

PSTricks:
PostScript macros for Generic TeX
Documented Code

Timothy Van Zandt¹

25 March 191997
Version 0.93a-97

PSTricks is a collection of PostScript macros that is compatible with most TeX macro packages, including Plain TeX and L^ATeX. Included are macros for color, graphics, rotation and overlays.

This is the documented code. There is also a *User's Guide* and a read-me file.

¹Author's address: Department of Economics, Princeton University, Princeton, NJ 08544-1021, USA. Internet: tvz@Princeton.EDU

Contents

| | | |
|----|--|----|
| 1 | Disclaimers, Guidelines and Other Comments | 1 |
| 2 | Preliminaries | 2 |
| 3 | Error messages | 3 |
| 4 | Scratch registers | 4 |
| 5 | Useful hacks | 4 |
| 6 | Arithmetic | 5 |
| 7 | Configuration file | 5 |
| 8 | PostScript header file | 6 |
| 9 | PostScript hacks | 8 |
| 10 | Converting T _E X things to PostScript | 9 |
| 11 | Colors | 11 |
| 12 | Setting graphics parameters | 13 |
| 13 | Dimensions | 14 |
| 14 | Normal Coordinates and angles | 15 |
| 15 | Special coordinates and angles | 16 |
| 16 | Basic graphics parameters | 18 |
| 17 | Line styles | 20 |
| 18 | Fill styles | 23 |
| 19 | Arrowheads and t-bars | 25 |
| 20 | Graphics objects: processing arguments | 30 |
| 21 | Graphics objects: Basics T _E X macros | 31 |
| 22 | Custom graphics | 36 |
| 23 | Graphics objects: Basic PostScript macros | 40 |
| 24 | Interpolated curves | 43 |
| 25 | Dots | 46 |
| 26 | Lines and polygons | 48 |
| 27 | Curves | 50 |
| 28 | Grids | 51 |
| 29 | LR-box commands | 54 |
| 30 | Frame boxes | 56 |

| | | |
|-----------|---|-----------|
| 31 | Circles, discs and ellipses | 60 |
| 32 | Repetition | 64 |
| 33 | Scaling | 65 |
| 34 | Rotation: The simple version | 66 |
| 35 | \rput and company | 66 |
| 35.1 | Reference point | 67 |
| 35.2 | Rotation | 68 |
| 35.3 | Translation | 69 |
| 35.4 | The real thing | 69 |
| 36 | \uput and company | 70 |
| 37 | Pictures | 72 |
| 38 | Overlays | 73 |
| 39 | Configuration file – revisited | 75 |
| | I pst-node.doc | 76 |
| 40 | Node header | 76 |
| 41 | Nodes | 76 |
| 42 | Node connections: Preliminaries | 80 |
| 43 | Node connections: The real thing | 83 |
| 44 | Node Labels | 89 |
| 45 | Node coordinates | 90 |

1 Disclaimers, Guidelines and Other Comments

Disclaimer These macros are extensive and were written hurriedly. Only modest attempts have been made to clean up and optimize the code. The internals may change substantially until version 1.0 comes up.

PostScript Guidelines The following guidelines were followed for macros using PostScript `\special`'s:

1. Almost no `gsave` and `grestore` commands are used (reducing the likelihood of conflicts with dvi-to-ps drivers or an unmatched `gsave` or `grestore` ending up on a page).
2. Most end-user macros (those without `@`) have error-checking so that bad arguments or other misuse will not generate PostScript errors.

Macros A “macro” means any command sequence that is documented in this with a heading entry. Macros with `@` are internal, and others are part of the user interface. Commands that are not in the heading preceding their definition are internal commands of a macro, and are not meant to be used directly by other macros.

Local and global variables There are various classes of scratch registers and commands:

Global These can be changed using `\global`, etc.

```
\pst@tempg
\upst@temph
\upst@ding
\upst@dimh
\upst@cntg
\upst@cnth
\upst@boxg
```

Local-I These cannot be changed with `\global`, but otherwise there are no restrictions.

```
\next
\upst@tempa
```

Local-I Changes to these must be local to the macro in which they occur (be grouping).

```
\pst@tempa
\upst@tempb
\upst@tempc
\upst@tempd
```

Local-II Changes to these must be local to the macro in which they occur, and it must be possible to use these as arguments of macros.

```
\pst@dima
\upst@dimb
\upst@dimc
\upst@dimd
\upst@ifpst
```

There is one exception. When using these in coordinates that are processed directly as Cartesian coordinates rather than with `\pst@getcoor`, they must be used in this order:

```
(\pst@dima,\pst@dimb)(\pst@dimc,\pst@dimd)
```

Shared These are used to share information between macros. Their value may be set by one macro and then used by another. Use with care. Do not set with `\global`.

| <i>command</i> | <i>usage</i> |
|-------------------------|---|
| <code>\pst@hbox</code> | Box created and manipulated in HR-box macros. |
| <code>\pst@coor</code> | PostScript code for a coordinate. |
| <code>\pst@angle</code> | PostScript code for an angle. |
| <code>\pst@rot</code> | PostScript code for a rotation angle. |
| <code>\if@star</code> | This is a flag to keep track of optional <code>*</code> . |

Plain TeX commands The commands

```
\newbox
\newcount
\newdimen
\newif
\loop ... \repeat ... \fi
\z@
\sixt@@n
```

are defined in `plain.tex` are part of most macro packages. PSTricks assumes that they are defined. Other than these, PSTricks only makes use of TeX primitives.

Dividing the file *Breaking up the file* `pstricks.tex` can be broken up into the following components:

Basics (Including color and simple rotation.) Sections 2, 7, 8, ??, 11, ??, 34 and 39.

Graphics Sections 12, 21, 17, 19, 26, 28, 18, ??, 30 and 31. Requires also **Basics**.

Rotation (Including picture environment.) Sections ??, ??, ?? and 37. Requires also **Basics**.

2 Preliminaries

Check whether file has been loaded already.

```
1 \csname PSTricksLoaded\endcsname
2 \let\PSTricksLoaded\endinput
```

Take care of the catcode of `@`:

```
3 \edef\PstAtCode{\the\catcode'\@}
4 \catcode'\@=11\relax
```

Here are some hacks borrowed from L^AT_EX, which are defined if L^AT_EX is not being used.

```
5 \expandafter\ifx\csname @latexerr\endcsname\relax
6 \long\def@ifundefined#1#2#3{\expandafter\ifx\csname
7   #1\endcsname\relax#2\else#3\fi}
8 \def@namedef#1{\expandafter\def\csname #1\endcsname}
9 \def@nameuse#1{\csname #1\endcsname}
10 \def@eha{%
11   Your command was ignored.^^J
12   Type \space I <command> <return> \space to replace
13   it with another command,^^J
14   or \space <return> \space to continue without it.}
15 \def@spaces{\space\space\space\space}
16 \def\typeout#1{\immediate\write\@unused{#1}}
17 \alloc@7\write\chardef\sixt@n\@unused
18 \def@empty{}
19 \def@gobble#1{}
20 \def@nnil{\@nil}
21 \def@ifnextchar#1#2#3{%
22   \let\@tempe#1\def\@tempa{#2}\def\@tempb{#3}\futurelet\@tempc\@ifnch}
23 \def@ifnch{%
24   \ifx\@tempc\@sptoken
25     \let\@tempd\@xifnch
26   \else
27     \ifx\@tempc\@tempe \let\@tempd\@tempa \else \let\@tempd\@tempb \fi
28   \fi
29   \@tempd}
30 \begingroup
31   \def\:{\global\let\@sptoken= } \:
32   \def\:\@xifnch \expandafter\gdef\:\: {\futurelet\@tempc\@ifnch}
33 \endgroup
34 \fi
```

Announce that the file is being loaded:

```
35 \typeout{'PSTricks' v\fileversion\space\space <\filedate> (tvz)}
```

3 Error messages

`\@pstrickserr`

`\@pstrickserr` is analogous to `\@latexerr`.

```
36 \def\@pstrickserr#1#2{%
37   \begingroup
38     \newlinechar'\^^J
39     \edef\pst@tempc{#2}%
40     \expandafter\errhelp\expandafter{\pst@tempc}%
41     \typeout{%
42       PSTricks error. \space See User's Guide for further information.^^J
43       \@spaces\@spaces\@spaces\@spaces
44       Type \space H <return> \space for immediate help.}%
45     \errmessage{#1}%
46   \endgroup}
```

`\@ehpa, \@ehpb, \@ehpc`

Here are some extra `\errhelp` message:

```
47 \def\@ehpa{%
48   Your command was ignored. Default value substituted.^^J
49   Type \space <return> \space to procede.}
50 \def\@ehpb{%
51   Your command was ignored. Will recover best I can.^^J
52   Type \space <return> \space to procede.}
53 \def\@ehpc{%
54   You better fix this before proceeding.^^J
55   See the PSTricks User's Guide or ask your system administrator for help.^^J
56   Type \space X <return> \space to quit.}
```

`\pst@misplaced`

```
57 \def\pst@misplaced#1{\@pstrickserr{Misplaced \string#1 command}\@ehpb}
```

4 Scratch registers

```
58 \newdimen\pst@dima
59 \newdimen\pst@dimb
60 \newdimen\pst@dimc
61 \newdimen\pst@dimd
62 \newdimen\pst@ding
63 \newdimen\pst@dimh
64 \newbox\pst@hbox
65 \newbox\pst@boxg
66 \newcount\pst@cna
67 \newcount\pst@cntb
68 \newcount\pst@cntc
69 \newcount\pst@cntd
70 \newcount\pst@cntg
71 \newcount\pst@cna
72 \newif\if@pst
```

5 Useful hacks

`\pst@ifstar, \if@star`

```
73 \newif\if@star
74 \def\pst@ifstar#1{%
75   \@ifnextchar*\@startrue\def\next*{#1}\next}{\@starfalse#1}}
```

`\pst@expandafter`

```
76 \def\pst@expandafter#1#2{%
77   \def\next{#1}%
78   \edef\@tempa{#2}%
79   \ifx\@tempa\@empty
80     \@pstrickserr{Unexpected empty argument!}\@ehpb
81     \def\@tempa{\@empty}%
82   \fi
83   \expandafter\next\@tempa}
```

6 Arithmetic

`\pst@dimtonum`, `\pst@@dimtonum`

This macro strips the value of `#1`, a dimension register, of the `pt`, and assigns the result to `#2`, a command sequence. This is used for arithmetic and for converting `TeX` dimensions to PostScript.

```
84 \def\pst@dimtonum#1#2{\edef#2{\pst@@dimtonum#1}}
85 \def\pst@@dimtonum#1{\expandafter\pst@@@dimtonum\the#1}
86 {\catcode'\p=12 \catcode'\t=12 \global\@namedef{pst@@@dimtonum}#1pt{#1}}
```

`\pst@pyth`

This is a piecewise-linear approximation to $(\#1^2 + \#2^2)^{1/2}$. The answer is assigned to `#3`. All arguments should be dimension registers.

```
87 \def\pst@pyth#1#2#3{%
88   \ifdim#1>#2\pst@@pyth#1#2#3\else\pst@pyth#2#1#3\fi}
89 \def\pst@@pyth#1#2#3{%
90   \ifdim#1>9#2%
91     #3=#1\advance#3 .2122#2%
92   \else
93     #3=.8384#1\advance#3 .5758#2%
94   \fi}
```

`\pst@divide`

This computes $\#3 = \#1 / \#2$ reasonably quickly. `#1` and `#2` should be dimensions, and `#3` should be a command sequence.

```
95 \def\pst@divide#1#2#3{%
96   \begingroup
97     \pst@dimg=#1\relax\pst@dimh=#2\relax
98     \pst@cna=\pst@dimg
99     \pst@cntb=1073741824
100    \pst@cntc=65536
101    \def\pst@tempa{\fi\ifnum}%
102    \loop\ifnum\pst@cna<\pst@cntb
103      \pst@tempa\pst@cntc>\@ne
104      \multiply\pst@cna2\divide\pst@cntc2
105    \repeat
106    \divide\pst@dimh\pst@cntc
107    \divide\pst@cna\pst@dimh
108    \global\pst@dimg\number\pst@cna sp
109  \endgroup
110  \pst@dimtonum\pst@dimg#3}
```

7 Configuration file

`\pst@configerr`

```
111 \def\pst@configerr#1{%
112   \@pstrickserr{\string#1 not defined in pstricks.con}\@ehpc}
```



```

113 \def\pstVerb#1{\pst@configerr\pstVerb}
114 \def\pstverb#1{\pst@configerr\pstverb}
115 \def\pstverbscale{\pst@configerr\pstverbscale}
116 \def\pstrotate{\pst@configerr\pstrotate}
117 \def\pstheader#1{\pst@configerr\pstheader}
118 \def\pstdriver{\pst@configerr\pstdriver}
119 \@ifundefined{pstcustomize}%
120   {\def\pstcustomize{\endinput\let\pstcustomize\relax}}{}
121 \input pstricks.con

\PSTricksOff

122 \newif\ifPSTricks
123 \PSTrickstrue
124 \def\PSTricksOff{%
125   \def\pstheader##1{}%
126   \def\pstverb##1{}%
127   \def\pstVerb##1{}%
128   \PSTricksfalse}

```

8 PostScript header file

`\pst@def`, `\pst@ATH`, `\pst@useheader`

There are three ways to use PSTricks:

1. Use `pstricks.doc` directly. No header file is used.
2. Use `\pst-make.tex` to generate a stripped input file for use without a header.
3. Use `\pst-make.tex` to generate a header and a stripped input file for use with the header.

PSTricks has been optimized for use with a header file (and the difference is speed and memory is very significant), but the flexible system described above makes it easier to maintain the code and allows one to use PSTricks with a DVI-to-PS driver that does not support header files.

The following commands should be used in this `.doc` file for PostScript macros and other goodies related to the header file. Their behavior for each of the three cases list above is given below. These commands should always come *at the beginning of the line*, and should not inside conditionals.

- `\pst@def{foo}<bar>`
 1. `\tx@foo` is defined to be `bar`.
 2. Writes

```

\def\tx@foo{bar}

```

to `pstricks.tex`.
 3. Writes

```

/foo { bar } def

```

to `pstricks.pro` and

```

\def\tx@foo{foo}

```

to `pstricks.tex`.

- `\pst@ATH<foo>`

1. Gobbles `foo`.
2. Ignores line.
3. Writes `foo` to `pstricks.pro`.

Note: `\pst@ATH` must come at the beginning of the line.

- `\ifx\pst@useheader\iftrue foo \else bar \fi`

1. Ignores `foo` and includes `bar`.
2. Ignores `foo` and processes `bar`.
3. Processes `foo` and ignores `bar`.

Note: `\ifx\pst@useheader\iftrue`, `\else` and `\fi` must each be on their own line.

`pst@make.tex` can be used to process other files at well, in the right order.

```
129 \@ifundefined{pst@def}{\def\pst@def#1<#2>{\@namedef{tx@#1}{#2 }}}{  
130 \@ifundefined{pst@ATH}{\def\pst@ATH<#1>{}}{}
```

`\pst@dict`

The PostScript dictionary `tx@Dict` is set up in the header file, if one is used. Otherwise, it is set up with each procedure that uses the dictionary, if it is not currently defined.

```
131 \pst@ATH<\/% Version \fileversion, \filedate.>  
132 \pst@ATH<\/% For use with \pstdriver.>  
133 \pst@ATH</tx@Dict 200 dict def tx@Dict begin>  
134 \pst@ATH</ADict 25 dict def>  
135 \pst@ATH</CM { matrix currentmatrix } bind def>  
136 \pst@ATH</SLW /setlinewidth load def>  
137 \pst@ATH</CLW /currentlinewidth load def>  
138 \pst@ATH</CP /currentpoint load def>  
139 \pst@ATH</ED { exch def } bind def>  
140 \pst@ATH</L /lineto load def>  
141 \pst@ATH</T /translate load def>  
142 \ifx\pst@useheader\iftrue  
143   \pstheader{pstricks.pro}  
144   \def\pst@dict{tx@Dict begin }  
145   \def\pst@theheaders{pstricks.pro}  
146 \else  
147   \def\pst@dict{%  
148     /tx@Dict where  
149     { pop tx@Dict begin}  
150     { userdict begin  
151       /tx@Dict 200 dict def  
152     end  
153     tx@Dict begin  
154       /ADict 25 dict def  
155       /CM { matrix currentmatrix } bind def
```

```

156      /SLW /setlinewidth load def
157      /CLW /currentlinewidth load def
158      /CP /currentpoint load def
159      /ED { exch def } bind def
160      /L /lineto load def }
161  ifelse }
162  \def\pst@theheaders{}%
163 \fi

```

```

\pst@Verb
164 \def\pst@Verb#1{\pstVerb{\pst@dict #1 end}}

```

9 PostScript hacks

```
\tx@Atan, \tx@Div
```

These are variants of atan, and div, that recover when result is not defined.

```

165 \pst@def{Atan}</atan load stopped { pop pop 0 } if>
166 \pst@def{Div}<dup 0 eq { pop } { div } ifelse>

```

```

\tx@NET
167 \pst@def{NET}<neg exch neg exch T>

```

```

\tx@Pyth
A B Pyth  $(A^2 + B^2)^{1/2}$ 
168 \pst@def{Pyth}<dup mul exch dup mul add sqrt>

```

```

\tx@PtoC
Polar to Cartesian:
r a PtoC x y
169 \pst@def{PtoC}<2 copy cos mul 3 1 roll sin mul>

```

```

\tx@PathLength
PathLength is taken from the Blue Book. It leaves on the stack the length of the current
path.
170 \pst@def{PathLength@}<%
171 /z z y y1 sub x x1 sub \tx@Pyth add def
172 /y1 y def /x1 x def>
173 \pst@def{PathLength}<%
174 flattenpath /z 0 def
175 { /y1 ED /x1 ED /y2 y1 def /x2 x1 def } % moveto
176 { /y ED /x ED \tx@PathLength@ } % lineto
177 {} % curveto; ignore because of flattenpath.
178 { /y y2 def /x x2 def \tx@PathLength@ } % closepath
179 pathforall z>

```

10 Converting T_EX things to PostScript

`\pst@number`, `\tx@STP`, `\tx@STV`

- PSTricks' PostScript unit is 1pt, rather than 1bp, because this is more efficient.
- `\pst@number{<dimen register>}` converts *dimen* to PostScript, in points (pt).
- `\tx@STP` scales the DVI-to-PS driver's `\pstverb` environment to points (pt).
- `\tx@STV` scales the DVI-to-PS driver's ungrouped PostScript `\special` environment (`\pstVerb`) to points (pt).

```
180 \pst@ding=\pstunit\relax
181 \ifdim\pst@ding=1bp
182   \def\pst@stp{.996264 dup scale}
183 \else
184   \edef\pst@stp{1 \pst@@dimtonum\pst@ding\space div dup scale}
185 \fi
186 \pst@def{STP}<\pst@stp>
187 \pst@def{STV}<\pstverbscale\space\tx@STP>
188 \def\pst@number#1{\pst@@dimtonum#1\space}
```

`\pst@checknum`

The first argument of `\pst@checknum` should be a number, and the second argument is a command. There are three possible outcomes:

1. The number is suitable for PostScript consumption, the command is set to the number, and `\pst@num` is set to 1 if the number is positive and to 2 if the number is negative.
2. `\pst@checknum` detects that the number is not suitable for PostScript; `\pst@num` is set to 0, an error is given, and the command is defined to be 0 .
3. The number is not suitable for PostScript consumption, but `\pst@checknum` does not detect this. `\pst@num` is set to 1 or 2, and the command is set to some number that *is* suitable for PostScript.

A trailing space is always added.

`\pst@checknum` should generate no extraneous errors nor output, even if the first argument is a bad number.

This macro is probably pretty close to optimal for what it does, as many variations have been tried.

```
189 \def\pst@checknum#1#2{%
190   \edef\next{#1}%
191   \ifx\next\@empty
192     \let\pst@num\z@
193   \else
194     \expandafter\pst@@checknum\next..\@nil
195   \fi
196   \ifnum\pst@num=\z@
197     \@pstrickserr{Bad number: '#1'. 0 substituted.}\@ehpa
```

```

198   \def#2{0 }%
199   \else
200     \edef#2{\ifnum\pst@num=2 -\fi\the\pst@cntg.%
201     \expandafter\@gobble\the\pst@cnth\space}%
202   \fi}
203 \def\pst@@checknum{%
204   \@ifnextchar-%
205     {\def\pst@num{2}\expandafter\pst@@checknum\@gobble}%
206     {\def\pst@num{1}\pst@@checknum}}
207 \def\pst@@@checknum#1.#2.#3\@nil{%
208   \afterassignment\pst@@@checknum\pst@cntg=0#1\relax\@nil
209   \afterassignment\pst@@@checknum\pst@cnth=1#2\relax\@nil}
210 \def\pst@@@checknum#1\relax\@nil{%
211   \ifx\@nil#1\@nil\else\let\pst@num\z@\fi}

```

`\pst@getnumii`, `\pst@getnumiii`, `\pst@getnumiv`

These are for processing comma-separated lists of numbers. They assign the numbers to `\pst@tempg`, `\pst@tempf`, etc. Use like

```
\pst@expandafter\pst@getnumiii{foo} {} {} {} {} \@ni
```

If there are too few numbers, an error results. If there are too many, the extra numbers are ignored.

```

212 \def\pst@getnumii#1 #2 #3\@nil{%
213   \pst@checknum{#1}\pst@tempg
214   \pst@checknum{#2}\pst@tempf}
215 \def\pst@getnumiii#1 #2 #3 #4\@nil{%
216   \pst@checknum{#1}\pst@tempg
217   \pst@checknum{#2}\pst@tempf
218   \pst@checknum{#3}\pst@tempj}
219 \def\pst@getnumiv#1 #2 #3 #4 #5\@nil{%
220   \pst@checknum{#1}\pst@tempg
221   \pst@checknum{#2}\pst@tempf
222   \pst@checknum{#3}\pst@tempj
223   \pst@checknum{#4}\pst@tempk}

```

`\pst@getdimnum`

Like `\pst@getnumii`, but first item is a dimension and second is a number.

```

224 \def\pst@getdimnum#1 #2 #3\@nil{%
225   \pssetlength\pst@ding{#1}%
226   \pst@checknum{#2}\pst@tempg}

```

`\pst@getscale`

`\pst@getscale` can have one or two numbers in its first argument.

```

227 \def\pst@getscale#1#2{%
228   \pst@expandafter\pst@getnumii{#1 #1} {} {} {} \@nil
229   \edef#2{\pst@tempg\space \pst@tempf\space scale }%
230   \ifdim\pst@tempg\p@=\z@

```

```

231   \@pstrickserr{Bad scaling argument ‘#1’}\@ehpa
232   \def#2{}%
233   \else
234     \ifdim\pst@temph\p@=\z@
235       \@pstrickserr{Bad scaling argument}\@ehpa
236       \def#2{}%
237     \else
238       \ifdim\pst@tempg\p@=\p@ \ifdim\pst@temph\p@=\p@ \def#2{\fi\fi
239     \fi
240   \fi}

```

`\pst@getint`

```

241 \def\pst@getint#1#2{%
242   \pst@cntg=#1\relax
243   \edef#2{\the\pst@cntg\space}}

```

`\pslbrace`, `\psrbrace`

When balanced braces are used, they work without problems in `\special`'s. `\pslbrace` and `\psrbrace` let you use unbalanced braces.

```

244 \begingroup
245   \catcode'\{=12
246   \catcode'\}=12
247   \catcode'\[=1
248   \catcode'\]=2
249   \gdef\pslbrace[{ ]
250   \gdef\psrbrace[} ]
251 \endgroup

```

11 Colors

`\@newcolor`

`\@newcolor{<color>}{<spec>}`, where *color* is a name and *spec* is the associated PostScript color specification, sets

- `<color>` to `\pst@color{<spec>}`, and
- `\color@<color>` to *spec*.

