grafik) aus der Containerdatei ein.

```

416 \newcommand*\ppf@@getpicture{%
417 \ifpr@outer
418 \refstepcounter{pspicture}%

```

```

419     \expandafter\ppf@getpicture\expandafter{\ppf@Gin@keys}%
420     {\the\c@ppicture}%
421     \fi}%

```

pst-pdf-defs Umgebung, die keine eigene Gruppe aufmacht. Innerhalb der Umgebung bekommt das Zeichen & den Kategoriecode „other“. Gedacht für eigene Makrodefinitionen, die z. B. eine `psmatrix` enthalten.

```

422 \renewenvironment*{pst-pdf-defs}%
423 {%
424     \endgroup
425 %     ??? \@currentvline
426     \chardef\ppf@temp=\catcode'\&%
427     \@makeother\&%
428 }{%
429     \catcode'\&=\ppf@temp
430     \begingroup
431     \def\@currentvir{pst-pdf-defs}%
432 }
433 \else

```

3.5 Inactiver Modus

Es werden nur die Pakete `pstricks` und `graphicx` geladen – keine weitere Einflussnahme. Die Paketoption „inactive“ sowie der \TeX -Compiler erzwingen diesen Modus.

```

434 \PackageInfo{pst-pdf}{MODE: \ppf@TeX@mode\space (inactive mode)}%
435 \newenvironment{postscript}[1][]{\ignorespaces}{}
436 \let\ppf@is@pdfTeX@graphic\relax
437 \fi

438 \InputIfFileExists{pst-pdf.cfg}{%
439     \PackageInfo{pst-pdf}{Local config file pst-pdf.cfg used}}{}
440 \endpackage

```