Then `<color>` can be used by the user to color text, etc., and `\color@<color>` is used by PSTricks graphics objects to find the specification for *color*.

```

252 \def\@newcolor#1#2{%
253   \expandafter\edef\csname #1\endcsname{\noexpand\pst@color{#2}}}%
254   \expandafter\edef\csname color@#1\endcsname{#2}%
255   \ignorespaces}

```

`\pst@color`, `\pst@endcolor`

The argument of `\pst@color` should be a PostScript command for setting the color; e.g., 0 `setgray`. It saves the command in `\pst@currentcolor`, and then switches to

`\pst@currentcolor` at the end of the current group. The color changes do not extend across pages, although this capability could be written into the output routines (so that `\pst@currentcolor` is set at the beginning of the page, and headers and footers begin with `\black`, etc.). Moving boxes cause problems, but there is no way around this until \TeX supports color internally.

```
256 \def\pst@color#1{%
257   \def\pst@currentcolor{#1}\pstVerb{#1}\aftergroup\pst@endcolor}
258 \def\pst@endcolor{\pstVerb{\pst@currentcolor}}
259 \def\pst@currentcolor{0 setgray}
```

`\altcolormode`, `\pst@grestore`

The color macros defined above can conflict with other color macros. `\altcolormode` sets up a different scheme that uses `gsave` and `grestore` to reset colors. This may reduce the likelihood of such conflict. It also makes moving boxes less of a problem, as long as the color command is itself grouped within the box. However, if the scope of a color command extends across pages in a \TeX input file, unmatched `gsave`'s and `grestore`'s will be left on pages, wreaking havoc on the output. `\pst@grestore` is defined to do various things that makes using `grestore` more robust.

```
260 \def\altcolormode{%
261   \def\pst@color##1{%
262     \pstVerb{gsave ##1}\aftergroup\pst@endcolor}%
263   \def\pst@endcolor{\pstVerb{\pst@grestore}}}
264 \def\pst@grestore{%
265   currentpoint
266   matrix currentmatrix
267   currentfont
268   grestore
269   setfont
270   setmatrix
271   moveto}
```

`\pst@usecolor`

This looks up the color specification.

```
272 \def\pst@usecolor#1{\csname color@#1\endcsname\space}
```

`\newgray`

`\newgray` uses PostScript's `setgray` operator.

```
273 \def\newgray#1#2{%
274   \pst@checknum{#2}\pst@tempg
275   \@newcolor{#1}{\pst@tempg setgray}}
```

`\newrgbcolor`

This works like `\newgray`, but the color specification should consist of 3 numbers rather than just 1, and the `setrgbcolor` operator is used.

```
276 \def\newrgbcolor#1#2{%
277   \pst@expandafter\pst@getnumiii{#2} {} {} {} {} \@nil
278   \@newcolor{#1}{\pst@tempg \pst@temph \pst@tempi setrgbcolor}}
```

`\newhsbcolor`

This is just like `\newrgbcolor`, but the `sethsbcolor` operator is used.

```
279 \def\newhsbcolor#1#2{%
280   \pst@expandafter\pst@getnumiii{#2} {} {} {} {} \@nil
281   \@newcolor{#1}{\pst@tempg \pst@temph \pst@tempi sethsbcolor}}
```

`\newcmykcolor`

This is like `\newrgbcolor`, the color specification consists of 4 numbers and the `setcmykcolor` operator is used.

```
282 \def\newcmykcolor#1#2{%
283   \pst@expandafter\pst@getnumiv{#2} {} {} {} {} {} \@nil
284   \@newcolor{#1}{\pst@tempg \pst@temph \pst@tempi \pst@tempj setcmykcolor}}
```

`\black`, `\darkgray`, `\gray`, `\lightgray`, `\white`

Here are some default gray definitions:

```
285 \newgray{black}{0}
286 \newgray{darkgray}{.25}
287 \newgray{gray}{.5}
288 \newgray{lightgray}{.75}
289 \newgray{white}{1}
```

`\red`, `\green`, `\blue`, `\yellow`, `\cyan`, `\magenta`

And some default rgb color definitions.

```
290 \newrgbcolor{red}{1 0 0}
291 \newrgbcolor{green}{0 1 0}
292 \newrgbcolor{blue}{0 0 1}
293 \newrgbcolor{yellow}{1 1 0}
294 \newrgbcolor{cyan}{0 1 1}
295 \newrgbcolor{magenta}{1 0 1}
```

12 Setting graphics parameters

`\psset`

For each *parameter=value* pair in its argument, `\psset` invokes

```
\psset@parameter{value}
```

The value is processed and typically stored in `\ps<parameters>` if the value is user-accessible and `\psk<parameter>` if not. `\psset` ignores spaces that follow the comma that separates key-value pairs.

When initializing *parameter* in this file, preferable use

```
\psset@parameter{value}
```


so that default values can be easily extracted for the *User's Guide*.

```
296 \def\psset#1{\@psset#1,\@nil\ignorespaces}
297 \def\@psset#1,{%
298   \@psset#1==\@nil
299   \@ifnextchar\@nil{\@gobble}{\@psset}}
300 \def\@psset#1=#2=#3\@nil{%
301   \@ifundefined{psset@#1}%
302     {\@pstrickserr{Graphics parameter ‘#1’ not defined.}\@ehpa}%
303     {\@nameuse{psset@#1}{#2}}}%
```

`\newpsstyle`

```
304 \def\psset@style#1{%
305   \@ifundefined{pscs@#1}%
306     {\@pstrickserr{Custom style ‘#1’ undefined}\@ehpa}%
307     {\@nameuse{pscs@#1}}}
308 \def\newpsstyle#1#2{\@namedef{pscs@#1}{\psset{#2}}}
```

`\@none`

Use to check when a parameter value is none.

```
309 \def\@none{none}
```

`\pst@getcolor`

This is used by various graphics parameters that are colors.

```
310 \def\pst@getcolor#1#2{%
311   \@ifundefined{color@#1}%
312     {\@pstrickserr{Color ‘#1’ not defined}\@eha}%
313     {\edef#2{#1}}}
```

13 Dimensions

`\psunit, \psxunit, \psyunit`

```
314 \newdimen\psunit \psunit 1cm
315 \newdimen\psxunit \psxunit 1cm
316 \newdimen\psyunit \psyunit 1cm
317 \let\psrunit\psunit
```

`\pssetlength, \psaddtolength, \pssetxlength, \pssetylength`

```
318 \def\pstunit@off{\let\@psunit\ignorespaces\ignorespaces}
319 \def\pssetlength#1#2{%
320   \let\@psunit\psunit
321   \afterassignment\pstunit@off
322   #1 #2\@psunit}
323 \def\psaddtolength#1#2{%
324   \let\@psunit\psunit
325   \afterassignment\pstunit@off
326   \advance#1 #2\@psunit}
327 \def\pssetxlength#1#2{%
```

```

328 \let\@psunit\psxunit
329 \afterassignment\pstunit@off
330 #1 #2\@psunit}
331 \def\pssetylength#1#2{%
332 \let\@psunit\psyunit
333 \afterassignment\pstunit@off
334 #1 #2\@psunit}

```

\psset@unit, \psset@xunit, \psset@yunit

```

335 \def\psset@unit#1{%
336 \pssetlength\psunit{#1}%
337 \psxunit=\psunit
338 \psyunit=\psunit}
339 \def\psset@runit#1{\pssetlength\psrunit{#1}}
340 \def\psset@xunit#1{\pssetxlength\psxunit{#1}}
341 \def\psset@yunit#1{\pssetylength\psyunit{#1}}

```

\pst@getlength, pst@@getlength

#1 is a TeX dimensions. \pst@getlength sets #2 to the PostScript code for #1, and \pst@@getlength set #2 to the TeX code for #1.

```

342 \def\pst@getlength#1#2{%
343 \pssetlength\pst@dimg{#1}%
344 \edef#2{\pst@number\pst@dimg}}
345 \def\pst@@getlength#1#2{%
346 \pssetlength\pst@dimg{#1}%
347 \edef#2{\number\pst@dimg sp}}

```

14 Normal Coordinates and angles

\pst@getcoor, \pst@coor

\pst@@getcoor should be defined to read a coordinate and convert it to PostScript, assigning the result to \pst@coor (including the trailing space).

\pst@getcoor invokes \pst@@getcoor and then sets its second argument to \pst@coor.

```

348 \def\pst@getcoor#1#2{\pst@@getcoor{#1}\let#2\pst@coor}
349 \def\pst@coor{0 0 }

```

\pst@getcoors, \pst@coors

\pst@getcoors reads coordinates until there are none left, adding them *in reverse order* to \pst@coors.

```

350 \def\pst@getcoors#1#2{%
351 \def\pst@aftercoors{\addto@pscode{#1 \pst@coors }#2}%
352 \def\pst@coors{}%
353 \pst@@getcoors}
354 \def\pst@@getcoors(#1){%
355 \pst@getcoor{#1}%
356 \edef\pst@coors{\pst@coor\pst@coors}%
357 \@ifnextchar{\pst@@getcoors}{\pst@aftercoors}}

```

`\pst@getangle, \pst@angle`

`\pst@getangle` should be defined to read an angle and convert it to PostScript, assigning the result to `\pst@angle` (including the trailing space).

`\pst@getangle` invokes `\pst@getangle` and then sets its second argument to `\pst@angle`.

```
358 \def\pst@getangle#1#2{\pst@getangle{#1}\let#2\pst@angle}
359 \def\pst@angle{0 }
```

`getcoor@c, \NormalCoor`

By default, coordinates are read as Cartesian coordinates by `\cartesian@coor`.

Angles are read as numbers, scaled by `\pst@angleunit`.

`\NormalCoor` sets these two defaults, and also defines the translation for the put commands to be done by T_EX using Cartesian coordinates.

```
360 \def\cartesian@coor#1,#2,#3\@nil{%
361   \pssetxlength\pst@ding{#1}%
362   \pssetylength\pst@dimh{#2}%
363   \edef\pst@coor{\pst@number\pst@ding \pst@number\pst@dimh}}
364 \def\NormalCoor{%
365   \def\pst@getcoor##1{\pst@expandafter\cartesian@coor{##1},\relax,\@nil}%
366   \def\pst@getangle##1{%
367     \pst@checknum{##1}\pst@angle
368     \edef\pst@angle{\pst@angle \pst@angleunit}}%
369   \def\psput@##1{\pst@getcoor{##1}\leavevmode\psput@cartesian}}
370 \NormalCoor
```

`\pst@angleunit, \degrees, \radians`

`\degrees` sets `\pst@angleunit` to the PostScript code for scaling the angle, including the trailing space.

```
371 \def\degrees{\@ifnextchar[{\@degrees}{\def\pst@angleunit{}}}
372 \def\@degrees[#1]{%
373   \pst@checknum{#1}\pst@tempg
374   \edef\pst@angleunit{360 \pst@tempg div mul }%
375   \ignorespaces}
376 \def\radians{\def\pst@angleunit{57.2956 mul }}
377 \def\pst@angleunit{}
```

15 Special coordinates and angles

This is a tedious but useful.

`\SpecialCoor`

```
378 \def\SpecialCoor{%
379   \def\pst@getcoor##1{\pst@expandafter\special@coor{##1}|\@nil}%
380   \def\pst@getangle##1{\pst@expandafter\special@angle{##1}\@empty}\@nil}%
381   \def\psput@##1{\pst@getcoor{##1}\leavevmode\psput@special}}
```

`\specialcoor`

```

382 \def\special@coor#1|#2|#3\@nil{%
383   \ifx#3|\relax
384     \mixed@coor{#1}{#2}%
385   \else
386     \special@@coor#1;;\@nil
387   \fi}
388 \def\special@@coor#1{%
389   \ifcat#1a\relax
390     \def\next{\node@coor#1}%
391   \else
392     \ifx#1[\relax
393       \def\next{\Node@coor[]}%
394     \else
395       \ifx#1!\relax
396         \def\next{\raw@coor}%
397       \else
398         \def\next{\special@@@coor#1}%
399       \fi
400     \fi
401   \fi
402   \next}
403 \def\special@@@coor#1|#2|#3\@nil{%
404   \ifx#3|\relax
405     \polar@coor{#1}{#2}%
406   \else
407     \cartesian@coor#1,\relax,\@nil
408   \fi}

```

`\mixed@coor`

This allows mixing of coordinate types with `\SpecialCoor`.

```

409 \def\mixed@coor#1#2{%
410   \begingroup
411     \specialcoor@ii#1;;\@nil
412     \let\pst@tempa\pst@coor
413     \specialcoor@ii#2;;\@nil
414     \xdef\pst@tempg{\pst@tempa pop \pst@coor exch pop }%
415   \endgroup
416   \let\pst@coor\pst@tempg}

```

`\polar@coor`

For polar coordinates

```

417 \def\polar@coor#1#2{%
418   \pssetlength\pst@dimg{#1}%
419   \pst@getangle{#2}%
420   \edef\pst@coor{\pst@number\pst@dimg \pst@angle \tx@PtoC}}

```

`\raw@coor`

For raw PostScript.

```

421 \def\raw@coor#1|#2\@nil{%

```

```

422 \edef\pst@coor{%
423   #1 \pst@number\psyunit mul exch \pst@number\psxunit mul exch }}

```

`\node@coor`, `\Node@coor`

These are defined in `pst-node.tex`.

```

424 \def\node@coor#1\@nil{%
425   \@pstrickserr{You must load 'pst-node.tex' to use node coordinates.}\@ehps
426   \def\pst@coor{0 0 }}
427 \def\Node@coor{\node@coor}

```

`\special@angle`

```

428 \def\special@angle#1#2)#3\@nil{%
429   \ifx#1!\relax
430     \edef\pst@angle{#2 \pst@angleunit}%
431   \else
432     \ifx#1(\relax
433       \pst@getcoor{#2}%
434       \edef\pst@angle{\pst@coor exch \tx@Atan}%
435     \else
436       \pst@checknum{#1#2}\pst@angle
437       \edef\pst@angle{\pst@angle \pst@angleunit}%
438     \fi
439   \fi}

```

`\Cartesian`, `\Polar`

These are obsolete.

```

440 \def\Cartesian{%
441   \def\cartesian@coor##1,##2,##3\@nil{%
442     \pssetxlength\pst@dimg{##1}%
443     \pssetylength\pst@dimh{##2}%
444     \edef\pst@coor{\pst@number\pst@dimg \pst@number\pst@dimh}}%
445   \@ifnextchar({\Cartesian@}{})}
446 \def\Cartesian@(#1,#2){%
447   \pssetxlength\psxunit{#1}%
448   \pssetylength\psyunit{#2}%
449   \ignorespaces}
450 \def\Polar{%
451   \def\psput@cartesian{\psput@special}%
452   \def\cartesian@coor##1,##2,##3\@nil{\polar@coor{##1}{##2}}}%

```

16 Basic graphics parameters

`\psset@origin`, `\psk@origin`

```

453 \def\psset@origin#1{%
454   \pst@getcoor{#1}%
455   \edef\psk@origin{\pst@coor \tx@NET }}
456 \def\psk@origin{}

```

```

\psset@swapaxes, \ifpsswapaxes
457 \newif\ifswapaxes
458 \def\psset@swapaxes#1{%
459   \@nameuse{@pst#1}%
460   \if@pst
461     \def\psk@swapaxes{-90 rotate -1 1 scale }%
462   \else
463     \def\psk@swapaxes{}%
464   \fi}
465 \psset@swapaxes{false}

\psset@showpoints, \ifshowpoints
466 \newif\ifshowpoints
467 \def\psset@showpoints#1{\@nameuse{showpoints#1}}
468 \psset@showpoints{false}

\psset@border, \psk@border
469 \let\pst@setrepeatarrowsflag\relax
470 \def\psset@border#1{%
471   \pst@getlength{#1}\psk@border
472   \pst@setrepeatarrowsflag}
473 \psset@border{Opt}

\psset@bordercolor, \psbordercolor
474 \def\psset@bordercolor#1{\pst@getcolor{#1}\psbordercolor}
475 \psset@bordercolor{white}

\psset@doubleline, \ifpsdoubleline
476 \newif\ifpsdoubleline
477 \def\psset@doubleline#1{%
478   \@nameuse{psdoubleline#1}%
479   \pst@setrepeatarrowsflag}
480 \psset@doubleline{false}

\psset@doublesep, \psdoublesep
481 \def\psset@doublesep#1{\def\psdoublesep{#1}}
482 \psset@doublesep{1.25\pslinewidth}

\psset@doublecolor, \psdoublecolor
483 \def\psset@doublecolor#1{\pst@getcolor{#1}\psdoublecolor}
484 \psset@doublecolor{white}

\psset@shadow, \ifpsshadow
485 \newif\ifpsshadow
486 \def\psset@shadow#1{%
487   \@nameuse{psshadow#1}%
488   \pst@setrepeatarrowsflag}
489 \psset@shadow{false}

\psset@shadowsize, \psk@shadowsize

```

```

490 \def\psset@shadowsize#1{\pst@getlength{#1}\psk@shadowsize}
491 \psset@shadowsize{3pt}

\psset@shadowangle, \psk@shadowangle
492 \def\psset@shadowangle#1{\pst@getangle{#1}\psk@shadowangle}
493 \psset@shadowangle{-45}

\psset@shadowcolor, \psshadowcolor
494 \def\psset@shadowcolor#1{\pst@getcolor{#1}\psshadowcolor}
495 \psset@shadowcolor{darkgray}

\pst@setrepeatarrowsflag
496 \def\pst@repeatarrowsflag{\z@}
497 \def\pst@setrepeatarrowsflag{%
498   \edef\pst@repeatarrowsflag{%
499     \ifdim\psk@border\p@>\z@ 1\else\ifpsdoubleline 1\else
500     \ifpsshadow 1\else \z@\fi\fi\fi}}

```

17 Line styles

For each *linestyle style*, the command `\psls@<style>` should be the PostScript code that strokes the path. The style can assume that the PostScript environment has linewidth equal to `\pslinewidth` and color equal to `\pslinecolor`.

`\pst@linetype`

Macros that draw lines should define `\pst@linetype` (not a count register) to be:

1,2,... A closed path whose length should be divided by `\pst@linetype` before fitting a pattern.

0 A line that has nothing at the tips.

-1 A line with an arrow or something at the end.

-2 A line with an arrow or something at the beginning.

-3 A line with an arrow or something at each end.

This information is used by the line styles that draw dashed and dotted lines in order to figure out how to adjust the patterns.

`\psls@none`

```

501 \def\psls@none{}

\psset@linewidth, \pslinewidth
502 \newdimen\pslinewidth
503 \def\psset@linewidth#1{\pssetlength\pslinewidth{#1}}
504 \psset@linewidth{.8pt}

\psset@linecolor, \pslinecolor
505 \def\psset@linecolor#1{\pst@getcolor{#1}\pslinecolor}

```

```

506 \psset@linecolor{black}

    \psls@solid
507 \def\psls@solid{0 setlinecap stroke }

    \psset@dash, \psk@dash

    \psk@dash is set to the PostScript code for the dash pattern (include the trailing space).

508 \def\psset@dash#1{%
509   \pst@expandafter\psset@@dash{#1} * * *\@nil
510   \edef\psk@dash{\pst@number\pst@dimg \pst@number\pst@dimh}}
511 \def\psset@@dash#1 #2 #3\@nil{%
512   \pssetlength\pst@dimg{#1}%
513   \pssetlength\pst@dimh{#2}}
514 \psset@dash{5pt 3pt}

    \psls@dashed
515 \def\psls@dashed{\psk@dash \pst@linetype\space \tx@DashLine}

    \tx@DashLine

Syntax:

    dim1 dim2 linetype DashLine

DashLine adjusts, and then sets, the dash pattern [<dim1 dim2>] so that it fits evenly
onto a path.

516 \pst@def{DashLine}<%
517   % "a" is set to the length of first and last black segment, as fraction of
518   % usual black segment.
519   dup 0 gt
520   { /a .5 def \tx@PathLength exch div }
521   { pop /a 1 def \tx@PathLength }
522   ifelse
523   /b ED           % Pattern should fit evenly in b
524   /x ED           % Length of white segment.
525   /y ED           % Length of black segment.
526   /z y x add def % Total length of dash pattern.
527   % If pattern is repeated n times, total length is (nz + 2(a-.5)y).
528   % Set length to b, solve for n, round, and leave on stack:
529   %   n = round((b - 2(a-.5)y)/z)
530   b a .5 sub 2 mul y mul sub z \tx@Div round
531   % Adjust x and y by factor k so that
532   %   (n(kz) + 2(a-.5)(ky)) = b.
533   % Solve for k and leave two copies on stack:
534   %   k = b/(nz + 2(a-.5)y)
535   z mul a .5 sub 2 mul y mul add b exch \tx@Div dup
536   % Scale x and y, set dash, and stroke:
537   y mul /y ED x mul /x ED
538   % Make sure both x and y aren't zero:
539   x 0 eq y 0 eq and { /x 1 def /y 1 def } if
540   [ y x ] 1 a sub y mul setdash stroke>

```


dotsep

```
541 \def\psset@dotsep#1{\pst@getlength{#1}\psk@dotsep}  
542 \psset@dotsep{3pt}
```

\psls@dotted

```
543 \def\psls@dotted{\psk@dotsep \pst@linetype\space \tx@DotLine}%
```

\tx@DotLine

Syntax:

dim linetype DotLine

DotLine adjusts, and then sets, the dash pattern to produce a dotted line with distance *dim* between dots so that it fits evenly onto a path. Dots are produced by setting dash pattern with length of white segment equal to distance from center of dot to center of dot, length of black segment equal to 0, and linecap equal to 1.

```
544 \pst@def{DotLine}<%  
545 /b \tx@PathLength def % Path length.  
546 /a ED % \pst@linetype.  
547 /z ED % dotsep.  
548 /y CLW def % linewidth (dot diameter).  
549 /z y z add def % Total length of dash pattern.  
550 % Set b to adjusted path length that pattern should be multiple of:  
551 a 0 gt  
552 % If closed, as many dots as spaces.  
553 { /b b a div def }  
554 { a 0 eq  
555 % If open with no arrows, one more dot than space.  
556 { /b b y sub def }  
557 % If open one arrow, as many dots as spaces (do nothing)  
558 % If open two arrows, one more space than dot.  
559 { a -3 eq { /b b y add def } if }  
560 ifelse }  
561 ifelse  
562 % Let n be number of times pattern is repeated:  
563 % n = round(b/z)  
564 % Adjust length of pattern so that it fits evenly in b:  
565 % z = b/n = b/(round(b/z))  
566 % z is length of white segment. Length of black segment is 0.  
567 [ 0 b b z \tx@Div round \tx@Div dup 0 le { pop 1 } if ]  
568 a 0 gt % setting dash pattern.  
569 % Set offset to 0 if path is closed]  
570 { 0 }  
571 % Set offset to -(y/2) if open curve begins with arrow, (y/2) otherwise:  
572 { y 2 div a -2 gt { neg } if }  
573 ifelse  
574 % Setting linecap to 1 produces the dots.  
575 setdash 1 setlinecap stroke>
```

\psset@linestyle

```
576 \def\psset@linestyle#1{%
```

```

577 \ifundefined{psls@#1}%
578   {\@pstrickserr{Line style '#1' not defined}\@eha}%
579   {\edef\pslinestyle{#1}}}
580 \psset@linestyle{solid}

```

18 Fill styles

For each fillstyle *style*, the command `\psfs@<style>` should be the PostScript code that fills the region. The style should not assume anything about the PostScript environment's linewidth or color.

```

\psfs@none
581 \def\psfs@none{}

\psset@fillcolor, \psfillcolor
582 \def\psset@fillcolor#1{\pst@getcolor{#1}\psfillcolor}
583 \psset@fillcolor{white}

\psfs@solid
584 \def\psfs@solid{\pst@usecolor\psfillcolor fill }

hatchwidth
585 \def\psset@hatchwidth#1{\pst@getlength{#1}\psk@hatchwidth}
586 \psset@hatchwidth{.8pt}

hatchsep
587 \def\psset@hatchsep#1{\pst@getlength{#1}\psk@hatchsep}
588 \psset@hatchsep{4pt}

hatchcolor
589 \def\psset@hatchcolor#1{\pst@getcolor{#1}\pshatchcolor}
590 \psset@hatchcolor{black}

hatchangle
591 \def\psset@hatchangle#1{\pst@getangle{#1}\psk@hatchangle}
592 \psset@hatchangle{45}

\psfs@hlines
593 \def\psfs@hlines{%
594   \psk@hatchangle rotate
595   \psk@hatchwidth SLW
596   \pst@usecolor\pshatchcolor
597   \psk@hatchsep \tx@LineFill}
598 \@namedef{psfs@hlines*}{gsave \psfs@solid grestore \psfs@hlines}

\tx@LineFill
599 \pst@def{LineFill}<%
600   abs CLW add /a ED           % hatchsep
601   gsave

```

```

602 clip
603 pathbbox %leave llx, lly, urx, ury on stack
604 a \tx@Div ceiling /y2 ED % Number of top line to be drawn.
605 /x2 ED
606 a \tx@Div floor /y1 ED % Number of bottom line to be drawn
607 /x1 ED
608 /n y2 y1 sub 1 add cvi def % Number of lines.
609 /y1 a y1 mul def % y-coordinate of bottom line.
610 newpath 2 setlinecap
611 n
612 { currentstrokeadjust ==
613 x1 y1 moveto
614 x2 y1 L
615 stroke
616 /y1 y1 a add def }
617 repeat
618 grestore>
619 \pst@def{LineFill}<%
620 abs CLW add /a ED % hatchsep
621 gsave
622 clip
623 pathbbox %leave llx, lly, urx, ury on stack
624 a \tx@Div ceiling /y2 ED % Number of top line to be drawn.
625 /x2 ED
626 a \tx@Div floor /y1 ED % Number of bottom line to be drawn
627 /x1 ED
628 /n y2 y1 sub 1 add cvi def % Number of lines.
629 /y1 a y1 mul def % y-coordinate of bottom line.
630 newpath 2 setlinecap
631 systemdict /currentstrokeadjust known % Level 2
632 { currentstrokeadjust }
633 { false }
634 ifelse
635 { /t { } def }
636 { /t {
637 transform
638 0.25 sub round 0.25 add exch
639 0.25 sub round 0.25 add exch
640 itransform
641 } bind def }
642 ifelse
643 n {
644 x1 y1 t moveto
645 x2 y1 t L
646 stroke
647 /y1 y1 a add def
648 } repeat
649 grestore>

\psfs@vlines
650 \def\psfs@vlines{%
651 90 rotate
652 \psfs@hlines}
653 \@namedef{psfs@vlines*}{gsave \psfs@solid grestore \psfs@vlines}

```

```

\psfs@crosshatch
654 \def\psfs@crosshatch{gsave \psfs@hlines grestore \psfs@vlines}
655 \@namedef{psfs@crosshatch*}{%
656   gsave \psfs@solid grestore
657   gsave \psfs@hlines grestore
658   \psfs@vlines}

```

fillstyle

```

659 \def\psset@fillstyle#1{%
660   \@ifundefined{psfs@#1}%
661     {\@pstrickserr{Undefined fill style: '#1'}\@eha}%
662     {\edef\psfillstyle{#1}}
663 \psset@fillstyle{none}

```

19 Arrowheads and t-bars

It would be nice to use a font, with hinting.

```
\psset@arrows, \psk@arrowA, \psk@arrowB
```

\if@pst is used as a flag for errors.

```

664 \def\psset@arrows#1{%
665   \begingroup
666     \pst@activearrows
667     \xdef\pst@tempg{#1}%
668   \endgroup
669   \expandafter\psset@@arrows\pst@tempg\@empty-\@empty\@nil
670   \if@pst\else
671     \@pstrickserr{Bad arrows specification: #1}\@ehpa
672   \fi}
673 \def\psset@@arrows#1-#2\@empty#3\@nil{%
674   \@psttrue
675   \def\next##1,#1-##2,##3\@nil{\def\pst@tempg{##2}}%
676   \expandafter\next\pst@arrowtable,#1-#1,\@nil
677   \@ifundefined{psas@\pst@tempg}%
678     {\@pstfalse\def\psk@arrowA{}}%
679     {\let\psk@arrowA\pst@tempg}%
680   \@ifundefined{psas@#2}%
681     {\@pstfalse\def\psk@arrowB{}}%
682     {\def\psk@arrowB{#2}}
683 \def\psk@arrowA{}
684 \def\psk@arrowB{}

```

```
\pst@arrowtable
```

This is a translator for arrowA. Add to it with \edef, as in

```

\edef\pst@arrowtable{\pst@arrowtable,*o-o*}
685 \def\pst@arrowtable{,<->,<<->>,>-<,>>-<<,( - ),[-]}

```

`\pst@activearrows`

This redefines certain characters in case they are active, before expanding the `arrows` argument. Add to it with `\expandafter`, as in

```
\begingroup
  \catcode'\:=13
  \expandafter\gdef\expandafter\pst@activearrows
  \expandafter{\def:{\string:}}
\endgroup

686 \begingroup
687   \catcode'\<=13
688   \catcode'\>=13
689   \catcode'\|=13
690   \gdef\pst@activearrows{\def<{\string<}\def>{\string>}\def|{\string|}}
691 \endgroup
```

`BeginArrow`, `EndArrow`

For each arrow *arrow*, `\psas@<arrow>` should be PostScript code so that

```
y2 x2 y1 x1 BeginArrow \psk@arrowscale \psas@arrow EndArrow
```

- Draws an arrow with the tip at $x1\ y1$, and
- Leaves on the stack $y2\ x2\ x1'\ y1'$, where $x1'\ y1'$ is the position that a connecting line should start from.

`BeginArrow` sets up an environment so that `\psas@<arrow>` only has to draw an arrow pointing down and with the tip at 0 0 and , and leave the current point where a connecting line should start from. `EndArrow` then restores the original environment and translates the current point into the original coordinate system.

A special dictionary `ADict` is used with arrows so that scratch variables will not conflict. The matrix is saved as `@mtrx` to indicate that the arrow procedures should not change this. The same is true for `@x1`, `@y1`, `@x2`, `@y2` and `@angle`, which are used by a patch of `BeginArrow` and `EndArrow` that is required for some versions of Sun's NewsPrint (see `read-me.pst`).

```
692 \pst@def{BeginArrow}<%
693   ADict begin
694   /@mtrx CM def
695   gsave
696     2 copy T
697     2 index sub neg exch 3 index sub exch \tx@Atan
698     rotate
699     newpath>
700 \pst@def{EndArrow}<@mtrx setmatrix CP grestore end>
```

`arrowscale`

```
701 \def\psset@arrowscale#1{\pst@getscale{#1}\psk@arrowscale}
702 \psset@arrowscale{1}
```

```

\psset@arrowsize, \psk@arrowsize
703 \def\psset@arrowsize#1{%
704   \pst@expandafter\pst@getdimnum{#1} {} {} {} \@nil
705   \edef\psk@arrowsize{\pst@number\pst@ding \pst@tempg}}
706 \psset@arrowsize{2pt 3}

\psset@arrowlength, \psk@arrowlength
707 \def\psset@arrowlength#1{\pst@checknum{#1}\psk@arrowlength}
708 \psset@arrowlength{1.4}

\psset@arrowinset, \psk@arrowinset
709 \def\psset@arrowinset#1{\pst@checknum{#1}\psk@arrowinset}%
710 \psset@arrowinset{.4}

```

`\tx@Arrow`

Syntax:

boolean `\psk@arrowinset` `\psk@arrowlength` `\psk@arrowsize` `Arrow`

boolean is true for reverse arrows and false for normal arrows.

```

711 \pst@def{Arrow}<%
712   CLW mul add dup           % width
713   2 div /w ED              % Half width
714   mul dup /h ED           % Height
715   mul /a ED                % Inset
716   { 0 h T 1 -1 scale } if % For reverse arrows
717   w neg h moveto
718   0 0 L
719   w h L
720   w neg a neg rlineto
721   gsave fill grestore>

\psas@>
722 \@namedef{psas@>}{%
723   false \psk@arrowinset \psk@arrowlength \psk@arrowsize \tx@Arrow}

\psas@>>
724 \@namedef{psas@>>}{%
725   false \psk@arrowinset \psk@arrowlength \psk@arrowsize \tx@Arrow
726   0 h T
727   gsave
728   newpath
729   false \psk@arrowinset \psk@arrowlength \psk@arrowsize \tx@Arrow
730   CP
731   grestore
732   CP newpath moveto
733   2 copy
734   L
735   stroke
736   moveto}

```

```

\psas@<
737 \@namedef{psas@<}{%
738   true \psk@arrowinset \psk@arrowlength \psk@arrowsize \tx@Arrow}

\psas@<<
739 \@namedef{psas@<<}{%
740   true \psk@arrowinset \psk@arrowlength \psk@arrowsize \tx@Arrow
741   CP newpath moveto 0 a neg L stroke 0 h neg T
742   false \psk@arrowinset \psk@arrowlength \psk@arrowsize \tx@Arrow}

\psset@tbarsize, \psk@tbarsize
743 \def\psset@tbarsize#1{%
744   \pst@expandafter\pst@getdimnum{#1} {} {} {} \@nil
745   \edef\psk@tbarsize{\pst@number\pst@ding \pst@tempg}}
746 \psset@tbarsize{2pt 5}

Tbar

Syntax

\psk@tbarsize Tbar

747 \pst@def{Tbar}<%
748   CLW mul add /z ED           % width
749   z -2 div CLW 2 div moveto
750   z 0 rlineto
751   stroke
752   0 CLW moveto>

\psas@|
753 \@namedef{psas@|}{\psk@tbarsize \tx@Tbar}

\psas@|*
754 \@namedef{psas@|*}{0 CLW -2 div T \psk@tbarsize \tx@Tbar}

\psset@bracketlength, \psk@bracketlength
755 \def\psset@bracketlength#1{\pst@checknum{#1}\psk@bracketlength}
756 \psset@bracketlength{.15}

\tx@Bracket, \tx@@Bracket

Syntax

\psk@bracketlength \psk@tbarsize Bracket

757 \pst@def{Bracket}<%
758   CLW mul add dup
759   CLW sub 2 div /x ED       % adjusted half width
760   mul CLW add /y ED        % y-position of height
761   /z CLW 2 div def
762   x neg y moveto

```

```

763 x neg CLW 2 div L
764 x CLW 2 div L
765 x y L
766 stroke
767 0 CLW moveto>

```

`\psas@]`

```

768 \@namedef{psas@]}\psk@bracketlength \psk@tbarsize \tx@Bracket}

```

`\psset@rbracketlength, \psk@rbracketlength`

```

769 \def\psset@rbracketlength#1{\pst@checknum{#1}\psk@rbracketlength}
770 \psset@rbracketlength{.15}

```

`RoundBracket`

Syntax

`\psk@bracketlength \psk@tbarsize RoundBracket`

```

771 \pst@def{RoundBracket}<%
772 CLW mul add dup
773 2 div /x ED % half width
774 mul /y ED % height
775 /mtrx CM def
776 0 CLW 2 div T
777 x y mul 0 ne { x y scale } if
778 1 1 moveto
779 .85 .5 .35 0 0 0 curveto
780 -.35 0 -.85 .5 -1 1 curveto
781 mtrx setmatrix
782 stroke
783 0 CLW moveto>

```

`\psas@(`

```

784 \@namedef{psas@()}\psk@rbracketlength \psk@tbarsize \tx@RoundBracket}

```

`\psas@c, \psas@cc, \psas@C`

This is not going to be used frequently, and so we don't bother defining a PostScript procedure in the header.

```

785 \def\psas@c{1 \psas@@c}
786 \def\psas@cc{0 CLW 2 div T 1 \psas@@c}
787 \def\psas@C{2 \psas@@c}
788 \def\psas@@c{%
789 setlinecap
790 0 0 moveto
791 0 CLW 2 div L
792 stroke
793 0 0 moveto}

```



```

\psas@
794 \def\psas@{}
795 \psset@arrows{-}

```

20 Graphics objects: processing arguments

`\pst@par`, `\addto@par`, `\use@par`

Graphics objects accumulate *parameter=value* pairs in the command sequence `\pst@par`. They use `\addto@par` to add to `\pst@par`, and `\use@par` to make the parameter changes effective.

```

796 \def\pst@par{}
797 \def\addto@par#1{%
798   \ifx\pst@par\empty
799     \def\pst@par{#1}%
800   \else
801     \expandafter\def\expandafter\pst@par\expandafter{\pst@par,#1}%
802   \fi}
803 \def\use@par{%
804   \ifx\pst@par\empty\else
805     \expandafter\@psset\pst@par,\@nil
806   \def\pst@par{}%
807   \fi}

```

`\pst@object`

Any macro, such as `\psline`, that uses graphics parameters should begin as follows:

```

\def\psline{\def\pst@par{}\pst@object{psline}}
\def\psline@i{ ... }

```

`\pst@object` checks for the optional [`<par>=<value>,...`] argument, adds key-value pairs to `\pst@par` if found, skips spaces, and then invokes `\psline@i`.

```

808 \def\pst@object#1{%
809   \pst@ifstar{\@ifnextchar[{\pst@@object{#1}}{\@nameuse{#1@i}}]}
810 \def\pst@@object#1[#2]{%
811   \addto@par{#2}\@ifnextchar+{\@nameuse{#1@i}}{\@nameuse{#1@i}}}

```

`\newpsobject`

For example,

```

\newpsobject{dottedline}{psline}{linestyle=dotted}

```

has the following effect:

```

\def\dottedline{%
  \def\pst@par{linestyle=dotted}\pst@object{psline}}

```

and thus `\dottedline` is just like `\psline`, except that the default value of `linestyle` is changed to `dotted`.

```

812 \def\newpsobject#1#2#3{%
813   \@ifundefined{#2#i}%
814     {\@pstrickserr{Graphics object ‘#2’ not defined}\@eha}%
815     {\@namedef{#1}{\def\pst@par{#3}\pst@object{#2}}}\ignorespaces}

```

`\pst@getarrows`

`\pst@getarrows{foo}` checks for an optional argument containing arrows, and then invokes `foo`. The arrows argument must be followed by `(`.

```

816 \def\pst@getarrows#1{\@ifnextchar({#1}{\pst@getarrows{#1}}}
817 \def\pst@@getarrows#1#2{\addto@par{arrows=#2}#1}

```

21 Graphics objects: Basics T_EX macros

Each graphics object should use one of the following:

| | |
|--|--------------------------|
| <code>\begin@OpenObj ... \end@OpenObj</code> | Open curves with arrows. |
| <code>\begin@AltOpenObj ... \end@AltOpenObj</code> | Open curves w/o arrows. |
| <code>\begin@ClosedObj ... \end@ClosedObj</code> | Closed curves. |
| <code>\begin@SpecialObj ... \end@SpecialObj</code> | Other. |

This makes it possible for `\pscustom` to work by redefining these.

`\begin@ClosedObj, \end@ClosedObj`

```

818 \def\begin@ClosedObj{%
819   \leavevmode
820   \pst@killglue
821   \begingroup
822     \use@par
823     \solid@star
824     \ifpsdoubleline \pst@setdoublesep \fi
825     \init@pscode}
826 \def\end@ClosedObj{%
827   \ifpsshadow \pst@closedshadow \fi
828   \ifdim\psk@border\p@>\z@ \pst@addborder \fi
829   \pst@fill
830   \pst@stroke
831   \ifpsdoubleline \pst@doublestroke \fi
832   \ifshowpoints
833     \addto@pscode{Points aload length 2 div cvi /N ED \psdots@iii}%
834     \fi
835   \use@pscode
836 \endgroup
837 \ignorespaces}

```

`\begin@OpenObj, \begin@AltOpenObj, \end@OpenObj`

```

838 \def\begin@OpenObj{%
839   \begin@ClosedObj
840   \let\pst@linetype\pst@arrowtype
841   \pst@addarrowdef}

```

```

842 \def\begin@AltOpenObj{%
843   \begin@ClosedObj
844     \def\pst@repeatarrowsflag{\z@}%
845     \def\pst@linetype{0}}
846 \def\end@OpenObj{%
847   \ifpsshadow \pst@openshadow \fi
848   \ifdim\psk@border\p@>\z@ \pst@addborder \fi
849   \pst@fill
850   \pst@stroke
851   \ifpsdoubleline \pst@doublestroke \fi
852   \ifnum\pst@repeatarrowsflag>\z@ \pst@repeatarrows \fi
853   \ifshowpoints \pst@OpenShowPoints \fi
854   \use@pscode
855 \endgroup
856 \ignorespaces}

```

`\begin@SpecialObj, \end@SpecialObj`

```

857 \def\begin@SpecialObj{%
858   \leavevmode
859   \pst@killglue
860   \begingroup
861     \use@par
862     \init@pscode}
863 \def\end@SpecialObj{%
864   \use@pscode
865   \endgroup
866   \ignorespaces}

```

`\init@pscode, \addto@pscode, \use@pscode`

Graphics objects are built up by adding PostScript code to `\pst@code` with `\addto@pscode`. `\use@pscode` then adds leading and trailing PostScript code, and (normally) inserts it in a `\special` (it also most empty `\pst@code`). Hacks like `\psclip`, `\multips` and `\pstextpath` work by redefining `\use@pscode`. These hacks use `\use@pscode` themselves when appropriate, making limited nesting of these hacks is possible. `\PSTtoEPS` works by redefining `\addto@pscode` so that it writes to a file. All this was carefully designed so that these hacks would work. Watch out!

```

867 \def\pst@code{}%
868 \def\init@pscode{%
869   \addto@pscode{%
870     \pst@number\pslinewidth SLW
871     \pst@usecolor\pslinecolor}}
872 \def\addto@pscode#1{\xdef\pst@code{\pst@code#1\space}}
873 \def\use@pscode{%
874   \pstverb{%
875     \pst@dict
876     \tx@STP
877     newpath
878     \psk@origin
879     \psk@swapaxes
880     \pst@code
881     end}%
882   \gdef\pst@code{}}

```

```

\pst@killglue
883 \def\KillGlue{%
884   \def\pst@killglue{\unskip\ifdim\lastskip>\z@\expandafter\pst@killglue\fi}}
885 \def\DontKillGlue{\let\pst@killglue\relax}
886 \DontKillGlue

```

`\solid@star`

The optional `*` is typically used to make a solid option. This means that `linestyle` is set to `none`, `linewidth` is set to 0, and `fillcolor` is set to `linecolor`.

```

887 \def\solid@star{%
888   \if@star
889     \pslinewidth=\z@
890     \psdoublelinefalse
891     \def\pslinestyle{none}%
892     \def\psfillstyle{solid}%
893     \let\psfillcolor\pslinecolor
894   \fi}

```

`\pst@setdoublesep`

```

895 \def\pst@setdoublesep{%
896   \pst@getlength\psdoublesep\psdoublesep
897   \pslinewidth=2\pslinewidth
898   \advance\pslinewidth\psdoublesep\p@
899   \let\pst@setdoublesep\relax}

```

`\tx@Shadow`

Syntax:

`x y Shadow`

translates current path by *x y*.

```

900 \pst@def{Shadow}<%
901   [
902     { /moveto load }
903     { /lineto load }
904     { /curveto load }
905     { /closepath load }
906     pathforall
907   ]
908   cvx
909   newpath
910   3 1 roll
911   T
912   exec>

```

`\pst@closedshadow`

```

913 \def\pst@closedshadow{%
914   \addto@pscode{%
915     gsave

```

```

916 \psk@shadowsize \psk@shadowangle \tx@PtoC
917 \tx@Shadow
918 \pst@usecolor\psshadowcolor
919 gsave fill grestore
920 stroke
921 grestore
922 gsave
923 \pst@usecolor\psfillcolor
924 gsave fill grestore
925 stroke
926 grestore}}

```

\pst@openshadow

```

927 \def\pst@openshadow{%
928 \addto@pscode{%
929 gsave
930 \psk@shadowsize \psk@shadowangle \tx@PtoC
931 \tx@Shadow
932 \pst@usecolor\psshadowcolor
933 \ifx\psfillstyle\@none\else
934 gsave fill grestore
935 \fi
936 stroke}%
937 \pst@repeatarrows
938 \addto@pscode{grestore}
939 \ifx\psfillstyle\@none\else
940 \addto@pscode{%
941 gsave
942 \pst@usecolor\psfillcolor
943 gsave fill grestore
944 stroke
945 grestore}
946 \fi}

```

\pst@addborder

```

947 \def\pst@addborder{%
948 \addto@pscode{%
949 gsave
950 \psk@border 2 mul
951 CLW add SLW
952 \pst@usecolor\psbordercolor
953 stroke
954 grestore}}

```

\pst@stroke

```

955 \def\pst@stroke{%
956 \ifx\pslinestyle\@none\else
957 \addto@pscode{%
958 gsave
959 \pst@number\pslinewidth SLW
960 \pst@usecolor\pslinecolor
961 \@nameuse{psls@\pslinestyle}
962 grestore}%

```

```

963 \fi}

\pst@fill
964 \def\pst@fill{%
965 \ifx\psfillstyle\@none\else
966 \addto@pscode{gsave \@nameuse{psfs@\psfillstyle} grestore}%
967 \fi}

\pst@doublestroke
968 \def\pst@doublestroke{%
969 \addto@pscode{%
970 gsave
971 \psdoublesep SLW
972 \pst@usecolor\psdoublecolor
973 stroke
974 grestore}}

\pst@arrowtype
975 \def\pst@arrowtype{%
976 \ifx\psk@arrowB\@empty 0 \else -2 \fi
977 \ifx\psk@arrowA\@empty 0 \else -1 \fi
978 add}

\pst@addarrowdef, \pst@arrowdef, \pst@arrowtype

ArrowA takes two coordinates from the stack, draws the arrow with the tip at the top
coordinate, leaves the second coordinate on the stack and leaves the current point where
a line should join.

ArrowB takes two coordinates from the stack, draws the arrow with the tip at the top
coordinate, and leaves both coordinates on the stack, without changing the graphics
state.

This particular definition of ArrowA is important for \pscustom.

ArrowA and ArrowB might also save the arrow coordinates, because sometimes it is
necessary to redraw the arrows (see \pst@setrepeatarrowsflag).

979 \def\pst@addarrowdef{%
980 \addto@pscode{%
981 /ArrowA {
982 \ifx\psk@arrowA\@empty
983 \pst@oplineto
984 \else
985 \pst@arrowdef{A}
986 moveto
987 \fi
988 } def
989 /ArrowB {
990 \ifx\psk@arrowB\@empty \else \pst@arrowdef{B} \fi
991 } def}}
992 \def\pst@arrowdef#1{%
993 \ifnum\pst@repeatarrowsflag>\z@
994 /Arrow#1c [ 6 2 roll ] cvx def Arrow#1c
995 \fi
996 \tx@BeginArrow

```

```

997 \psk@arrowscale
998 \@nameuse{psas@\@nameuse{psk@arrow#1}}
999 \tx@EndArrow}

\pst@repeatarrows
1000 \def\pst@repeatarrows{%
1001 \addto@pscode{%
1002 gsave
1003 \ifx\psk@arrowA\@empty\else
1004 ArrowAc ArrowA pop pop
1005 \fi
1006 \ifx\psk@arrowB\@empty\else
1007 ArrowBc ArrowB pop pop pop pop
1008 \fi
1009 grestore}}

\pst@OpenShowPoints
1010 \def\pst@OpenShowPoints{%
1011 \addto@pscode{%
1012 gsave
1013 \psk@dotsize
1014 \@nameuse{psds@\psk@dotstyle}
1015 /TheDot {
1016 gsave T \psk@dotangle \psk@dotscale Dot grestore
1017 } def
1018 newpath
1019 Points aload length 2 div 2 sub cvi /N ED
1020 N 0 ge
1021 { \ifx\psk@arrowA\@empty
1022 TheDot
1023 \else
1024 pop pop
1025 \fi
1026 N { TheDot } repeat
1027 \ifx\psk@arrowB\@empty
1028 TheDot
1029 \else
1030 pop pop
1031 \fi }
1032 { N 2 mul { pop } repeat }
1033 ifelse
1034 grestore}}

```

22 Custom graphics

Graphics objects using `\begin@SpecialObj` cannot be used with `\pscustom`. It is up to the other graphics objects to be compatible with `\pscustom`. This means:

- To use the current point as an additional coordinate, when it exists, the graphics object should insert `\pst@cp`.
- For graphics objects that use `\begin@OpenObj`, the `ArrowA` is defined by `\pscustom` to connect the top coordinate with the current point by a line, if

there is a current point. Other graphics objects should use `\pst@oplineto` as a substitute for `moveto` if they wish to connect a coordinate to the current point if it exists.

Closed graphics objects are not under an obligation to anything particularly sensible the current point exists.

`\pscustom`

The main graphics object modifies `\begin@OpenObj` and `\end@OpenObj` so that the open curves extend the current path.

```

1035 \def\pscustom{\def\pst@par{}\pst@object{pscustom}}
1036 \long\def\pscustom@i#1{%
1037   \begin@SpecialObj
1038     \solid@star
1039     \let\pst@ifcustom\iftrue
1040     \let\begin@ClosedObj\begin@CustomObj
1041     \let\end@ClosedObj\endgroup
1042     \def\begin@OpenObj{\begin@CustomObj\pst@addarrowdef}%
1043     \let\end@OpenObj\endgroup
1044     \let\begin@AltOpenObj\begin@CustomObj
1045     \def\begin@SpecialObj{%
1046       \begingroup
1047       \pst@misplaced{special graphics object}%
1048       \def\addto@pscode####1{}
1049       \let\end@SpecialObj\endgroup}%
1050     \def\@multips(##1)(##2)##3##4{\pst@misplaced\multips}%
1051     \def\psclip##1{\pst@misplaced\psclip}%
1052     \def\pst@repeatarrowsflag{\z0}%
1053     \let\pst@setrepeatarrowsflag\relax
1054     \showpointsfalse
1055     \let\showpointstrue\relax
1056     \def\pst@linetype{\pslinetype}%
1057     \let\psset@liftpen\psset@cliftpen
1058     \psset@liftpen{\z0}%
1059     \def\pst@cp{/currentpoint load stopped pop }%
1060     \def\pst@oplineto{/lineto load stopped { moveto } if }%
1061     \def\pst@optcp##1##2{%
1062       \ifnum##1=\z0\def##2{/currentpoint load stopped { 0 0 } if }\fi}%
1063     \let\caddto@pscode\addto@pscode
1064     \def\cuse@par##1{\use@par##1}%
1065     \the\pst@customdefs
1066     \setbox\pst@hbox=\hbox{#1}%
1067     \pst@fill
1068     \pst@stroke
1069   \end@SpecialObj}

\begin@CustomObj, \end@CustomObj

1070 \def\begin@CustomObj{%
1071   \begingroup
1072   \use@par
1073   \addto@pscode{%

```



```

1074      \pst@number\pslinewidth SLW
1075      \pst@usecolor\pslinecolor}}

      \psset@liftpen, \pst@cp, \pst@oplineto, \pst@optcp

1076 \def\pst@oplineto{moveto }
1077 \def\pst@cp{}
1078 \def\pst@optcp#1#2{}
1079 \def\psset@liftpen#1{}
1080 \def\psset@@liftpen#1{%
1081   \ifcase#1\relax
1082     \def\psk@liftpen{\z@}%
1083     \def\pst@cp{/currentpoint load stopped pop }%
1084     \def\pst@oplineto{/lineto load stopped { moveto } if }%
1085   \or
1086     \def\psk@liftpen{1}%
1087     \def\pst@cp{}%
1088     \def\pst@oplineto{/lineto load stopped { moveto } if }%
1089   \or
1090     \def\psk@liftpen{2}%
1091     \def\pst@cp{}%
1092     \def\pst@oplineto{moveto }%
1093   \fi}
1094 \psset@liftpen{0}
1095 \def\psk@liftpen{-1}

```

\psset@linetype, \pslinetype

```

1096 \def\psset@linetype#1{%
1097   \pst@getint{#1}\pslinetype
1098   \ifnum\pst@ding<-3
1099     \@pstrickserr{linetype must be greater than -3}\@ehpa
1100     \def\pslinetype{0}%
1101   \fi}
1102 \psset@linetype{0}

```

\caddto@pscode

Commands that should only occur in \pscustom should use this. Obsolete?

```

1103 \def\caddto@pscode#1{%
1104   \@pstrickserr{Command can only be used in \string\pscustom}\@ehpa}
1105 \let\cuse@par\caddto@pscode

```

\tx@MSave, \tx@MRestore

It doesn't seem worth adding these to the header file.

```

1106 \def\tx@MSave{%
1107   /msavemtrx
1108   [ tx@Dict /msavemtrx known { msavemtrx aload pop } if CM ]
1109   def }
1110 \def\tx@MRestore{%
1111   tx@Dict /msavemtrx known { length 0 gt } { false } ifelse
1112   { /msavematrix [ msavematrix aload pop setmatrix ] def }
1113   if }

```

```

\psmove, \pscclosepath, \psgroup
1114 \newtoks\pst@customdefs
1115 \pst@customdefs{%
1116   \def\newpath{\addto@pscode{newpath}}%
1117   \def\moveto(#1){\pst@@getcoor{#1}\addto@pscode{\pst@coor moveto}}%
1118   \def\closepath{\addto@pscode{closepath}}%
1119   \def\gsave{\begingroup\addto@pscode{gsave}}%
1120   \def\grestore{\endgroup\addto@pscode{grestore}}%
1121   \def\translate(#1){\pst@@getcoor{#1}\addto@pscode{\pst@coor moveto}}%
1122   \def\rotate#1{\pst@@getangle{#1}\addto@pscode{\pst@angle rotate}}%
1123   \def\scale#1{\pst@getscale{#1}\pst@tempg\addto@pscode{\pst@tempg}}%
1124   \def\msave{\addto@pscode{\tx@MSave}}%
1125   \def\mrestore{\addto@pscode{\tx@MRestore}}%
1126   \def\swapaxes{\addto@pscode{-90 rotate -1 1 scale}}%
1127   \def\stroke{\def\pst@par{}\pst@object{stroke}}%
1128   \def\fill{\def\pst@par{}\pst@object{fill}}%
1129   \def\openshadow{\def\pst@par{}\pst@object{openshadow}}%
1130   \def\closedshadow{\def\pst@par{}\pst@object{closedshadow}}%
1131   \def\movepath(#1){\pst@@getcoor{#1}\addto@pscode{\pst@coor tx@Shadow}}%
1132   \def\lineto{\pst@onecoor{lineto}}%
1133   \def\rlineto{\pst@onecoor{rlineto}}%
1134   \def\curveto{\pst@threecoor{curveto}}%
1135   \def\rcurveto{\pst@threecoor{rcurveto}}%
1136   \def\code#1{\addto@pscode{#1}}%
1137   \def\coor(#1){\pst@@getcoor{#1}\addto@pscode{\pst@coor@ifnextchar({\coor}{})}}%
1138   \def\rcoor{\pst@getcoors{}}%
1139   \def\dim#1{\pssetlength\pst@dimg{#1}\addto@pscode{\pst@number\pst@dimg}}%
1140   \def\setcolor#1{%
1141     \@ifundefined{color@#1}{}{\addto@pscode{\use@color{#1}}}%
1142   \def\arrows#1{\psset@arrows{#1}\pst@addarrowdef}}%
1143   \let\file\pst@rawfile
1144 } % END \pst@customdefs
1145 \def\closedshadow@i{\cuse@par\pst@closedshadow}
1146 \def\openshadow@i{\cuse@par\pst@openshadow}
1147 \def\stroke@i{\cuse@par\pst@stroke}%
1148 \def\fill@i{\cuse@par\pst@fill}%
1149 \def\pst@onecoor#1(#2){%
1150   \pst@@getcoor{#2}%
1151   \addto@pscode{\pst@coor #1}}
1152 \def\pst@threecoor#1(#2)#3(#4)#5(#6){%
1153   \begingroup
1154     \pst@getcoor{#2}\pst@tempa
1155     \pst@getcoor{#4}\pst@tempb
1156     \pst@getcoor{#6}\pst@tempc
1157     \addto@pscode{\pst@tempa \pst@tempb \pst@tempc #1}%
1158   \endgroup}

\psrawfile, \pst@rawfile
1159 \def\pst@rawfile#1{%
1160   \begingroup
1161     \def\do##1{\catcode'##1=12\relax}"
1162     \dospecials
1163     \catcode'\%=14

```

```

1164     \pst@rawfile{#1}%
1165   \endgroup}
1166 \def\pst@rawfile#1{%
1167   \immediate\openin1 #1
1168   \ifeof1
1169     \@pstrickserr{File ‘#1’ not found}\@ehpa
1170   \else
1171     \immediate\read1 to \pst@tempg
1172     \loop
1173       \ifeof1 \@pstfalse\else\@psttrue\fi
1174     \if@pst
1175       \addto@pscode\pst@tempg
1176       \immediate\read1 to \pst@tempg
1177     \repeat
1178   \fi
1179   \immediate\closein1\relax}

```

23 Graphics objects: Basic PostScript macros

SD

```

1180 \pst@def{SD}<%
1181   0 360 arc fill>
1182 \pst@def{SQ}<%
1183   /r ED
1184   r r moveto
1185   r r neg L
1186   r neg r neg L
1187   r neg r L
1188   fill>
1189 \pst@def{ST}<%
1190   /y ED /x ED
1191   x y moveto
1192   x neg y L
1193   0 x L
1194   fill>
1195 \pst@def{SP}<%
1196   /r ED
1197   gsave
1198     0 r moveto
1199     4 { 72 rotate 0 r L } repeat
1200     fill
1201   grestore>
1202 \@namedef{psds@*}{/Dot { 0 0 DS \tx@SD } def}
1203 \@namedef{psds@o}{%
1204   /r2 DS CLW sub def
1205   /Dot { 0 0 DS \tx@SD \pst@usecolor\psfillcolor 0 0 r2 \tx@SD } def}
1206 \@namedef{psds@square*}{%
1207   /r1 DS .886 mul def
1208   /Dot { r1 \tx@SQ } def}
1209 \@namedef{psds@square}{%
1210   /r1 DS .886 mul def /r2 r1 CLW sub def
1211   /Dot { r1 \tx@SQ \pst@usecolor\psfillcolor r2 \tx@SQ } def}
1212 \@namedef{psds@triangle*}{%

```

```

1213 /y1 DS .778 mul neg def /x1 y1 1.732 mul neg def
1214 /Dot { x1 y1 \tx@ST } def}
1215 \@namedef{psds@triangle}{%
1216 /y1 DS .778 mul neg def /x1 y1 1.732 mul neg def
1217 /y2 y1 CLW add def /x2 y2 1.732 mul neg def
1218 /Dot { x1 y1 \tx@ST \pst@usecolor\psfillcolor x2 y2 \tx@ST } def}
1219 \@namedef{psds@pentagon*}{%
1220 /r1 DS 1.149 mul def
1221 /Dot { r1 \tx@SP } def}
1222 \@namedef{psds@pentagon}{%
1223 DS .93 mul dup 1.236 mul /r1 ED CLW sub 1.236 mul /r2 ED
1224 /Dot { r1 \tx@SP \pst@usecolor\psfillcolor
1225 r2 \tx@SP } def}
1226 \@namedef{psds@+}{%
1227 /DS DS 1.253 mul def
1228 /Dot { DS 0 moveto DS neg 0 L stroke
1229 0 DS moveto 0 DS neg L stroke } def}
1230 \@namedef{psds@|}{%
1231 \psk@tbarsize CLW mul add 2 div /DS ED
1232 /Dot { 0 DS moveto 0 DS neg L stroke } def}

```

dotstyle

```

1233 \def\psset@dotstyle#1{%
1234 \@ifundefined{psds@#1}%
1235 {\@pstrickserr{Dot style '#1' not defined}\@eha}%
1236 {\edef\psk@dotstyle{#1}}}
1237 \psset@dotstyle{*}

```

NArray

Syntax:

array of points NArray *points*

Sets *n* to the number of pairs in the array, and makes sure there is an even number of elements.

```

1238 \pst@def{NArray}<%
1239 aload length 2 div dup
1240 dup cvi eq not { exch pop } if
1241 /n exch cvi def>
1242 \pst@def{NArray}<%
1243 /f ED
1244 counttomark 2 div
1245 dup cvi /n ED
1246 n eq not { exch pop } if
1247 f
1248 { ] aload /Points ED }
1249 { n 2 mul 1 add -1 roll pop }
1250 ifelse>

```

Line

Syntax:

array of points Line -

ArrowA and ArrowB should be defined to draw arrows, and Lineto should be the procedure used to draw the path; either `lineto` or `Arcto`.

```
1251 \pst@def{Line}<%
1252   \tx@NArray
1253   n 0 eq not
1254   { n 1 eq { 0 0 /n 2 def } if
1255     ArrowA
1256     /n n 2 sub def
1257     n { Lineto } repeat
1258     CP 4 2 roll ArrowB L
1259     pop pop }
1260   if>
```

Arcto

Syntax:

x2 y2 x1 y1 Arcto *x2 y2*

`r` should be set to the arc radius. Adds to the path with `arcto`, with the corner at *x1 y1* and going towards *x2 y2*. Works even when the points are equal. For use with `Line` and `Polygon`.

```
1261 \pst@def{Arcto}<%
1262   /a [ 6 -2 roll ] cvx def
1263   a r /arcto load stopped { 5 } { 4 } ifelse { pop } repeat a>
```

Polygon

Syntax:

array of points Line -

`Lineto` should be the procedure used to draw the path; either `lineto` or `Arcto`.

```
1264 \pst@def{CheckClosed}<%
1265   dup n 2 mul 1 sub index eq 2 index n 2 mul 1 add index eq and
1266   { pop pop /n n 1 sub def }
1267   if>
1268 \pst@def{Polygon}<%
1269   \tx@NArray
1270   n 2 eq { 0 0 /n 3 def } if
1271   n 3 lt
1272   { n { pop pop } repeat }
1273   { n 3 gt { \tx@CheckClosed } if
1274     n 2 mul -2 roll /y0 ED /x0 ED
1275     /y1 ED /x1 ED x1 y1
1276     /x1 x0 x1 add 2 div def
1277     /y1 y0 y1 add 2 div def
1278     x1 y1 moveto
1279     /n n 2 sub def
1280     n { Lineto } repeat
1281     x1 y1 x0 y0 6 4 roll
1282     Lineto Lineto pop pop closepath }
1283   ifelse>
```

24 Interpolated curves

This documentation is largely junk.

There one was an alternate algorithm that had the nice property that when the coordinates were scaled, the interpolated curve would scale in the same way. It was also simpler. However, this one gives nicer looking results in most cases.

Two parameters should be defined:

- a Lower values make the curve tighter. (Default: 1)
- b Higher values make the curve tighter where the angle ABC is less than 45 degrees, and loosen the curve elsewhere. (Default: .1)

`ArrowA` and `ArrowB` should be defined as well.

Each two points are connected by a single Bezier curve, using `curveto`. For each point P, let P- and P+ be the control points before and after the point. I.e., If A, B and C are consecutive points, then A and B are connected by the Bezier curve with control points A, A+, B- and B, and B and C are connected with control points B, B+, C- and C.

The interpolation is local, meaning that control points B- and B+ depend only on points A, B and C.

`\tx@CCA`, `\tx@CC`

The first three lines before CCA set $x1 = Ax + l0$, $y1 = Ay + l0$, $l0 = d(A, B)$, $dx0 = Bx - Ax$, and $dy0 = By - Ay$. After CCA, $x = Bx$, $y = By$, $dx1 = Cx - Bx$, $dy2 = Cy - By$, and $l1 = d(B, C)$.

```

1284 \pst@def{CCA}<%
1285   /y ED /x ED 2 copy
1286   y sub /dy1 ED x sub /dx1 ED
1287   /l1 dx1 dy1 \tx@Pyth def>
1288 \pst@def{CCA}<%
1289   /y ED /x ED 2 copy
1290   y sub /dy1 ED x sub /dx1 ED
1291   /l1 dx1 dy1 \tx@Pyth def>
1292 \pst@def{CC}<%
1293   /l0 l1 def
1294   /x1 x dx sub def /y1 y dy sub def
1295   /dx0 dx1 def /dy0 dy1 def
1296   \tx@CCA

```

The task is now to calculate $B-$ and $B+$. We first calculate the slope dx and dy at B . This tangent at B should be perpendicular to the bisection of the angle ABC . Recalling that $dx0$ and $dy0$ “point” from A to B , this tangency thus passes through $B + (dx, dy)$, where (dx, dy) is the average of $dx0, dy0$ and $dx1, dy1$, once these have been normalized to have the same length. If we normalize by dividing each by their length, and then multiplying both by both lengths, we get

$$\begin{aligned} dx &= l1 \times dx0 + l0 \times dx1 \\ dy &= l1 \times dy0 + l0 \times dy1 \end{aligned}$$

```

1297 /dx dx0 l1 c exp mul dx1 l0 c exp mul add def
1298 /dy dy0 l1 c exp mul dy1 l0 c exp mul add def

```

dx and dy give us the direction of the control points $B-$ and $B+$ from B . Now we adjust the distance of these control points. The first component is sine of the angle ABC , so that smaller angles give closer control points. This is raised to the b , so that b controls the extent of this dependency (and can even reverse the relation). Then this amount is multiplied times a , which those adjusts the overall tightness, independently of the angle. Let's call this amount M . This amount is then divided by the length of the vector (dx, dy) , thereby normalizing this vector to unit length, and then, multiplied times the distance between A and B (for calculating $B-$). Thus, $B-$ is distance $Md(A, B)$ from B . $(x2, y2)$ are set to $B-$, thus calculated, and $B+$ is temporarily stored in (dx, dy) .

```

1299 /m dx0 dy0 \tx@Atan dx1 dy1 \tx@Atan sub
1300     2 div cos abs b exp a mul
1301     dx dy \tx@Pyth \tx@Div 2 div def
1302 /x2 x l0 dx mul m mul sub def
1303 /y2 y l0 dy mul m mul sub def
1304 /dx l1 dx mul m mul neg def
1305 /dy l1 dy mul m mul neg def>

```

$\backslash\text{tx@IC}$, $\backslash\text{tx@BOC}$, $\backslash\text{tx@NC}$, $\backslash\text{tx@EOC}$, $\backslash\text{tx@BAC}$, $\backslash\text{tx@NAC}$, $\backslash\text{tx@EAC}$

These are the components of the loops that go through the lists of points that are to be interpolated. These are abbreviations, as follows:

| | |
|-----|-------------------------|
| IC | Initialize Curve |
| BOC | Begin Open Curve |
| NC | Next Curve |
| EOC | End Open Curve |
| BAC | Begin Alternative Curve |
| NAC | Next Alternative Curve |
| EAC | End Alternative Curve |

```

1306 \pst@def{IC}<%
1307 /c c 1 add def
1308 c 0 lt { /c 0 def } { c 3 gt { /c 3 def } if } ifelse
1309 /a a 2 mul 3 div 45 cos b exp div def
1310 \tx@CCA /dx 0 def /dy 0 def>
1311 \pst@def{BOC}<%
1312 \tx@IC \tx@CC x2 y2 x1 y1 ArrowA
1313 CP 4 2 roll x y curveto>
1314 \pst@def{NC}<\tx@CC x1 y1 x2 y2 x y curveto>
1315 \pst@def{EOC}<%
1316 x dx sub y dy sub 4 2 roll ArrowB 2 copy curveto>
1317 \pst@def{BAC}<%
1318 \tx@IC \tx@CC x y moveto \tx@CC
1319 x1 y1 CP ArrowA>
1320 \pst@def{NAC}<x2 y2 x y curveto \tx@CC x1 y1>
1321 \pst@def{EAC}<x2 y2 x y ArrowB curveto pop pop>

```

OpenCurve

Syntax:

array of points OpenCurve

```

1322 \pst@def{OpenCurve}<%
1323   \tx@NArray
1324   n 3 lt
1325   { n { pop pop } repeat }
1326   { \tx@BOC
1327     /n n 3 sub def
1328     n { \tx@NC } repeat
1329     \tx@EOC }
1330   ifelse>

```

AltCurve

Syntax:

array of points AltCurve

```

1331 \pst@def{AltCurve}<%
1332   { false \tx@NArray
1333     n 2 mul 2 roll
1334     [ n 2 mul 3 sub 1 roll ]
1335     aload /Points ED
1336     n 2 mul -2 roll }
1337   { false \tx@NArray }
1338   ifelse
1339   n 4 lt
1340   { n { pop pop } repeat }
1341   { \tx@BAC
1342     /n n 4 sub def
1343     n { \tx@NAC } repeat
1344     \tx@EAC }
1345   ifelse>

```

ClosedCurve

Syntax:

array of points ClosedCurve

```

1346 \pst@def{ClosedCurve}<%
1347   \tx@NArray
1348   n 3 lt
1349   { n { pop pop } repeat }
1350   { n 3 gt { \tx@CheckClosed } if
1351     6 copy n 2 mul 6 add 6 roll
1352     \tx@IC \tx@CC x y moveto
1353     n { \tx@NC } repeat
1354     closepath pop pop }
1355   ifelse>

```

curvature

```

1356 \def\psset@curvature#1{%
1357   \edef\pst@tempg{#1 }%

```



```

1358 \expandafter\psset@curvature\pst@tempg * * * \@nil}
1359 \def\psset@curvature#1 #2 #3 #4\@nil{%
1360 \pst@checknum{#1}\pst@tempg
1361 \pst@checknum{#2}\pst@tempg
1362 \pst@checknum{#3}\pst@tempg
1363 \edef\psk@curvature{\pst@tempg \pst@tempg \pst@tempg}
1364 \psset@curvature{1 .1 0}

```

\pscurve

```

1365 \def\pscurve{\def\pst@par{}\pst@object{pscurve}}
1366 \def\pscurve@i{%
1367 \pst@getarrows{%
1368 \begin@OpenObj
1369 \pst@getcoors[\pscurve@ii]}
1370 \def\pscurve@ii{%
1371 \addto@pscode{%
1372 \pst@cp
1373 \psk@curvature\space /c ED /b ED /a ED
1374 \ifshowpoints true \else false \fi
1375 \tx@OpenCurve}%
1376 \end@OpenObj}

```

\psecurve

```

1377 \def\psecurve{\def\pst@par{}\pst@object{psecurve}}
1378 \def\psecurve@i{%
1379 \pst@getarrows{%
1380 \begin@OpenObj
1381 \pst@getcoors[\psecurve@ii]}
1382 \def\psecurve@ii{%
1383 \addto@pscode{%
1384 \psk@curvature\space /c ED /b ED /a ED
1385 \ifshowpoints true \else false \fi
1386 \tx@AltCurve}%
1387 \end@OpenObj}

```

\psccurve

```

1388 \def\psccurve{\def\pst@par{}\pst@object{psccurve}}
1389 \def\psccurve@i{%
1390 \begin@ClosedObj
1391 \pst@getcoors[\psccurve@ii]}
1392 \def\psccurve@ii{%
1393 \addto@pscode{%
1394 \psk@curvature\space /c ED /b ED /a ED
1395 \ifshowpoints true \else false \fi
1396 \tx@ClosedCurve}%
1397 \def\pst@linetype{1}%
1398 \end@ClosedObj}

```

25 Dots

It would be nice to use a font, with hinting.

```

dotsize
1399 \def\psset@dotsize#1{%
1400   \edef\pst@tempg{#1 }%
1401   \expandafter\psset@@dotsize\pst@tempg -1 -1 -1\@nil}
1402 \def\psset@@dotsize#1 #2 #3\@nil{%
1403   \pst@checknum{#2}\pst@tempg
1404   \pssetlength\pst@ding{#1}%
1405   \edef\psk@dotsize{%
1406     /DS \pst@number\pst@ding \pst@tempg CLW mul add 2 div def }}
1407 \psset@dotsize{.5pt 2.5}

\psset@dotscale
1408 \def\psset@dotscale#1{\pst@getscale{#1}\psk@dotscale}
1409 \psset@dotscale{1}

\pst@Getangle
1410 \def\pst@Getangle#1#2{%
1411   \pst@getangle{#1}\pst@tempg
1412   \def\pst@temph{0. }%
1413   \ifx\pst@tempg\pst@temph
1414     \def#2{}%
1415   \else
1416     \edef#2{\pst@tempg\space rotate }%
1417   \fi}

dotangle
1418 \def\psset@dotangle#1{\pst@Getangle{#1}\psk@dotangle}
1419 \psset@dotangle{0}

\psdots
1420 \def\psdots{\def\pst@par{}\pst@object{psdots}}
1421 \def\psdots@i{%
1422   \begin@SpecialObj
1423     \pst@getcoors[\psdots@ii}
1424 \def\psdots@ii{%
1425   \addto@pscode{false \tx@NArray \psdots@iii}%
1426   \end@SpecialObj}
1427 \def\psdots@iii{%
1428   \psk@dotsize
1429   \@nameuse{psds@\psk@dotstyle}
1430   newpath
1431   n { gsave T \psk@dotangle \psk@dotscale Dot grestore } repeat}

EndDot

Syntax

{fill} {displace} EndDot

DS should be defined to be the dot size.
1432 \pst@def{EndDot}<%

```

```

1433 { /z DS def } { /z 0 def } ifelse
1434 /b ED
1435 0 z DS \tx@SD
1436 b { 0 z DS CLW sub \tx@SD } if
1437 0 DS z add CLW 4 div sub moveto>

\psas@oo
1438 \def\psas@oo{\pst@usecolor\psfillcolor true} true \psk@dotsize \tx@EndDot}

\psas@o
1439 \def\psas@o{\pst@usecolor\psfillcolor true} false \psk@dotsize \tx@EndDot}

\psas@**
1440 \@namedef{psas@**}{false} true \psk@dotsize \tx@EndDot}

\psas@*
1441 \@namedef{psas@*}{false} false \psk@dotsize \tx@EndDot}

```

26 Lines and polygons

```

linearc
1442 \newdimen\pslinearc
1443 \def\psset@linearc#1{\pssetlength\pslinearc{#1}}
1444 \psset@linearc{0pt}

\psline
1445 \def\psline{\def\pst@par{}\pst@object{psline}}
1446 \def\psline@i{%
1447   \pst@getarrows{%
1448     \begin@OpenObj
1449       \pst@getcoors[\psline@ii]}
1450 \def\psline@ii{%
1451   \addto@pscode{\pst@cp \psline@iii \tx@Line}%
1452 \end@OpenObj}
1453 \def\psline@iii{%
1454   \ifdim\pslinearc>\z@
1455     /r \pst@number\pslinearc def
1456     /Lineto { \tx@Arcto } def
1457   \else
1458     /Lineto /lineto load def
1459   \fi
1460   \ifshowpoints true \else false \fi}

\qline
1461 \def\qline(#1)(#2){%
1462   \def\pst@par{%
1463     \begin@SpecialObj
1464       \def\pst@linetype{0}%
1465       \pst@getcoor{#1}\pst@tempa
1466       \pst@@getcoor{#2}%

```

```

1467   \addto@pscode{%
1468       \pst@tempa moveto \pst@coor L
1469       \@nameuse{psls@\pslinestyle}}%
1470   \end@SpecialObj}

\pspolygon
1471 \def\pspolygon{\def\pst@par{}\pst@object{pspolygon}}
1472 \def\pspolygon@i{%
1473   \begin@ClosedObj
1474   \def\pst@cp{}%
1475   \pst@getcoors[\pspolygon@ii}
1476 \def\pspolygon@ii{%
1477   \addto@pscode{\psline@iii \tx@Polygon}%
1478   \def\pst@linetype{1}%
1479   \end@ClosedObj}

framearc
1480 \def\psset@framearc#1{\pst@checknum{#1}\psk@framearc}
1481 \psset@framearc{0}

cornersize
1482 \def\psset@cornersize#1{%
1483   \pst@expandafter\psset@@cornersize{#1}\@nil}
1484 \def\psset@@cornersize#1#2\@nil{%
1485   \if #1a\relax
1486     \def\psk@cornersize{\pst@number\pslinearc false }%
1487   \else
1488     \def\psk@cornersize{\psk@framearc true }%
1489   \fi}
1490 \psset@cornersize{relative}

Frame

Syntax

framearc/linearc bool x1 y1 x2 y2 dimen Frame

1491 \pst@def{Rect}<%
1492   x1 y1 y2 add 2 div moveto
1493   x1 y2 lineto
1494   x2 y2 lineto
1495   x2 y1 lineto
1496   x1 y1 lineto
1497   closepath>
1498 \pst@def{OvalFrame}<%
1499   x1 x2 eq y1 y2 eq or
1500   { pop pop x1 y1 moveto x2 y2 L }
1501   { y1 y2 sub abs x1 x2 sub abs
1502     2 copy gt { exch pop } { pop } ifelse
1503     2 div
1504     exch   % STACK: cornersize halfwidth boolean
1505     { dup 3 1 roll mul exch }
1506     if

```

```

1507     2 copy lt { pop } { exch pop } ifelse
1508     /b ED
1509     x1 y1 y2 add 2 div moveto
1510     x1 y2 x2 y2 b arcto
1511     x2 y2 x2 y1 b arcto
1512     x2 y1 x1 y1 b arcto
1513     x1 y1 x1 y2 b arcto
1514     16 { pop } repeat
1515     closepath }
1516   ifelse>
1517 \pst@def{Frame}<%
1518   CLW mul /a ED
1519   3 -1 roll 2 copy gt { exch } if
1520   a sub /y2 ED a add /y1 ED
1521   2 copy gt { exch } if
1522   a sub /x2 ED a add /x1 ED
1523   1 index 0 eq { pop pop \tx@Rect } { \tx@ovalFrame } ifelse>

```

dimen

```

1524 \def\psset@dimen#1{%
1525   \pst@expandafter\psset@@dimen{#1}\@nil}
1526 \def\psset@@dimen#1#2\@nil{%
1527   \if #1o\relax
1528     \def\psk@dimen{.5 }%
1529   \else
1530     \if #1m\relax
1531       \def\psk@dimen{0 }%
1532     \else
1533       \if #1i\relax
1534         \def\psk@dimen{- .5 }%
1535       \fi
1536     \fi
1537   \fi}
1538 \psset@dimen{outer}

```

\psframe

```

1539 \def\psframe{\def\pst@par{}\pst@object{psframe}}
1540 \def\psframe@i(#1){%
1541   \@ifnextchar({\psframe@ii(#1)}{\psframe@ii(0,0)(#1)}}
1542 \def\psframe@ii(#1)(#2){%
1543   \begin@ClosedObj
1544     \pst@getcoor{#1}\pst@tempa
1545     \pst@getcoor{#2}%
1546     \addto@pscode{\psk@cornersize \pst@tempa \pst@coor \psk@dimen \tx@Frame}%
1547     \def\pst@linetype{2}%
1548     \showpointsfalse
1549   \end@ClosedObj}

```

27 Curves

\psbezier

```

1550 \def\psbezier{\def\pst@par{}\pst@object{psbezier}}
1551 \def\psbezier@i{\pst@getarrows\psbezier@ii}
1552 \def\psbezier@ii#1(#2)#3(#4)#5(#6){%
1553   \@ifnextchar({\psbezier@iii{1}(#2)(#4)(#6)}%
1554     {\psbezier@iii{\z@}(0,0)(#2)(#4)(#6)}}}
1555 \def\psbezier@iii#1(#2)(#3)(#4)(#5){%
1556   \begin@OpenObj
1557   \pst@getcoor{#2}\pst@tempa
1558   \pst@getcoor{#3}\pst@tempb
1559   \pst@getcoor{#4}\pst@tempc
1560   \pst@getcoor{#5}\pst@tempd
1561   \pst@optcp{#1}\pst@tempa
1562   \ifshowpoints\psbezier@iv\fi
1563   \addto@pscode{
1564     \pst@tempb \pst@tempa ArrowA
1565     \pst@tempc \pst@tempd ArrowB
1566     curveto}%
1567   \end@OpenObj}
1568 \def\psbezier@iv{%
1569   \addto@pscode{%
1570     gsave
1571     \pst@tempa \pst@tempb \pst@tempc \pst@tempd
1572     newpath moveto L L L
1573     CLW 2 div SLW
1574     [ \psk@dash\space ] 0 setdash stroke
1575     grestore
1576     /Points [\pst@tempa\pst@tempb\pst@tempc\pst@tempd] def}}

```

\parabola

```

1577 \pst@def{Parab}<%
1578   /y0 exch def
1579   /x0 exch def
1580   /y1 exch def
1581   /x1 exch def
1582   /dx x0 x1 sub 3 div def
1583   /dy y0 y1 sub 3 div def
1584   x0 dx sub y0 dy add x1 y1 ArrowA
1585   x0 dx add y0 dy add x0 2 mul x1 sub y1 ArrowB
1586   curveto
1587   /Points [ x1 y1 x0 y0 x0 2 mul x1 sub y1 ] def>
1588 \def\parabola{\def\pst@par{}\pst@object{parabola}}
1589 \def\parabola@i{\pst@getarrows\parabola@ii}
1590 \def\parabola@ii#1(#2)#3(#4){%
1591   \begin@OpenObj
1592     \pst@getcoor{#2}\pst@tempa
1593     \pst@getcoor{#4}%
1594     \addto@pscode{\pst@tempa \pst@coor \tx@Parab}%
1595   \end@OpenObj}

```

28 Grids

gridwidth

```

1596 \def\psset@gridwidth#1{\pst@getlength{#1}\psk@gridwidth}

```

```

1597 \psset@gridwidth{.8pt}

      griddots
1598 \def\psset@griddots#1{%
1599   \pst@cntg=#1\relax
1600   \edef\psk@griddots{\the\pst@cntg}}
1601 \psset@griddots{0}

      gridcolor
1602 \def\psset@gridcolor#1{\pst@getcolor{#1}\psgridcolor}
1603 \psset@gridcolor{black}

      subgridwidth
1604 \def\psset@subgridwidth#1{\pst@getlength{#1}\psk@subgridwidth}
1605 \psset@subgridwidth{.4pt}

      subgridcolor
1606 \def\psset@subgridcolor#1{\pst@getcolor{#1}\pssubgridcolor}
1607 \psset@subgridcolor{gray}

      subgriddots
1608 \def\psset@subgriddots#1{%
1609   \pst@cntg=#1\relax\edef\psk@subgriddots{\the\pst@cntg}}
1610 \psset@subgriddots{0}

      subgriddiv
1611 \def\psset@subgriddiv#1{%
1612   \pst@cntg=#1\relax\edef\psk@subgriddiv{\the\pst@cntg}}
1613 \psset@subgriddiv{5}

      gridlabels
1614 \def\psset@gridlabels#1{\pst@getlength{#1}\psk@gridlabels}
1615 \psset@gridlabels{10pt}

      gridlabelcolor
1616 \def\psset@gridlabelcolor#1{\pst@getcolor{#1}\psgridlabelcolor}
1617 \psset@gridlabelcolor{black}

```

Grid

Syntax:

*x1 y1 x2 y2 x-origin y-origin x-divsize y-divsize
numsubdiv griddots labelcolor labelsize Grid*

Coordinates should all be integers. Font needs to be defined before invoking this procedure. This could probably be simplified.

```

1618 \pst@def{Grid}<%
1619   /a 4 string def           % Empty string
1620   /b ED                     % Label size

```

```

1621 /d ED % Label color procedure.
1622 /n ED % Number of grid dots
1623 cvi dup 1 lt { pop 1 } if /c ED % Number subdivisions
1624 c div dup 0 eq { pop 1 } if /cy ED
1625 c div dup 0 eq { pop 1 } if /cx ED % division spacing
1626 cy div cvi /y ED % origin y
1627 cx div cvi /x ED % origin x
1628 cy div cvi /y2 ED % y2
1629 cx div cvi /x2 ED % x2
1630 cy div cvi /y1 ED % y1
1631 cx div cvi /x1 ED % x1
1632 /h y2 y1 sub 0 gt { 1 } { -1 } ifelse def % Sign of y2-y1
1633 /w x2 x1 sub 0 gt { 1 } { -1 } ifelse def % Sign of x2-x1
1634 b 0 gt
1635 { /z1 b 4 div CLW 2 div add def
1636 /Helvetica findfont b scalefont setfont
1637 /b b .95 mul CLW 2 div add def }
1638 if
1639 gsave
1640 n 0 gt
1641 { 1 setlinecap [ 0 cy n div ] 0 setdash }
1642 { 2 setlinecap }
1643 ifelse
1644 /c x1 def /i 500 w mul x1 add def % Index
1645 /e y cy mul def /f y1 cy mul def /g y2 cy mul def
1646 x1 cx mul 0 T
1647 { newpath
1648 0 e moveto
1649 b 0 gt
1650 { gsave d c a cvs dup
1651 stringwidth pop /z2 ED
1652 w 0 gt {z1} {z1 z2 add neg} ifelse
1653 h 0 gt {b neg} {z1} ifelse
1654 rmoveto show grestore } if
1655 0 f moveto 0 g L stroke
1656 cx w mul 0 T
1657 c x2 eq c i eq or {exit} if
1658 /c c w add def
1659 } loop
1660 grestore
1661 gsave
1662 n 0 gt
1663 { 1 setlinecap [ 0 cx n div ] 0 setdash }
1664 { 2 setlinecap }
1665 ifelse
1666 /c y1 def /i 500 h mul y1 add def
1667 /e x cx mul def /f x1 cx mul def /g x2 cx mul def
1668 0 y1 cy mul T
1669 { newpath
1670 e 0 moveto
1671 b 0 gt { gsave d
1672 c a cvs dup
1673 stringwidth pop /z2 ED
1674 w 0 gt {z1 z2 add neg} {z1} ifelse

```



```

1675     h 0 gt {z1} {b neg} ifelse
1676     rmoveto show grestore } if
1677     f 0 moveto g 0 L stroke
1678     0 cy h mul T
1679     c y2 eq c i eq or {exit} if
1680     /c c h add def
1681 } loop
1682 grestore>

```

\psgrid

```

1683 \def\psgrid{\def\pst@par{}\pst@object{psgrid}}
1684 \def\psgrid@i{\@ifnextchar(
1685   {\psgrid@ii}{\expandafter\psgrid@iv\pic@coor}}
1686 \def\psgrid@ii(#1){\@ifnextchar(
1687   {\psgrid@iii(#1)}{\psgrid@iv(0,0)(0,0)(#1)}}
1688 \def\psgrid@iii(#1)(#2){\@ifnextchar(
1689   {\psgrid@iv(#1)(#2)}{\psgrid@iv(#1)(#1)(#2)}}
1690 \def\psgrid@iv(#1)(#2)(#3){%
1691   \begin@SpecialObj
1692   \pst@getcoor{#1}\pst@tempa
1693   \pst@getcoor{#2}\pst@tempb
1694   \pst@getcoor{#3}%
1695   \ifnum\psk@subgriddiv>1
1696     \addto@pscode{gsave
1697       \psk@subgridwidth SLW \pst@usecolor\psubgridcolor
1698       \pst@tempb \pst@coor \pst@tempa
1699       \pst@number\psxunit \pst@number\psyunit
1700       \psk@subgriddiv\space \psk@subgriddots\space
1701       } 0 \tx@Grid grestore}%
1702   \fi
1703   \addto@pscode{gsave
1704     \psk@gridwidth SLW \pst@usecolor\psgridcolor
1705     \pst@tempb \pst@coor \pst@tempa
1706     \pst@number\psxunit \pst@number\psyunit
1707     1 \psk@griddots\space { \pst@usecolor\psgridlabelcolor }
1708     \psk@gridlabels \tx@Grid grestore}%
1709   \end@SpecialObj}

```

29 LR-box commands

\ifpsmathbox, \everypsbox

```

1710 \newif\ifpsmathbox
1711 \psmathboxtrue
1712 \def\pst@mathflag{z@}
1713 \newtoks\everypsbox

```

\pst@makenotverbbox

```

1714 \long\def\pst@makenotverbbox#1#2{%
1715   \edef\pst@mathflag{%
1716     \ifpsmathbox\ifmmode\ifinner 1\else 2\fi\else z@\fi\else z@\fi}%
1717   \setbox\pst@hbox=\hbox{%
1718     \ifcase\pst@mathflag\or$\m@th\textstyle\or$\m@th\displaystyle\fi

```

```

1719     {\the\everypsbox#2}%
1720     \ifnum\pst@mathflag>z@$\fi}%
1721 #1}

```

`\pst@makeverbbox`

There is no way to do this such that with

```
\psframebox{\aftergroup\foo}
```

`\foo` does not end up outside the box. That is why this is not the default mode.

```

1722 \def\pst@makeverbbox#1{%
1723   \def\pst@afterbox{#1}%
1724   \edef\pst@mathflag{%
1725     \ifpsmathbox\ifmmode\ifinner 1\else 2\fi\else z@\fi\else z@\fi}%
1726   \afterassignment\pst@beginbox
1727   \setbox\pst@hbox\hbox}
1728 \def\pst@beginbox{%
1729   \ifcase\pst@mathflag\or$\m@th\or$\m@th\displaystyle\fi
1730   \bgroup\aftergroup\pst@endbox
1731   \the\everypsbox}
1732 \def\pst@endbox{%
1733   \ifnum\pst@mathflag>z@$\fi
1734   \egroup
1735   \pst@afterbox}

```

`\psverbboxtrue`, `\psverbboxfalse`

```

1736 \def\pst@makebox{\pst@makebox}
1737 \def\psverbboxtrue{\def\pst@makebox{\pst@makeverbbox}}
1738 \def\psverbboxfalse{\def\pst@makebox{\pst@makenotverbbox}}
1739 \psverbboxfalse

```

`\pst@longbox`, `\pst@makelongbox`

There is no way to do this such that with

```
\psframebox{\aftergroup\foo}
```

`\foo` does not end up outside the box. That is why this is not the default mode.

```

1740 \def\pst@longbox{%
1741   \def\pst@makebox{%
1742     \gdef\pst@makebox{\pst@makebox}%
1743     \pst@makelongbox}}
1744 \def\pst@makelongbox#1{%
1745   \def\pst@afterbox{#1}%
1746   \edef\pst@mathflag{%
1747     \ifpsmathbox\ifmmode\ifinner 1\else 2\fi\else z@\fi\else z@\fi}%
1748   \setbox\pst@hbox\hbox\bgroup
1749   \aftergroup\pst@afterbox
1750   \ifcase\pst@mathflag\or$\m@th\or$\m@th\displaystyle\fi
1751   \begingroup
1752     \the\everypsbox}

```

```

1753 \def\pst@endlongbox{%
1754     \endgroup
1755     \ifnum\pst@mathflag>\z$\fi
1756 \egroup}

\pslongbox

1757 \def\pslongbox#1#2{%
1758     \@namedef{#1}{\pst@longbox#2}%
1759     \@namedef{end#1}{\pst@endlongbox}}

```

30 Frame boxes

framesep

```

1760 \newdimen\psframesep
1761 \def\psset@framesep#1{\pssetlength\psframesep{#1}}
1762 \psset@framesep{3pt}

```

boxsep

```

1763 \newif\ifpsboxsep
1764 \def\psset@boxsep#1{\@nameuse{psboxsep#1}}
1765 \psset@boxsep{true}

```

\pst@useboxpar

```

1766 \def\pst@useboxpar{%
1767     \use@par
1768     \if@star
1769         \let\pslinecolor\psfillcolor
1770         \solid@star
1771         \let\solid@star\relax
1772     \fi
1773     \ifpsdoubleline \pst@setdoublesep \fi}

```

\psframebox

\psframebox puts its argument in an \hbox and draws a frame around it with thickness \pst@linewidth, and with distance \pst@framesep between each side of the frame (between the line making up each side) and each side of the box. The result is a box with no depth and with width and height equal to the width and height of the original box, plus 2(\pslinewidth+\psframesep).

\pst@dima is set to the distance between each side of the original box and the outer side of the frame (i.e., the side of the resulting box). \pst@dimb is set to the depth of the resulting box, \pst@dimc is set to the height plus depth of this box, and \pst@dimd is set to the width. \psframe does the drawing of the frame.

```

1774 \def\psframebox{\def\pst@par{}\pst@object{psframebox}}
1775 \def\psframebox@i{\pst@makebox\psframebox@ii}
1776 \def\psframebox@ii{%
1777     \begingroup
1778         \pst@useboxpar
1779         \pst@dima=\pslinewidth
1780         \advance\pst@dima by \psframesep

```

```

1781 \pst@dimc=\wd\pst@hbox\advance\pst@dimc by \pst@dima
1782 \pst@dimb=\dp\pst@hbox\advance\pst@dimb by \pst@dima
1783 \pst@dimd=\ht\pst@hbox\advance\pst@dimd by \pst@dima
1784 \setbox\pst@hbox=\hbox{%
1785   \ifpsboxsep\kern\pst@dima\fi
1786   \begin@ClosedObj
1787     \addto@pscode{%
1788       \psk@cornersize
1789       \pst@number\pst@dima neg
1790       \pst@number\pst@dimb neg
1791       \pst@number\pst@dimc
1792       \pst@number\pst@dimd
1793       .5
1794       \tx@Frame}%
1795   \def\pst@linetype{2}%
1796   \showpointsfalse
1797   \end@ClosedObj
1798   \box\pst@hbox
1799   \ifpsboxsep\kern\pst@dima\fi}%
1800 \ifpsboxsep\dp\pst@hbox=\pst@dimb\ht\pst@hbox=\pst@dimd\fi
1801 \leavevmode\box\pst@hbox
1802 \endgroup}

```

\psdblframebox

```

1803 \def\psdblframebox{\def\pst@par{}\pst@object{psdblframebox}}
1804 \def\psdblframebox@i{\addto@par{doubleline=true}\psframebox@i}

```

\psclip, \endclip

Clipping involves drawing graphics objects, not grouped by `gsave` and `grestore`, which may affect the graphics environment. Furthermore, to reset the clipping path, we must either use `grestore` or `initclip`, neither of which is robust.

```

1805 \def\psclip#1{%
1806   \leavevmode
1807   \begingroup
1808     \begin@psclip
1809     \begingroup
1810       \def\use@pscode{%
1811         \pstVerb{%
1812           \pst@dict
1813           /mtrxc CM def
1814           CP CP T
1815           \tx@STV
1816           \psk@origin
1817           \psk@swapaxes
1818           newpath
1819           \pst@code
1820           clip
1821           newpath
1822           mtrxc setmatrix
1823           moveto
1824           0 setgray
1825         end}%

```

```

1826     \gdef\pst@code{}}%
1827     \def\@multips(##1)(##2)##3##4{\pst@misplaced\multips}%
1828     \def\nc@object##1##2##3##4{\pst@misplaced{node connection}}%
1829     \hbox to\z@{##1}%
1830     \endgroup
1831 \def\endpsclip{%
1832     \end@psclip
1833     \endgroup}%
1834 \ignorespaces}
1835 \def\endpsclip{\pst@misplaced\endpsclip}
1836 \let\begin@psclip\relax
1837 \def\end@psclip{\pstVerb{currentpoint initclip moveto}}
1838 \def\AltClipMode{%
1839     \def\end@psclip{\pstVerb{\pst@grestore}}%
1840     \def\begin@psclip{\pstVerb{gsave}}}}

```

\psclipbox

```

1841 \def\clipbox{\@ifnextchar[{\psclipbox@}{psclipbox@[z@]}}
1842 \def\clipbox@[#1]{\pst@makebox\psclipbox@#{#1}}
1843 \def\clipbox@#1{%
1844     \pssetlength\pst@dimg{#1}%
1845     \leavevmode\hbox{%
1846         \begin@psclip
1847         \pst@Verb{%
1848             CM \tx@STV CP T newpath
1849             /a \pst@number\pst@dimg def
1850             /w \pst@number{\wd\pst@hbox}a add def
1851             /d \pst@number{\dp\pst@hbox}a add neg def
1852             /h \pst@number{\ht\pst@hbox}a add def
1853             a neg d moveto
1854             a neg h L
1855             w h L
1856             w d L
1857             closepath
1858             clip
1859             newpath
1860             O O moveto
1861             setmatrix}%
1862         \unhbox\pst@hbox
1863         \end@psclip}}

```

\psshadowbox

```

1864 \def\psshadowbox{%
1865     \def\pst@par{\pst@object{psshadowbox}}
1866 \def\psshadowbox@i{\pst@makebox\psshadowbox@ii}
1867 \def\psshadowbox@ii{%
1868     \begin@group
1869         \pst@useboxpar
1870         \psshadowtrue
1871         \psboxseptrue
1872         \def\psk@shadowangle{-45 }%
1873         \setbox\pst@hbox=\hbox{\psframebox@ii}%
1874         \pst@dimh=\psk@shadowsize\p@

```

```

1875 \pst@dimh=.7071\pst@dimh
1876 \pst@ding=\dp\pst@hbox
1877 \advance\pst@ding\pst@dimh
1878 \dp\pst@hbox=\pst@ding
1879 \pst@ding=\wd\pst@hbox
1880 \advance\pst@ding\pst@dimh
1881 \wd\pst@hbox=\pst@ding
1882 \leavevmode
1883 \box\pst@hbox
1884 \endgroup}
1885 %
1886 %
1887 % \begin{macro}{\pscirclebox}
1888 % "\pscirclebox@ii"'s argument is a hook that is used by node commands.
1889 % \begin{macrocode}
1890 \def\pscirclebox{\def\pst@par{}\pst@object{pscirclebox}}
1891 \def\pscirclebox@i{\pst@makebox{\pscirclebox@ii{}}}
1892 \def\pscirclebox@ii#1{%
1893 \begingroup
1894 \pst@useboxpar
1895 \setbox\pst@hbox=\hbox{#1\pscirclebox@iii\box\pst@hbox}%
1896 \ifpsboxsep
1897 \pst@dima=.5\wd\pst@hbox
1898 \pst@pyth\pst@dima\pst@dimb\pst@dimc
1899 \advance\pst@dimc\pslinewidth
1900 \advance\pst@dimc\psframesep
1901 \setbox\pst@hbox=\hbox to2\pst@dimc{%
1902 \hss
1903 \vbox{\vskip\pst@dimc\vskip-\pst@dimb\box\pst@hbox}%
1904 \hss}%
1905 \advance\pst@dimc-\pst@dimb
1906 \dp\pst@hbox=\pst@dimc
1907 \fi
1908 \leavevmode\box\pst@hbox
1909 \endgroup}
1910 \def\pscirclebox@iii{%
1911 \if@star
1912 \pslinewidth\z@
1913 \pstverb{\pst@dict \tx@STP \pst@usecolor\psfillcolor
1914 newpath \pscirclebox@iv \tx@SD end}%
1915 \else
1916 \begin@ClosedObj
1917 \def\pst@linetype{4}\showpointsfalse
1918 \addto@pscode{%
1919 \pscirclebox@iv CLW 2 div add 0 360 arc closepath}%
1920 \end@ClosedObj
1921 \fi}
1922 \def\pscirclebox@iv{%
1923 \pst@number{\wd\pst@hbox}2 div
1924 \pst@number{\ht\pst@hbox}\pst@number{\dp\pst@hbox}add 2 div
1925 2 copy \pst@number{\dp\pst@hbox}sub 4 2 roll
1926 \tx@Pyth \pst@number\psframesep add }

```

`\psovalbox`

The argument of `\psovalbox@ii` is a hook used by node commands.

```
1927 \def\psovalbox{\def\pst@par{}\pst@object{psovalbox}}
1928 \def\psovalbox@ii{\pst@makebox{\psovalbox@ii{}}}
1929 \def\psovalbox@ii#1{%
1930   \begingroup
1931     \pst@useboxpar
1932     \pst@dimd=.707\pslinewidth\advance\pst@dimd by 1.414\psframesep
1933     \pst@ding=\ht\pst@hbox\advance\pst@ding\dp\pst@hbox
1934     \pst@dimb=.707\pst@ding\advance\pst@dimb\pst@dimd
1935     \pst@dima=.707\wd\pst@hbox\advance\pst@dima\pst@dimd
1936     \setbox\pst@hbox=\hbox{#1\psovalbox@iii\box\pst@hbox}%
1937     \ifpsboxsep
1938       \setbox\pst@hbox\hbox to 2\pst@dima{\hss\unhbox\pst@hbox\hss}%
1939       \advance\pst@dimb-.5\pst@ding
1940       \pst@ding\ht\pst@hbox
1941       \advance\pst@ding\pst@dimb
1942       \ht\pst@hbox=\pst@dimb
1943       \pst@ding=\dp\pst@hbox
1944       \advance\pst@ding\pst@dimb
1945       \dp\pst@hbox=\pst@dimb
1946     \fi
1947     \leavevmode\box\pst@hbox
1948   \endgroup}
1949 \def\psovalbox@iii{%
1950   \begin@ClosedObj
1951   \addto@pscode{%
1952     0 360
1953     \pst@number\pst@dima \pst@number\pst@dimb
1954     \pst@number{\wd\pst@hbox}2 div
1955     \pst@number\pst@ding 2 div \pst@number{\dp\pst@hbox}sub
1956     \tx@Ellipse
1957     closepath}%
1958   \def\pst@linetype{2}%
1959   \end@ClosedObj}
```

31 Circles, discs and ellipses

`\psset@arcsep`, `\psk@arcsepA`, `\psk@arcsepB`

```
1960 \def\psset@arcsepA#1{\pst@getlength{#1}\psk@arcsepA}
1961 \def\psset@arcsepB#1{\pst@getlength{#1}\psk@arcsepB}
1962 \def\psset@arcsep#1{%
1963   \psset@arcsepA{#1}\let\psk@arcsepB\psk@arcsepA}
1964 \psset@arcsep{0}
```

`\tx@Arc`

Syntax:

angle {*arrow*} {*add/sub*} ArcArrow *angle*

r =radius and $c=57.2957/r$ should also be defined.

```

1965 \pst@def{ArcArrow}<%
1966 /d ED      % add/sub
1967 /b ED      % arrow procedure
1968 /a ED      % angle
1969 gsave
1970   newpath
1971   0 -1000 moveto
1972   clip      % Set clippath far from arrow.
1973   newpath
1974   0 1 0 0 b      % Draw arrow to determine length.
1975 grestore
1976 c mul
1977 /e ED      % /e equals angle to adjust for arrow length.
1978 pop pop pop
1979 r a e d \tx@PtoC % 'a e d' is end angle for arrow.
1980 y add exch x add exch
1981 r a \tx@PtoC      % Now arrow end coor and begin coor are on stack.
1982 y add exch x add exch
1983 b pop pop pop pop % Draw arrow, and discard coordinates.
1984 a e d              % End angle of arrow.
1985 CLW 8 div c mul neg d> % Adjust angle to give a little overlap.

```

\psarc

```

1986 \def\psarc{\def\pst@par{}\pst@object{psarc}}
1987 \def\psarc@i{%
1988   \@ifnextchar({\psarc@iii}{\psarc@ii}}
1989 \def\psarc@ii#1{\addto@par{arrows=#1}%
1990   \@ifnextchar({\psarc@iii}{\psarc@iii(0,0)}}
1991 \def\psarc@iii(#1)#2#3#4{%
1992   \begin@OpenObj
1993     \pst@getangle{#3}\pst@tempa
1994     \pst@getangle{#4}\pst@tempb
1995     \pst@getcoor{#1}%
1996     \pssetlength\pst@dima{#2}%
1997     \addto@pscode{\psarc@iv \psarc@v}%
1998     \gdef\psarc@type{0}%
1999     \showpointsfalse
2000   \end@OpenObj}
2001 \def\psarc@iv{%
2002   \pst@coor /y ED /x ED
2003   /r \pst@number\pst@dima def
2004   /c 57.2957 r \tx@Div def
2005   /angleA
2006     \pst@tempa
2007     \psk@arcsepA c mul 2 div
2008     \ifcase \psarc@type add \or sub \fi
2009   def
2010   /angleB
2011     \pst@tempb
2012     \psk@arcsepB c mul 2 div
2013     \ifcase \psarc@type sub \or add \fi
2014   def

```



```

2015 \ifshowpoints\psarc@showpoints\fi
2016 \ifx\psk@arrowA\@empty
2017   \ifnum\psk@liftpen=2
2018     r angleA \tx@PtoC
2019     y add exch x add exch
2020     moveto
2021   \fi
2022 \fi}
2023 \def\psarc@v{%
2024   x y r
2025   angleA
2026   \ifx\psk@arrowA\@empty\else
2027     { ArrowA CP }
2028     { \ifcase\psarc@type add \or sub \fi }
2029     \tx@ArcArrow
2030   \fi
2031   angleB
2032   \ifx\psk@arrowB\@empty\else
2033     { ArrowB }
2034     { \ifcase\psarc@type sub \or add \fi }
2035     \tx@ArcArrow
2036   \fi
2037   \ifcase\psarc@type arc \or arcn \fi}
2038 \def\psarc@type{0}
2039 \def\psarc@showpoints{%
2040   gsave
2041     newpath
2042     x y moveto
2043     x y r \pst@tempa \pst@tempb
2044     \ifcase\psarc@type arc \or arcn \fi
2045     closepath
2046     CLW 2 div SLW
2047     [ \psk@dash\space ] 0 setdash stroke
2048     grestore }

\psarcn

2049 \def\psarcn{\def\pst@par{}\pst@object{psarcn}}
2050 \def\psarcn@i{\def\psarc@type{1}\psarc@i}

\pscircle

2051 \def\pscircle{\def\pst@par{}\pst@object{pscircle}}
2052 \def\pscircle@i{\@ifnextchar{\pscircle@do}{\pscircle@do(0,0)}}
2053 \def\pscircle@do(#1)#2{%
2054   \if@star
2055     {\use@par\qdisk(#1){#2}}%
2056   \else
2057     \begin@ClosedObj
2058       \pst@@getcoor{#1}%
2059       \pssetlength\pst@dimc{#2}%
2060       \def\pst@linetype{4}%
2061       \addto@pscode{%
2062         \pst@coor
2063         \pst@number\pst@dimc

```

```

2064     \psk@dimen CLW mul sub
2065     0 360 arc
2066     closepath}%
2067     \showpointsfalse
2068     \end@ClosedObj
2069     \fi
2070     \ignorespaces}

\qdisk

2071 \def\qdisk(#1)#2{%
2072   \def\pst@par{%
2073     \begin@SpecialObj
2074       \pst@@getcoor{#1}%
2075       \pssetlength\pst@dimc{#2}%
2076       \addto@pscode{\pst@coor \pst@number\pst@dimc \tx@SD}%
2077     \end@SpecialObj}

\pswedge

2078 \def\pswedge{\def\pst@par{\pst@object{pswedge}}
2079 \def\pswedge@i{@ifnextchar({\pswedge@ii}{\pswedge@ii(0,0)}}
2080 \def\pswedge@ii(#1)#2#3#4{%
2081   \begin@ClosedObj
2082     \pssetlength\pst@dimc{#2}
2083     \pst@getangle{#3}\pst@tempa
2084     \pst@getangle{#4}\pst@tempb
2085     \pst@@getcoor{#1}%
2086     \def\pst@linetype{1}%
2087     \addto@pscode{%
2088       \pst@coor
2089       2 copy
2090       moveto
2091       \pst@number\pst@dimc \psk@dimen CLW mul sub % Adjusted radius
2092       \pst@tempa \pst@tempb
2093       arc
2094       closepath}%
2095     \showpointsfalse
2096     \end@ClosedObj}

```

Ellipse

Syntax:

angle1 angle2 x-radius y-radius x-origin y-origin Ellipse

```

2097 \pst@def{Ellipse}<%
2098 /mtrx CM def
2099 T
2100 scale
2101 0 0 1 5 3 roll arc
2102 mtrx setmatrix>

```

\psellipse

```

2103 \def\psellipse{\def\pst@par{}\pst@object{psellipse}}
2104 \def\psellipse@i(#1){\@ifnextchar(
2105   {\psellipse@ii(#1)}{\psellipse@ii(0,0)(#1)}}
2106 \def\psellipse@ii(#1)(#2){%
2107   \begin@ClosedObj
2108     \pst@getcoor{#1}\pst@tempa
2109     \pst@@getcoor{#2}%
2110     \addto@pscode{%
2111       0 360
2112       \pst@coor
2113       \ifdim\psk@dimen\p@=\z@\else
2114         \psk@dimen CLW mul dup 3 1 roll
2115         sub 3 1 roll sub exch
2116       \fi
2117       \pst@tempa
2118       \tx@Ellipse
2119       closepath}%
2120     \def\pst@linetype{2}%
2121   \end@ClosedObj}

```

32 Repetition

\multirput

```

2122 \def\multirput{%
2123   \begin@group\pst@getref{\pst@getrputrot\multirput@i}}
2124 \def\multirput@i(#1){\@ifnextchar(
2125   {\multirput@ii(#1)}{\multirput@ii(0,0)(#1)}}
2126 \def\multirput@ii(#1,#2)(#3,#4)#5{%
2127   \pst@makebox{\multirput@iii(#1,#2)(#3,#4){#5}}
2128 \def\multirput@iii(#1,#2)(#3,#4)#5{%
2129   \pst@makesmall\pst@hbox
2130   \ifx\pst@rot@empty\else\pst@rotate\pst@hbox\fi
2131   \pssetxlength\pst@dima{#1}\pssetylength\pst@dimb{#2}
2132   \pssetxlength\pst@dimc{#3}\pssetylength\pst@dimd{#4}
2133   \pst@cntg=#5\relax\pst@cnth=0\relax
2134   \leavevmode
2135   \loop\ifnum\pst@cntg>\pst@cnth
2136     \vbox to \z@{\vss\hbox to \z@{
2137       \kern\pst@dima\copy\pst@hbox\hss}\vskip\pst@dimb}%
2138     \advance\pst@dima by\pst@dimc
2139     \advance\pst@dimb by\pst@dimd
2140     \advance\pst@cnth by 1
2141   \repeat
2142   \end@group\ignorespaces}

```

\multips

```

2143 \def\multips{\begin@group\pst@getrputrot\multips@i}
2144 \def\multips@i(#1){\@ifnextchar({\@multips@ii(#1)}{\@multips@ii(0,0)(#1)}}
2145 \def\@multips@ii(#1)(#2)#3#4{%
2146   \pst@getcoor{#1}\pst@tempa
2147   \pst@@getcoor{#2}%
2148   \pst@canta=#3\relax
2149   \addto@pscode{%

```

```

2150     \pst@tempa T \the\pst@cmta\space \pslbrace
2151     gsave \ifx\pst@rot\@empty\else\pst@rot rotate \fi }%
2152 \hbox to\z@{%
2153     \def\init@pscode{%
2154     \addto@pscode{%
2155         gsave
2156         \pst@number\pslinewidth SLW
2157         \pst@usecolor\pslinecolor}}%
2158     \def\use@pscode{\addto@pscode{grestore}}%
2159     \def\psclip##1{\pst@misplaced\psclip}%
2160     \def\nc@object##1##2##3##4{\pst@misplaced{node connection}}%
2161     #4}%
2162     \addto@pscode{grestore \pst@coord T \psrbrace repeat}%
2163     \leavevmode
2164     \use@pscode
2165 \endgroup
2166 \ignorespaces}

```

33 Scaling

\scalebox

```

2167 \def\sbox#1{%
2168     \begingroup
2169     \pst@getscale{#1}\pst@tempa
2170     \pst@makebox{\@scalebox}}
2171 \def\@scalebox{%
2172     \leavevmode
2173     \ifx\pst@tempa\@empty
2174     \box\pst@hbox
2175     \else
2176     \hbox{%
2177         \ht\pst@hbox=\pst@temp\ht\pst@hbox%
2178         \dp\pst@hbox=\pst@temp\dp\pst@hbox%
2179         \pst@dima=\pst@temp\wd\pst@hbox%
2180         \ifdim\pst@dima<\z@\kern-\pst@dima\fi
2181         \pst@Verb{CP CP T \pst@tempa \tx@NET}%
2182         \hbox to \z@{\box\pst@hbox\hss}%
2183         \pst@Verb{
2184             CP CP T
2185             1 \pst@temp\space div 1 \pst@temp\space div scale
2186             \tx@NET}%
2187         \ifdim\pst@dima>\z@\kern\pst@dima\fi}%
2188     \fi
2189     \endgroup}
2190 \pslongbox{Scalebox}{\scalebox}

```

\scaleboxto

```

2191 \def\sboxto(#1,#2){%
2192     \begingroup
2193     \pssetlength\pst@dima{#1}%
2194     \pssetlength\pst@dimb{#2}%
2195     \pst@makebox{\@scaleboxto\@scalebox}}
2196 \def\@scaleboxto{%

```

```

2197 \ifdim\pst@dima=\z@\else
2198 \pst@divide{\pst@dima}{\wd\pst@hbox}\pst@tempg
2199 \fi
2200 \ifdim\pst@dimb=\z@
2201 \let\pst@temph\pst@tempg
2202 \else
2203 \pst@dimc=\ht\pst@hbox\advance\pst@dimc\dp\pst@hbox
2204 \pst@divide{\pst@dimb}{\pst@dimc}\pst@temph
2205 \ifdim\pst@dima=\z@\let\pst@tempg\pst@temph\fi
2206 \fi
2207 \edef\pst@tempa{\pst@tempg\space\pst@temph\space scale }%
2208 \ifdim\pst@dima=\z@
2209 \ifdim\pst@dimb=\z@
2210 \@pstrickserr{%
2211 \string\scaleboxto\space dimensions cannot both be zero}\@ehpa
2212 \def\pst@tempa{}%
2213 \fi\fi}
2214 \pslongbox{Scaleboxto}{\scaleboxto}

```

34 Rotation: The simple version

```

\tx@Rot
2215 \pst@def{Rot}<\pstrotate>

\rotateleft, \rotateright, \rotatedown

These are pretty standard, except that they do not use gsave and grestore.

2216 \def\rotateleft{\pst@makebox{\@rotateleft\pst@hbox}}
2217 \def\@rotateleft#1{%
2218 \leavevmode\hbox{\hskip\ht#1\hskip\dp#1\vbox{\vskip\wd#1%
2219 \pst@Verb{90 \tx@Rot}
2220 \vbox to \z@{\vss\hbox to \z@{\box#1\hss}\vskip\z@}%
2221 \pst@Verb{-90 \tx@Rot}}}}
2222 \def\rotateright{\pst@makebox{\@rotateright\pst@hbox}}
2223 \def\@rotateright#1{%
2224 \hbox{\hskip\ht#1\hskip\dp#1\vbox{\vskip\wd#1%
2225 \pst@Verb{-90 \tx@Rot}
2226 \vbox to \z@{\hbox to \z@{\hss\box#1}\vss}%
2227 \pst@Verb{90 \tx@Rot}}}}
2228 \def\rotatedown{\pst@makebox{\@rotatedown\pst@hbox}}
2229 \def\@rotatedown#1{%
2230 \hbox{\hskip\wd#1\vbox{\vskip\ht#1\vskip\dp#1%
2231 \pst@Verb{180 \tx@Rot}%
2232 \vbox to \z@{\hbox to \z@{\box#1\hss}\vss}%
2233 \pst@Verb{-180 \tx@Rot}}}}
2234 \pslongbox{Rotateleft}{\rotateleft}
2235 \pslongbox{Rotateright}{\rotateright}
2236 \pslongbox{Rotatedown}{\rotatedown}

```

35 \rput and company

\rput and similar commands are divided into four steps:

1. The four arguments are collected:
 - (a) The reference point argument is stored in `\refpoint@x` and `\refpoint@y`.
 - (b) The rotation angle is store in `\pst@rot`.
 - (c) The translation coordinate is passed to the command that is returned to after the box is made.
 - (d) The RH-box is assigned to the register `\pst@hbox`.
2. The box is made zero-dimension and positioned at the reference point by `\pst@makesmall`.
3. The box is rotated by `\pst@rotate`.
4. The box is translated by `\psput@`.

35.1 Reference point

`\pst@getref`

```

2237 \def\pst@getref#1{%
2238   \@ifnextchar[%
2239     {\def\refpoint@x{.5}\def\refpoint@y{.5}\pst@@getref{#1}}%
2240     {\let\refpoint@x\relax#1}}
2241 \def\pst@@getref#1[#2]{%
2242   \pst@expandafter\pst@@@getref{#2}\@empty,,\@nil#1}
2243 \def\pst@@@getref#1#2,#3,#4\@nil{%
2244   \ifx\@empty#3\@empty
2245     \@nameuse{getref@#1}\@nameuse{getref@#2}%
2246   \else
2247     \pst@checknum{#1#2}\refpoint@x
2248     \pst@checknum{#3}\refpoint@y
2249   \fi}
2250 \def\getref@t{\def\refpoint@y{1}}
2251 \def\getref@b{\def\refpoint@y{0}}
2252 \def\getref@B{\let\refpoint@y\relax}
2253 \def\getref@l{\def\refpoint@x{0}}
2254 \def\getref@r{\def\refpoint@x{1}}

```

`\pst@makesmall`

```

2255 \def\pst@makesmall#1{%
2256   \ifx\refpoint@x\relax
2257     \setbox#1=\hbox to\z@{\hss\vbox to \z@{\vss\box#1\vss}\hss}%
2258   \else
2259     \pst@@makesmall{#1}%
2260   \fi}
2261 \def\pst@@makesmall#1{%
2262   \pst@dimh=\refpoint@x\wd#1%
2263   \ifx\refpoint@y\relax
2264     \pst@dimg=\dp#1%
2265   \else
2266     \pst@dimg=\refpoint@y\ht#1%
2267     \advance\pst@dimg\refpoint@y\dp#1%
2268   \fi
2269   \setbox#1=\hbox to\z@{%

```

```
2270 \hskip-\pst@dimh\vbox to\z@{\vss\box#1\vskip-\pst@ding}\hss}}
```

35.2 Rotation

`\pst@getrputrot`

```
2271 \def\pst@getrputrot#1{%
2272 \ifnextchar(%
2273 {\def\pst@rot{#1}%
2274 {\pst@getrot{\ifnextchar({#1}{#1(0,0)}}}}
```

`\pst@getrot`

```
2275 \def\pst@getrot#1#2{%
2276 \pst@expandafter{\ifnextchar*{\pst@@@getrot}{\pst@@getrot}}{#2}\@nil
2277 \ifx\pst@rotlist\@empty\else
2278 \edef\pst@rotlist{\pst@rotlist \pst@rot add }%
2279 \fi
2280 #1}
2281 \def\pst@@getrot#1\@nil{%
2282 \def\next##1@#1=##2@##3\@nil{%
2283 \ifx\relax##2%
2284 \pst@getangle{#1}\pst@rot
2285 \else
2286 \def\pst@rot{##2}%
2287 \fi}%
2288 \expandafter\next\pst@rotable @#1=\relax @\@nil}
2289 \def\pst@@@getrot#1#2\@nil{%
2290 \pst@@getrot#2\@nil
2291 \edef\pst@rot{\pst@rotlist neg \ifx\pst@rot\@empty\else\pst@rot add \fi}}%
2292 \def\pst@rotlist{0 }
2293 \def\pst@rot{}
```

`\pst@rotable`

The trailing spaces must be included, except when empty.

```
2294 \def\pst@rotable{%
2295 @O=%
2296 @U=%
2297 @L=90 %
2298 @D=180 %
2299 @R=-90 %
2300 @N=\pst@rotlist neg %
2301 @W=\pst@rotlist neg 90 add %
2302 @S=\pst@rotlist neg 180 add %
2303 @E=\pst@rotlist neg 90 sub }
```

`\pst@rotate`

The last argument should be the register for a zero-dimensional box that is to be rotated. By first putting the box in a zero-dimension box centered at the reference point of the original box, we do not have to use `gsave` and `grestore`.

```
2304 \def\pst@rotate#1{%
2305 \setbox#1=\hbox{%
```

```

2306 \pst@Verb{\pst@rot \tx@Rot}%
2307 \box#1%
2308 \pst@Verb{\pst@rot neg \tx@Rot}}

```

35.3 Translation

`\psput@cartesian`, `\psput@special`

`\psput@` is defined by the `\NormalCoor` and `\SpecialCoor` commands to invoke either `\psput@cartesian` or `\psput@special`.

`\psput@cartesian` is for Cartesian coordinates only. T_EX does the translation.

`\psput@special` works for any coordinates. PostScript does the translation. `/lmtx` is used to store a stack of transformation for nested translations.

```

2309 \def\psput@cartesian#1{%
2310 \hbox to \z@{\kern\pst@dimg{\vbox to \z@{\vss\box#1\vskip\pst@dimh}\hss}}}
2311 \def\psput@special#1{%
2312 \hbox{%
2313 \pst@Verb{{ \pst@coor } \tx@PutCoor \tx@PutBegin}%
2314 \box#1%
2315 \pst@Verb{\tx@PutEnd}}}
2316 \pst@def{PutCoor}<%
2317 gsave
2318 CP T
2319 CM
2320 \tx@STV
2321 exch exec
2322 moveto
2323 setmatrix
2324 CP
2325 grestore>
2326 \pst@def{PutBegin}<%
2327 /lmtx [ tx@Dict /lmtx known { lmtx aload pop } if CM ] def
2328 CP 4 2 roll T moveto>
2329 \pst@def{PutEnd}<CP /lmtx [ lmtx aload pop setmatrix ] def moveto>

```

35.4 The real thing

`\begin@psput`, `\end@psput`

```

2330 \def\begin@psput#1{\begingroup\pst@killglue\leavevmode\pst@ifstar{#1}}%
2331 \def\end@psput#1(#2){%
2332 \pst@makebox{%
2333 \if@star
2334 \setbox\pst@hbox\hbox{\psframebox*[boxsep=false]{\unhbox\pst@hbox}}%
2335 \fi
2336 #1(#2)%
2337 \endgroup
2338 \ignorespaces}}

```

`\rput`

```

2339 \def\rput{\begin@psput{\pst@getref{\pst@getrputrot{\end@psput\rput@i}}}}
2340 \def\rput@i(#1){%

```



```

2341 \pst@makesmall\pst@hbox
2342 \ifx\pst@rot\@empty\else\pst@rotate\pst@hbox\fi
2343 \psput@{#1}\pst@hbox}

```

\cput

The first argument of \cput@iii is a hook used by node commands.

```

2344 \def\cput{\def\pst@par{}\pst@object{cput}}
2345 \def\cput@i{\begingroup\pst@killglue\leavevmode\pst@getrputrot\cput@ii}
2346 \def\cput@ii(#1){\pst@makebox{\cput@iii}{#1}}
2347 \def\cput@iii#1(#2){%
2348   \setbox\pst@hbox=\hbox{\psboxsepfalse\pscircularbox@ii{#1}}%
2349   \let\refpoint@x\relax
2350   \rput@i{#2}%
2351 \endgroup
2352 \ignorespaces}

```

36 \uput and company

The difference between \uput and \rput is that \rput's reference point is replaced by labelsep and reference angle arguments.

\psset@labelsep, \pslabelsep

```

2353 \newdimen\pslabelsep
2354 \def\psset@labelsep#1{\pssetlength\pslabelsep{#1}}
2355 \psset@labelsep{5pt}

```

\pst@getrefangle

```

2356 \def\pst@getrefangle#1\@nil{%
2357   \def\next##10#1=##2"##3@##4\@nil{%
2358     \ifx\relax##2%
2359       \pst@getangle{#1}\pst@refangle
2360       \def\pst@uputref{}%
2361     \else
2362       \edef\pst@refangle{##2}%
2363       \edef\pst@uputref{##3}%
2364     \fi}%
2365   \expandafter\next\pst@refangletable @#1=\relax"@@nil}

```

\pst@refangletable

```

2366 \def\pst@refangletable{%
2367   @r=0"20%
2368   @u=90"02%
2369   @l=180"10%
2370   @d=-90"01%
2371   @ur=45"22%
2372   @ul=135"12%
2373   @dr=-135"21%
2374   @dl=-45"11}

```

\uput

```

2375 \def\uput{\begin@psput{\@ifnextchar[{\uput@ii}{\uput@i}}}
2376 \def\uput@i#1{\pssetlength\pslabelsep{#1}\uput@ii}
2377 \def\uput@ii[#1]{%
2378   \pst@expandafter\pst@getrefangle{#1}\@nil
2379   \pst@getrputrot{\end@psput\uput@iii}}
2380 \def\uput@iii(#1){%
2381   \ifx\pst@uputref\@empty
2382     \uput@iv\tx@UUput
2383   \else
2384     \ifx\pst@rot\@empty
2385       \expandafter\uput@v\pst@uputref
2386     \else
2387       \uput@iv\tx@UUput
2388     \fi
2389   \fi
2390   \psput@{#1}\pst@hbox}
2391 \def\uput@iv#1{%
2392   \edef\pst@coor{%
2393     \pst@number\pslabelsep
2394     \pst@number{\wd\pst@hbox}%
2395     \pst@number{\ht\pst@hbox}%
2396     \pst@number{\dp\pst@hbox}%
2397     \pst@refangle\space \ifx\pst@rot\@empty\else\pst@rot\space sub \fi
2398     \tx@Uput #1}%
2399   \setbox\pst@hbox=\hbox to\z@{\hss\vbox to\z@{\vss\box\pst@hbox\vss}\hss}%
2400   \setbox\pst@hbox=\psput@special\pst@hbox
2401   \ifx\pst@rot\@empty\else\pst@rotate\pst@hbox\fi}
2402 \def\uput@v#1#2{%
2403   \ifnum#1>\z@\ifnum#2>\z@\pslabelsep=.707\pslabelsep\fi\fi
2404   \setbox\pst@hbox=\vbox to\z@{%
2405     \ifnum#2=1 \vskip\pslabelsep\else\vss\fi
2406     \hbox to\z@{%
2407       \ifnum#1=2 \hskip\pslabelsep\else\hss\fi
2408       \box\pst@hbox
2409       \ifnum#1=1 \hskip\pslabelsep\else\hss\fi}%
2410   \ifnum#2=2 \vskip\pslabelsep\else\vss\fi}}

```

\tx@Uput

I forget how this works, but it does.

```

2411 \pst@def{Uput}<%
2412 /a ED
2413 add 2 div /h ED
2414 2 div /w ED
2415 /s a sin def
2416 /c a cos def
2417 /b
2418 s abs c abs 2 copy gt
2419 dup /q ED
2420 { pop } { exch pop } ifelse
2421 def
2422 /w1 c b div w mul def
2423 /h1 s b div h mul def
2424 q

```

```

2425 { w1 abs w sub dup c mul abs }
2426 { h1 abs h sub dup s mul abs }
2427 ifelse>
2428 \pst@def{UUput}<%
2429 /z ED
2430 abs /y ED
2431 /x ED
2432 q
2433 { x s div c mul abs y gt }
2434 { x c div s mul abs y gt }
2435 ifelse
2436 { x x mul y y mul sub z z mul add sqrt z add }
2437 { q { x s div } { x c div } ifelse abs }
2438 ifelse
2439 a \tx@PtoC h1 add exch w1 add exch>

```

`\pst@getlabelsep, \Rput`

`\Rput` is an obsolete version of `\uput`.

```

2440 \def\pst@getlabelsep#1{%
2441   \@ifnextchar[%
2442     {\def\refpoint@x{.5}\def\refpoint@y{.5}\pst@@getref{#1}}%
2443     {\pst@@getlabelsep{#1}}}
2444 \def\pst@@getlabelsep#1#2{\pssetlength\pslabelsep{#2}\pst@getref{#1}}
2445 \def\Rput{%
2446   \begin@psput{\pst@getlabelsep{\pst@getrputrot{\end@psput{\Rput@i\rput@i}}}}
2447 \def\Rput@i{%
2448   \pst@dimg=\dp\pst@hbox
2449   \advance\pst@dimg\pslabelsep
2450   \dp\pst@hbox=\pst@dimg
2451   \pst@dimg=\ht\pst@hbox
2452   \advance\pst@dimg\pslabelsep
2453   \ht\pst@hbox=\pst@dimg
2454   \setbox\pst@hbox\hbox{\kern\pslabelsep\box\pst@hbox\kern\pslabelsep}}%

```

37 Pictures

`\pspicture`

```

2455 \def\pspicture{\begingroup\pst@ifstar\pst@picture}
2456 \def\pst@picture{%
2457   \@ifnextchar[{\pst@@picture}{\pst@@picture[0]}
2458 \def\pst@@picture[#1]#2(#3,#4){%
2459   \@ifnextchar({\pst@@@picture[#1](#3,#4)}%
2460   {\pst@@@picture[#1](0,0)(#3,#4)}}
2461 \def\pst@@@picture[#1]#2(#3,#4)(#4,#5){%
2462   \pssetxlength\pst@dima{#2}\pssetylength\pst@dimb{#3}%
2463   \pssetxlength\pst@dimc{#4}\pssetylength\pst@dimd{#5}%
2464   \def\pst@tempa{#1}%
2465   \setbox\pst@hbox=\hbox\bgroup
2466   \begingroup\KillGlue
2467   \@ifundefined{@latexerr}{\let\unitlength\psunit}}%

```

```

2468 \edef\pic@coord{(#2,#3)(#2,#3)(#4,#5)}\ignorespaces}
2469 \def\pic@coord{(0,0)(0,0)(10,10)}
2470 \def\endpspicture{%
2471   \pst@killglue
2472   \endgroup
2473   \egroup
2474   \ifdim\wd\pst@hbox=\z@\else
2475     \@pstrickserr{Extraneous space in the pspicture environment}%
2476     {Type \space <return> \space to procede.}%
2477   \fi
2478   \ht\pst@hbox=\pst@dimd
2479   \dp\pst@hbox=-\pst@dimb
2480   \setbox\pst@hbox=\hbox{%
2481     \kern-\pst@dima
2482     \ifx\pst@tempa\@empty\else
2483       \advance\pst@dimd-\pst@dimb
2484       \pst@dimd=\pst@tempa\pst@dimd
2485       \advance\pst@dimd\pst@dimb
2486       \lower\pst@dimd
2487     \fi
2488     \box\pst@hbox
2489     \kern\pst@dimc}%
2490   \if@star\setbox\pst@hbox=\hbox{\clipbox@@\z@}\fi
2491   \leavevmode\box\pst@hbox
2492 \endgroup}
2493 \@namedef{pspicture*}{\pspicture*}
2494 \@namedef{endpspicture*}{\endpspicture}

```

38 Overlays

Overlays work by translating invisible material. They take advantage of the fact that PostScript is running parallel to \TeX , and so we can redefine the value of some PostScript variables in order to get a different overlay printed each time we output a box containing overlay commands (even though the box has already been typeset by \TeX).

BeginOverlay

BeginOL is a PostScript procedure, with syntax:

```
(string) BeginOL
```

If the string is not (all) and does not match TheOL, then the output is made invisible by translating it over by the coffee pot (actually, by a distance OLUnit). Otherwise, it is made visible by translating it back to the page.

Rather than translating the page, we could define a small clipping path off the page, but that would be more likely to be messed up by someone's `initclip` (e.g., by `PSTricks' initclip!`).

```

2495 \pst@def{BeginOL}<%
2496   dup (all) eq exch TheOL eq or
2497   { IfVisible not
2498     { CP OLUnit T moveto
2499       /IfVisible true def }

```

```

2500     if }
2501   { IfVisible
2502     { CP OUnit \tx@NET moveto
2503       /IfVisible false def }
2504     if }
2505   ifelse>

```

InitOL

This figures out how far in the current units used by the driver is 50 inches up and to the right. This works even though drivers use unusual coordinate systems (even dvips). This macro also defines BOL to be BeginOL and sets the default value of IfVisible.

```

2506 \pst@ding=40in
2507 \edef\pst@OLunit{\pst@number\pst@ding}
2508 \pst@def{InitOL}<%
2509   /OLUnit [ gsave CM \tx@STV \pst@OLunit
2510     dup moveto setmatrix CP grestore ] cvx def
2511   /BOL { \tx@BeginOL } def /IfVisible true def>

```

\pst@initoverlay

This defines TheOL to be #1. It must be inserted just before printing overlay #1.

```

2512 \def\pst@initoverlay#1{\pst@Verb{\tx@InitOL /TheOL (#1) def}}

```

\pst@overlay, \pst@endoverlay

\pst@overlay just calls BeginOverlay.

```

2513 \def\pst@overlay#1{%
2514   \edef\curr@overlay{#1}%
2515   \pst@Verb{(#1) BOL}%
2516   \aftergroup\pst@endoverlay}
2517 \def\pst@endoverlay{%
2518   \pst@Verb{(\curr@overlay) BOL}}
2519 \def\curr@overlay{all}

```

\overlaybox, \endoverlaybox, \putoverlaybox

\pst@initoverlay, \pst@overlay, and \pst@endoverlay are the overlays primitives. An interface must be set up that guarantees that \pst@overlay and \pst@endoverlay are only used inside a box, and that \pst@initoverlay is inserted each type the box is printed. Here is one such interface (see seminar.sty for an interface for slides). The extra \begingroup and \endgroup assure that each \pst@endoverlay is executed within the box.

```

2520 \newbox\theoverlaybox
2521 \def\overlaybox{%
2522   \setbox\theoverlaybox=\hbox\bgroup
2523   \begingroup
2524   \let\psoverlay\pst@overlay
2525   \def\overlaybox{%
2526     \@pstrickserr{Overlays cannot be nested}\@eha}%
2527   \def\putoverlaybox{%

```

```

2528     \@pstrickserr{You must end the overlay box
2529     before using \string\putoverlaybox}}%
2530     \psoverlay{main}}
2531 \def\endoverlaybox{\endgroup\egroup}
2532 \def\putoverlaybox#1{%
2533   \hbox{\pst@initoverlay{#1}\copy\theoverlaybox}}
2534 \def\psoverlay{\@pstrickserr{\string\psoverlay\space
2535   can only be used after \string\overlaybox}}

```

39 Configuration file – revisited

```

2536 \ifx\pstcustomize\relax \input pstricks.con \fi
2537 \pst@ATH<end>
2538 \catcode'\@=\PstAtCode\relax
2539 \endinput

```

Part I

pst-node.doc

Check whether file has been loaded already.

```
2540 \csname PSTnodesLoaded\endcsname
2541 \let\PSTnodesLoaded\endinput

Load pstricks.tex if necessary:

2542 \ifx\PSTricksLoaded\endinput\else
2543   \def\next{\input pstricks.tex}\expandafter\next
2544 \fi

Take care of the catcode of @:

2545 \edef\TheAtCode{\the\catcode'\@}
2546 \catcode'\@=11
```

40 Node header

Nodes use the dictionary `tx@NodeDict`, which is always put on the stack after `tx@Dict`. `tx@NodeDict` should avoid using the same procedure names as are found in `tx@Dict`, especially those that do not use scratch variables and hence can be used without problem when `tx@NodeDict` is on top of the stack. When invoking a `tx@Dict` procedure that does use scratch variables, `tx@Dict` should be put on top.

`\pst@nodedict`

```
2547 \pst@ATH< \% Version \fileversion, \filedate.>
2548 \pst@ATH< \% For use with \pstdriver.>
2549 \pst@ATH</tx@NodeDict 200 dict def tx@NodeDict begin>
2550 \ifx\pst@useheader\iftrue
2551   \pstheader{pst-node.pro}
2552   \def\pst@nodedict{tx@NodeDict begin }
2553 \else
2554   \def\pst@nodedict{%
2555     /tx@NodeDict where
2556     { pop }
2557     { userdict begin /tx@NodeDict 200 dict def end }
2558     ifelse
2559     tx@NodeDict begin }
2560 \fi
```

41 Nodes

`\pst@getnode`

```
2561 % A node is a dictionary. To reduce the chance of errors, we check that the
2562 % name begins with a letter and does not contain any spaces.
2563 %   \begin{macrocode}
2564 \def\pst@getnode#1#2{%
2565   \pst@expandafter\pst@getnode{#1} * \@nil{#1}#2}
```

```

2566 \def\pst@getnode#1#2 #3\@nil#4#5{%
2567   \ifcat#1a\relax
2568     \def#5{/TheNode#1#2 }%
2569   \else
2570     \def#5{/BadNode }%
2571     \@pstrickserr{Bad node name: '#4'}\@ehpa
2572   \fi}

```

Before a node is defined, the coordinate system is scaled to PSTricks' standard coordinate system, with the origin at \TeX 's current point. The following objects should then be added to the node dictionary:

NodeMtrx The current matrix.

X The x-coordinate of the center.

Y The y-coordinate of the center.

NodePos A procedure that, given the values of **Sin**, **Cos**, and **Nodesep**, gives the relative position of the point that is distance **Nodesep** from the edge of the node, in the direction **(Cos,Sin)** from the center. "Relative" means relative to **(X,Y)** and for the coordinate system in effect when the node was defined.

`\tx@NewNode`, `\pst@newnode`

The node's dictionary size should be large enough for the 7 key's mentioned above, plus any keys the node needs for **NodePos**, plus a few more to avoid mistakes.

Syntax for **NewNode**

```
{beforenode'proc} /node'name dict'size {node'proc} NewNode"
```

`<beforenode_proc>` is stuff to be done with `tx@Dict`. It might leave things on the stack for use by `<node_proc>`.

Syntax for `\pst@newnode`:

```
\pst@newnode{node'name}{dict'size}{beforenode'proc}{node'proc}
```

```

2573 \pst@def{NewNode}<%
2574   gsave
2575     /next ED
2576     dict
2577     dup 3 -1 roll ED
2578     begin
2579       tx@Dict begin
2580         \tx@STV
2581         CP T
2582         exec
2583       end
2584       /NodeMtrx CM def
2585     next
2586   end
2587   grestore>
2588 \def\pst@newnode#1#2#3#4{%
2589   \leavevmode

```



```

2590 \pst@getnode{#1}\pst@thenode
2591 \pst@Verb{%
2592   \pst@nodedict
2593   { #3 } \pst@thenode #2 { #4 } \tx@NewNode
2594   end}}

\tx@InitPnode, \pnode
2595 \pst@def{InitPnode}<%
2596 /Y ED /X ED
2597 /NodePos { Nodesep Cos mul Nodesep Sin mul } def>
2598 \def\pnode{\@ifnextchar({\pnode@}{\pnode@(0,0)}}
2599 \def\pnode@(#1)#2{%
2600   \pst@getcoor{#1}%
2601   \pst@newnode{#2}{10}{\pst@coor}{\tx@InitPnode}%
2602   \ignorespaces}

\tx@InitCnode, \cnode
2603 \pst@def{InitCnode}<%
2604 /r ED /Y ED /X ED
2605 /NodePos { Nodesep r add dup Cos mul exch Sin mul } def>
2606 \def\cnode{\def\pst@par{}\pst@object{cnode}}
2607 \def\cnode@i{\@ifnextchar({\cnode@ii}{\cnode@ii(0,0)}}
2608 \def\cnode@ii(#1)#2#3{%
2609   \begingroup
2610     \use@par
2611     \pscircle@do(#1){#2}%
2612     \pst@getcoor{#1}%
2613     \pssetlength\pst@dimc{#2}%
2614     \pst@newnode{#3}{11}{%
2615       \pst@coor
2616       \pst@number\pst@dimc
2617       \pst@number\pslinewidth
2618       \psk@dimen .5 sub mul sub}%
2619     {\tx@InitCnode}%
2620   \endgroup
2621   \ignorespaces}
2622 \def\cnodeput{\def\pst@par{}\pst@object{cnodeput}}
2623 \def\cnodeput@i{%
2624   \begingroup
2625     \pst@killglue
2626     \leavevmode
2627     \pst@getrputrot
2628     \cnodeput@ii}
2629 \def\cnodeput@ii(#1)#2{%
2630   \pst@makebox{\cput@iii{\cnodeput@iii{#2}}{#1}}
2631   \def\cnodeput@iii#1{%
2632     \pst@newnode{#1}{11}{\pscirclebox@iv \pst@number\pslinewidth add}%
2633     {\tx@InitCnode}}

\circlenode
2634 \def\circlenode{\def\pst@par{}\pst@object{circlenode}}
2635 \def\circlenode@i#1{\pst@makebox{\pscirclebox@ii{\cnodeput@iii{#1}}}}

```

```

2636 \pst@def{GetRnodePos}<%
2637   Cos 0 gt
2638   { /dx r Nodesep add def }
2639   { /dx l Nodesep sub def }
2640   ifelse
2641   Sin 0 gt
2642   { /dy u Nodesep add def }
2643   { /dy d Nodesep sub def }
2644   ifelse
2645   dx Sin mul abs dy Cos mul abs gt
2646   { dy Cos mul Sin div dy }
2647   { dx dup Sin mul Cos \tx@Div }
2648   ifelse>

```

InitRnode

Syntax:

yref ht dp bool xref wd InitRnode

Additional keys: r, l, d, u, dx and dy.

```

2649 \pst@def{InitRnode}<%
2650   /r ED r mul neg /l ED /r r l add def
2651   /X l neg def
2652   { neg /d ED /u ED /Y 0 def }
2653   { neg /Y ED
2654     Y sub /u ED
2655     u mul neg /d ED
2656     /u u d add def
2657     /Y Y d sub def }
2658   ifelse
2659   /NodePos { \tx@GetRnodePos } def>

```

\rnode

The ability to set the refpoint is an undocumented feature that may be omitted.

```

2660 \def\rnode{\begingroup\pst@getref\rnode@}
2661 \def\rnode@#1{\pst@makebox{\rnode@@{#1}}}
2662 \def\rnode@@#1{%
2663   \ifx\refpoint@x\relax
2664     \def\refpoint@y{.5}%
2665     \def\refpoint@x{.5}%
2666   \fi
2667   \pst@newnode{#1}{16}{-}{%
2668     \ifx\refpoint@x\relax .5 \else \refpoint@y\space \fi
2669     \pst@number{\ht\pst@hbox}%
2670     \pst@number{\dp\pst@hbox}%
2671     \ifx\refpoint@y@empty true \else false \fi
2672     \refpoint@x\space
2673     \pst@number{\wd\pst@hbox}%
2674     \tx@InitRnode}%
2675   \box\pst@hbox
2676   \endgroup}

```

InitRNode

Syntax:

ht dp wd xref yref InitRNode

```
2677 \pst@def{InitRNode}<%
2678 /Y ED /X ED /r ED /X r 2 div X add def /r r X sub def /l X neg def
2679 Y add neg /d ED Y sub /u ED
2680 /NodePos { \tx@GetRnodePos } def>
```

\Rnode

```
2681 \def\Rnode{\@ifnextchar({\Rnode@}{\Rnode@(\RnodeRef)}}
2682 \def\Rnode@(#1)#2{\pst@makebox{\Rnode@@(#1){#2}}}
2683 \def\Rnode@@(#1)#2{%
2684 \begingroup
2685 \pst@getcoor{#1}%
2686 \pst@newnode{#2}{16}{%
2687 \pst@number{\ht\pst@hbox}\pst@number{\dp\pst@hbox}%
2688 \pst@number{\wd\pst@hbox}\pst@coor}{\tx@InitRNode}%
2689 \box\pst@hbox
2690 \endgroup}
2691 \def\RnodeRef{0,.7ex}
```

GetOnodePos

```
2692 \pst@def{GetOnodePos}<%
2693 /ww w Nodesep add def /hh h Nodesep add def
2694 Sin ww mul Cos hh mul \tx@Atan dup
2695 cos ww mul exch sin hh mul>
```

\ovalnode

Additional keys: w, h, ww, hh.

```
2696 \def\ovalnode{\def\pst@par{}\pst@object{ovalnode}}
2697 \def\ovalnode@i#1{\pst@makebox{\psovalbox@ii{\ovalnode@ii{#1}}}}
2698 \def\ovalnode@ii#1{%
2699 \pst@newnode{#1}{14}{-%
2700 /X \pst@number{\wd\pst@hbox}2 div def
2701 /Y \pst@number\pst@dimg 2 div \pst@number{\dp\pst@hbox}sub def
2702 /w \pst@number\pst@dima def
2703 /h \pst@number\pst@dimb def
2704 /NodePos { \tx@GetOnodePos } def}}
```

42 Node connections: Preliminaries

\tx@GetCenter, \tx@GetAngle

Syntax:

- GetCenter *x y* (Center coordinates)
- GetAngle *angle* (Angle from A to B)

```

2705 \pst@def{GetCenter}<begin X Y NodeMtrx transform CM itransform end>
2706 \pst@def{GetAngle}<%
2707   nodeA \tx@GetCenter
2708   nodeB \tx@GetCenter
2709   3 -1 roll sub 3 1 roll sub neg \tx@Atan>

```

`\tx@GetEdge, \tx@GetPos`

Syntax:

offset angle nodesep node GetEdge x y

GetPos defines (x1,y1) and (x2,y2) to be coordinates of position for node A and B, taking into account AngleA, AngleB, OffsetA, OffsetB, NodesepA and NodesepB.

```

2710 \pst@def{GetEdge}<%
2711   begin
2712     /Nodesep ED
2713     dup
2714     1 0 NodeMtrx dtransform CM idtransform exch atan sub
2715     dup sin /Sin ED cos /Cos ED
2716     NodePos Y add exch X add exch
2717     NodeMtrx transform CM itransform
2718   end % offset angle x y
2719   4 2 roll
2720 % Now add the offsets:
2721   1 index 0 eq
2722   { pop pop }
2723   { 2 copy 5 2 roll % x offset angle y offset angle
2724     cos mul add
2725     4 1 roll
2726     sin mul sub
2727     exch }
2728   ifelse>
2729 \pst@def{GetPos}<%
2730   OffsetA AngleA NodesepA nodeA \tx@GetEdge /y1 ED /x1 ED
2731   OffsetB AngleB NodesepB nodeB \tx@GetEdge /y2 ED /x2 ED>
2732 \def\check@arrow#1#2{%
2733   \check@@arrow#2-\@nil
2734   \if@pst
2735     \addto@par{arrows=#2}%
2736     \def\next{#1}%
2737   \else
2738     \def\next{#1{#2}}%
2739   \fi
2740   \next}
2741 \def\check@@arrow#1-#2\@nil{%
2742   \ifx\@nil#2\@nil\@pstfalse\else\@psttrue\fi}
2743 \pst@def{InitNC}<%
2744   /nodeB ED /nodeA ED
2745   /NodesepB ED /NodesepA ED
2746   /OffsetB ED /OffsetA ED
2747   tx@NodeDict nodeA known tx@NodeDict nodeB known and dup
2748   { /nodeA nodeA load def /nodeB nodeB load def } if>

```

```

2749 \def\nc@object#1#2#3#4{%
2750   \begin@OpenObj
2751     \showpointsfalse
2752     \pst@getnode{#1}\pst@tempa
2753     \pst@getnode{#2}\pst@tempb
2754     \gdef\lputpos@default{#3}%
2755     \addto@pscode{%
2756       \pst@nodedict
2757       \psk@offsetA
2758       \psk@offsetB neg
2759       \psk@nodesepA
2760       \psk@nodesepB
2761       \pst@tempa
2762       \pst@tempb
2763       \tx@InitNC { #4 } if
2764     end}%
2765     \def\use@pscode{%
2766       \pst@Verb{gsave \tx@STV newpath \pst@code\space grestore}%
2767       \gdef\pst@code{}}%
2768   \end@OpenObj}
2769 \def\lputpos@default{.5}

2770 \def\pc@object#1{%
2771   \@ifnextchar({\pc@@object#1}{\pst@getarrows{\pc@@object#1}})
2772 \def\pc@@object#1(#2)(#3){%
2773   \pnode{#2}{@@A}\pnode{#3}{@@B}%
2774   #1{@@A}{@@B}}

2775 \def\psset@nodesepA#1{\pst@getlength{#1}\psk@nodesepA}
2776 \def\psset@nodesepB#1{\pst@getlength{#1}\psk@nodesepB}
2777 \def\psset@nodesep#1{%
2778   \psset@nodesepA{#1}\let\psk@nodesepB\psk@nodesepA}
2779 \psset@nodesep{0}

\psset@offset, \psk@offsetA, \psk@offsetB

2780 \def\psset@offsetA#1{\pst@getlength{#1}\psk@offsetA}
2781 \def\psset@offsetB#1{\pst@getlength{#1}\psk@offsetA}
2782 \def\psset@offset#1{%
2783   \psset@offsetA{#1}\let\psk@offsetB\psk@offsetA}
2784 \psset@offset{0}

\psset@arm, \psk@armA, \psk@armB

2785 \def\psset@armA#1{\pst@getlength{#1}\psk@armA}
2786 \def\psset@armB#1{\pst@getlength{#1}\psk@armB}
2787 \def\psset@arm#1{\psset@armA{#1}\let\psk@armB\psk@armA}
2788 \psset@arm{10pt}

\psset@angle, \psk@angleA, \psk@angleB

2789 \def\psset@angleA#1{\pst@getangle{#1}\psk@angleA}
2790 \def\psset@angleB#1{\pst@getangle{#1}\psk@angleB}%
2791 \def\psset@angle#1{\pst@getangle{#1}\psk@angleA}
2792 \let\psk@angleB\psk@angleA}

```

```

2793 \psset@angle{0}

      \psset@arcangle, \psk@arcangleA, \psk@arcangleB
2794 \def\psset@arcangleA#1{\pst@getangle{#1}\psk@arcangleA}
2795 \def\psset@arcangleB#1{\pst@getangle{#1}\psk@arcangleB}%
2796 \def\psset@arcangle#1{\pst@getangle{#1}\psk@arcangleA
2797   \let\psk@arcangleB\psk@arcangleA}
2798 \psset@arcangle{8}

      \psset@ncurv, \psk@ncurvA, \psk@ncurvB
2799 \def\psset@ncurvA#1{\pst@checknum{#1}\psk@ncurvA}
2800 \def\psset@ncurvB#1{\pst@checknum{#1}\psk@ncurvB}%
2801 \def\psset@ncurv#1{\psset@ncurvA{#1}\let\psk@ncurvB\psk@ncurvA}
2802 \psset@ncurv{.67}

```

43 Node connections: The real thing

```

2803 \pst@def{LineMP}<%
2804   4 copy
2805   1 t sub mul exch t mul add 3 1 roll
2806   1 t sub mul exch t mul add exch 6 2 roll
2807   sub 3 1 roll sub \tx@Atan>

```

`\tx@NCCoor, \tx@NCLine`

Syntax:

OffsetB NodesepB OffsetA NodesepA NCLine

Leaves coordinates on stack rather than actually drawing line.

```

2808 \pst@def{NCCoor}<%
2809   \tx@GetAngle
2810   /AngleA ED /AngleB AngleA 180 add def
2811   \tx@GetPos
2812   /LPutVar [ x2 x1 y2 y1 ] cvx def
2813   /LPutPos { LPutVar \tx@LineMP } def
2814   x1 y1 x2 y2>
2815 \pst@def{NCLine}<%
2816   \tx@NCCoor
2817   tx@Dict begin
2818     ArrowB
2819     4 2 roll
2820     ArrowA
2821     lineto
2822   end>

2823 \def\ncline{\def\pst@par{\pst@object{ncline}}
2824 \def\ncline@i{\check@arrow{ncline@ii}}
2825 \def\ncline@ii#1#2{\nc@object{#1}{#2}{.5}}{\tx@NCLine}}

2826 \def\pcline{\def\pst@par{\pst@object{pcline}}
2827 \def\pcline@i{\pc@object\ncline@ii}

```

```

2828 \def\ncLine{\def\pst@par{ }\pst@object{ncLine}}
2829 \def\ncLine@i{\check@arrow{\ncLine@ii}}
2830 \def\ncLine@ii#1#2{\nc@object{#1}{#2}{.5}%
2831   {\tx@NCLine
2832   /LPutVar [
2833     nodeA \tx@GetCenter
2834     nodeB \tx@GetCenter
2835     3 1 roll 4 1 roll
2836   ] cvx def }}

2837 \pst@def{BezierMidpoint}<%
2838   /y3 ED /x3 ED
2839   /y2 ED /x2 ED
2840   /y1 ED /x1 ED
2841   /y0 ED /x0 ED
2842   /t ED
2843   /cx x1 x0 sub 3 mul def
2844   /cy y1 y0 sub 3 mul def
2845   /bx x2 x1 sub 3 mul cx sub def
2846   /by y2 y1 sub 3 mul cy sub def
2847   /ax x3 x0 sub cx sub bx sub def
2848   /ay y3 y0 sub cy sub by sub def
2849   ax t 3 exp mul bx t t mul mul add cx t mul add x0 add
2850   ay t 3 exp mul by t t mul mul add cy t mul add y0 add
2851   3 ay t t mul mul mul 2 by t mul mul add cy add
2852   3 ax t t mul mul mul 2 bx t mul mul add cx add
2853   atan>

2854 \pst@def{GetArms}<%
2855   /x1a armA AngleA cos mul x1 add def
2856   /y1a armA AngleA sin mul y1 add def
2857   /x2a armB AngleB cos mul x2 add def
2858   /y2a armB AngleB sin mul y2 add def>

2859 \pst@def{NCCurve}<%
2860   \tx@GetPos
2861   x1 x2 sub y1 y2 sub \tx@Pyth
2862   2 div dup
2863   3 -1 roll mul /armA ED mul /armB ED
2864   \tx@GetArms
2865   x1a y1a x1 y1 tx@Dict begin ArrowA end
2866   x2a y2a x2 y2 tx@Dict begin ArrowB end
2867   curveto
2868   /LPutVar [ x1 y1 x1a y1a x2a y2a x2 y2 ] cvx def
2869   /LPutPos { t LPutVar \tx@BezierMidpoint } def>

2870 \def\ncurve{\def\pst@par{ }\pst@object{ncurve}}
2871 \def\ncurve@i{\check@arrow{\ncurve@ii}}
2872 \def\ncurve@ii#1#2{\nc@object{#1}{#2}{.5}{%
2873   /AngleA \psk@angleA\space def /AngleB \psk@angleB\space def
2874   \psk@ncurvB\space \psk@ncurvA\space
2875   \tx@NCCurve}}

2876 \def\pccurve{\def\pst@par{ }\pst@object{pccurve}}
2877 \def\pccurve@i{\pc@object\ncurve@ii}

```

```

2878 \def\ncarc{\def\pst@par{}\pst@object{ncarc}}
2879 \def\ncarc@i{\check@arrow{\ncarc@ii}}
2880 \def\ncarc@ii#1#2{\nc@object{#1}{#2}{.5}}{
2881   \tx@GetAngle dup
2882   \psk@arcangleA\space add /AngleA ED
2883   \psk@arcangleB\space sub 180 add /AngleB ED
2884   \psk@ncurvB\space \psk@ncurvA\space
2885   \tx@NCCurve}}

2886 \def\pcarc{\def\pst@par{}\pst@object{pcarc}}
2887 \def\pcarc@i{\pc@object\ncarc@ii}

2888 \pst@def{AnglesMP}<%
2889   LPutVar
2890   t 3 gt
2891   { /t t 3 sub def }
2892   { t 2 gt
2893     { /t t 2 sub def 10 -2 roll }
2894     { t 1 gt
2895       { /t t 1 sub def 10 -4 roll }
2896       { 10 4 roll }
2897       ifelse }
2898     ifelse }
2899   ifelse
2900   6 { pop } repeat
2901   3 -1 roll exch \tx@LineMP>

2902 \pst@def{NCAngles}<%
2903   \tx@GetPos
2904   \tx@GetArms
2905   /mtrx AngleA matrix rotate def
2906   x1a y1a mtrx transform pop
2907   x2a y2a mtrx transform exch pop
2908   mtrx itransform
2909   /y0 ED /x0 ED
2910   mark
2911   armB 0 ne { x2 y2 } if x2a y2a x0 y0 x1a y1a armA 0 ne { x1 y1 } if
2912   tx@Dict begin false \tx@Line end
2913   /LPutVar [ x2 y2 x2a y2a x0 y0 x1a y1a x1 y1 ] cvx def
2914   /LPutPos { \tx@AnglesMP } def>

2915 \def\ncangles{\def\pst@par{}\pst@object{ncangles}}
2916 \def\ncangles@i{\check@arrow{\ncangles@ii}}
2917 \def\ncangles@ii#1#2{%
2918   \nc@object{#1}{#2}{1.5}}{\ncangles@iii \tx@NCAngles}}
2919 \def\ncangles@iii{%
2920   tx@Dict begin
2921   \ifdim\pslinearc>\z@
2922     /r \pst@number\pslinearc def
2923     /Lineto { \tx@Arcto } def
2924   \else
2925     /Lineto { L } def
2926   \fi
2927   end
2928   /AngleA \psk@angleA\space def /AngleB \psk@angleB\space def
2929   /armA \psk@armA\space def /armB \psk@armB\space def }

```



```

2930 \def\pcangles{\def\pst@par{}\pst@object{pcangles}}
2931 \def\pcangles@i{\pc@object\ncangles@ii}

2932 \pst@def{NCAngle}<%
2933   \tx@GetPos
2934   /x2a armB AngleB cos mul x2 add def
2935   /y2a armB AngleB sin mul y2 add def
2936   /mtrx AngleA matrix rotate def
2937   x2a y2a mtrx transform pop
2938   x1 y1 mtrx transform exch pop
2939   mtrx itransform
2940   /y0 ED /x0 ED
2941   mark
2942   armB 0 ne { x2 y2 } if x2a y2a x0 y0 x1 y1
2943   tx@Dict begin false \tx@Line end
2944   /LPutVar [ x2 y2 x2 y2 x2a y2a x0 y0 x1 y1 ] cvx def
2945   /LPutPos { \tx@AnglesMP } def>

2946 \def\ncangle{\def\pst@par{}\pst@object{ncangle}}
2947 \def\ncangle@i{\check@arrow{\ncangle@ii}}
2948 \def\ncangle@ii#1#2{%
2949   \nc@object{#1}{#2}{1.5}{\ncangles@iii \tx@NCAngle}}

2950 \def\pcangle{\def\pst@par{}\pst@object{pcangle}}
2951 \def\pcangle@i{\pc@object\ncangle@ii}

2952 \pst@def{NCBar}<%
2953   \tx@GetPos
2954   \tx@GetArms
2955   /mtrx AngleA matrix rotate def
2956   x1a y1a mtrx transform pop
2957   x2a y2a mtrx transform pop sub
2958   dup 0 mtrx itransform
2959   3 -1 roll 0 gt
2960   { /y2a exch y2a add def /x2a exch x2a add def }
2961   { /y1a exch neg y1a add def /x2a exch neg x2a add def }
2962   ifelse
2963   mark
2964   x2 y2 x2a y2a x1a y1a x1 y1
2965   tx@Dict begin false \tx@Line end
2966   /LPutVar [ x2 y2 x2 y2 x2a y2a x1a y1a x1 y1 ] cvx def
2967   /LPutPos { LPutVar \tx@AnglesMP } def>

2968 \def\ncbar{\def\pst@par{}\pst@object{ncbar}}
2969 \def\ncbar@i{\check@arrow{\ncbar@ii}}
2970 \def\ncbar@ii#1#2{\nc@object{#1}{#2}{1.5}{%
2971   \ncangles@iii /AngleB \psk@angleA def \tx@NCBar}}

2972 \def\pcbar{\def\pst@par{}\pst@object{pcbar}}
2973 \def\pcbar@i{\pc@object\ncbar@ii}

2974 \pst@def{NCDiag}<%
2975   \tx@GetPos
2976   \tx@GetArms
2977   mark
2978   x2 y2 x2a y2a x1a y1a x1 y1

```

```

2979   tx@Dict begin false \tx@Line end
2980   /LPutVar [ x2 y2 x2 y2 x2a y2a x1a y1a x1 y1 ] cvx def
2981   /LPutPos { \tx@AnglesMP } def>

2982 \def\ncdiag{\def\pst@par{}\pst@object{ncdiag}}
2983 \def\ncdiag@i{\check@arrow{\ncdiag@ii}}
2984 \def\ncdiag@ii#1#2{%
2985   \ncobject{#1}{#2}{1.5}{\ncangles@iii \tx@NCDiag}}

\pcdiag
2986 \def\pcdiag{\def\pst@par{}\pst@object{pcdiag}}
2987 \def\pcdiag@i{\pcobject\ncdiag@ii}

2988 \pst@def{NCDiagg}<%
2989   OffsetA AngleA NodesepA nodeA \tx@GetEdge
2990   /y1 ED /x1 ED
2991   /x1a armA AngleA cos mul x1 add def
2992   /y1a armA AngleA sin mul y1 add def
2993   nodeB \tx@GetCenter
2994   y1a sub exch x1a sub \tx@Atan 180 add /AngleB ED
2995   OffsetB AngleB NodesepB nodeB \tx@GetEdge
2996   /y2 ED /x2 ED
2997   mark
2998   x2 y2 x1a y1a x1 y1
2999   tx@Dict begin false \tx@Line end
3000   /LPutVar [ x2 y2 x2 y2 x2 y2 x1a y1a x1 y1] cvx def
3001   /LPutPos { \tx@AnglesMP } def>

3002 \def\ncdiagg{\def\pst@par{}\pst@object{ncdiagg}}
3003 \def\ncdiagg@i{\check@arrow{\ncdiagg@ii}}
3004 \def\ncdiagg@ii#1#2{%
3005   \ncobject{#1}{#2}{.5}{\ncangles@iii \tx@NCDiagg}}

\pcdiagg
3006 \def\pcdiagg{\def\pst@par{}\pst@object{pcdiagg}}
3007 \def\pcdiagg@i{\pcobject\ncdiagg@ii}

\tx@LoopMP
3008 \pst@def{LoopMP}<%
3009   /t t abs def
3010   [ LPutVar ] length 2 div 1 sub dup t lt { /t ED } { pop } ifelse
3011   mark LPutVar
3012   t cvi { /t t 1 sub def pop pop } repeat
3013   counttomark 1 add 4 roll clearmark
3014   3 -1 roll exch \tx@LineMP>

3015 \pst@def{NCLoop}<%
3016   \tx@GetPos
3017   \tx@GetArms
3018   /mtrx AngleA matrix rotate def
3019   x1a y1a mtrx transform loopsize add /y1b ED /x1b ED
3020   /x2b x2a y2a mtrx transform pop def
3021   x2b y1b mtrx itransform /y2b ED /x2b ED
3022   x1b y1b mtrx itransform /y1b ED /x1b ED

```

```

3023 mark
3024 armB 0 ne { x2 y2 } if x2a y2a x2b y2b x1b y1b x1a y1a armA
3025 0 ne { x1 y1 } if
3026 tx@Dict begin false \tx@Line end
3027 /LPutVar [ x2 y2 x2a y2a x2b y2b x1b y1b x1a y1a x1 y1 ] cvx def
3028 /LPutPos { \tx@LoopMP } def>

\psset@loopsize

3029 \def\psset@loopsize#1{\pst@getlength{#1}\psk@loopsize}
3030 \psset@loopsize{1cm}

3031 \def\nccloop{\def\pst@par{}\pst@object{nccloop}}
3032 \def\nccloop@i{\check@arrow{nccloop@ii}}
3033 \def\nccloop@ii#1#2{%
3034 \nc@object{#1}{#2}{2.5}%
3035 {\ncangles@iii /loopsize \psk@loopsize\space def \tx@NCCLoop}}

3036 \def\pcloop{\def\pst@par{}\pst@object{pcloop}}
3037 \def\pcloop@i{\pc@object\nccloop@ii}

\tx@NCCircle

3038 \pst@def{NCCircle}<%
3039 nodeA \tx@GetCenter
3040 0 0 NodesepA nodeA \tx@GetEdge
3041 % Stack: x-center y-center x-edge y-origin
3042 pop 3 1 roll
3043 /Y ED /X ED % center
3044 X sub 2 div % half distance to edge
3045 dup 2 exp r r mul sub abs sqrt atan 2 mul /a ED % angle to edge
3046 r AngleA 90 add \tx@PtoC % displacement to origin
3047 Y add exch X add exch % origin
3048 2 copy /LPutVar [ 4 2 roll r a ] def
3049 /LPutPos { LPutVar aload pop t 360 mul add dup 5 1 roll
3050 90 sub \tx@PtoC 3 -1 roll add 3 1 roll add exch 3 -1 roll } def
3051 r
3052 AngleA 90 sub a add % begin arc angle
3053 AngleA 270 add a sub % end arc angle
3054 % Stack: x0 y0 r a1 a2
3055 tx@Dict begin
3056 /angleB ED
3057 /angleA ED
3058 /r ED
3059 /c 57.2957 r \tx@Div def
3060 /y ED
3061 /x ED>

3062 \def\nccircle{\def\pst@par{}\pst@object{nccircle}}
3063 \def\nccircle@i{\check@arrow{nccircle@ii}}
3064 \def\nccircle@ii#1#2{%
3065 \pssetlength\pst@dima{#2}%
3066 \nc@object{#1}{#1}{.5}%
3067 /AngleA \psk@angleA def
3068 /r \pst@number\pst@dima def
3069 \tx@NCCircle \psarc@v end}}

```

44 Node Labels

```
\pst@getlputrot
3070 \def\pst@getlputrot#1{%
3071   \@ifnextchar(%
3072     {\def\pst@rot{#1}%
3073     {\pst@@getlputrot{\@ifnextchar({#1}{#1(\lputpos@default)}}}}
3074 \def\pst@@getlputrot#1#2{%
3075   \pst@expandafter{\@ifnextchar:{\pst@@@getlputrot}%
3076     {\@ifstar{\pst@@@getrot}{\pst@@getrot}}{#2}\@nil
3077   \ifx\pst@rotlist\@empty\else
3078     \edef\pst@rotlist{\pst@rotlist \pst@rot add }%
3079   \fi
3080   #1}
3081 \def\pst@@@getlputrot#1#2\@nil{%
3082   \pst@@getrot#2\@nil
3083   \edef\pst@rot{langle \ifx\pst@rot\@empty\else\pst@rot add \fi}}%

LPutCoor
3084 \pst@def{LPutCoor}<%
3085   tx@NodeDict /LPutPos known
3086   { gsave
3087     LPutPos
3088     tx@Dict begin
3089       /langle ED
3090       CM 3 1 roll
3091       \tx@STV
3092       CP 3 -1 roll sub neg 3 1 roll sub exch
3093       moveto
3094       setmatrix
3095       CP
3096     end
3097     grestore }
3098   { 0 0 tx@Dict /langle 0 def end }
3099   ifelse>

\psput@lput
3100 \def\psput@lput#1#2{%
3101   \pst@checknum{#1}\pst@tempa
3102   \hbox{%
3103     \pst@Verb{%
3104       \pst@nodedict
3105       /t \pst@tempa\space def
3106       \tx@LPutCoor
3107     end
3108     \tx@PutBegin}%
3109   \box#2%
3110   \pst@Verb{\tx@PutEnd}}

\lput
3111 \def\lput{\begin@psput{\pst@getref{\pst@getlputrot{\end@psput\lput@i}}}}
3112 \def\lput@i(#1){%
```

```

3113 \pst@makesmall\pst@hbox
3114 \ifx\pst@rot@\empty\else\pst@rotate\pst@hbox\fi
3115 \psput@lput{#1}\pst@hbox}

\mput

3116 \def\mput{%
3117 \begin@psput{\def\pst@rot{}}\pst@getref{\end@psput\lput@i(\lputpos@default)}}}

\aput, \Aput, \bput, \Bput

3118 \def\aput@#1{\begin@psput{%
3119 \def\pst@refangle{#1 }%
3120 \@ifnextchar[{\aput@i}{\pst@getlputrot{\end@psput\aput@ii}}}]
3121 \def\aput@i[#1]{%
3122 \pssetlength\pslabelsep{#1}\pst@getlputrot{\end@psput\aput@ii}}
3123 \def\aput@ii(#1){%
3124 \uput@iv\aput@iii
3125 \psput@lput{#1}\pst@hbox}
3126 \def\aput@iii{exch pop add a \tx@PtoC h1 add exch w1 add exch }
3127 \def\aput{\aput@{langle 90 add}}
3128 \def\bput{\aput@{langle 90 sub}}
3129 \def\Aput@#1{\begin@psput{%
3130 \def\pst@refangle{#1 }%
3131 \def\pst@rot{}}%
3132 \@ifnextchar[{\Aput@i}{\end@psput\aput@ii(\lputpos@default)}}}]
3133 \def\Aput@i[#1]{%
3134 \pssetlength\pslabelsep{#1}%
3135 \end@psput\aput@ii(\lputpos@default)}
3136 \def\Aput{\Aput@{langle 90 add}}
3137 \def\Bput{\Aput@{langle 90 sub}}

\Lput, \Mput

These are obsolete.

3138 \def\Lput{%
3139 \begin@psput{\pst@getlabelsep{\pst@getlputrot{\end@psput{\Rput@i\lput@i}}}}]
3140 \def\Mput{%
3141 \begin@psput{%
3142 \def\pst@rot{}}%
3143 \pst@getlabelsep{\end@psput{\Rput@i\lput@i}(\lputpos@default)}}}]

```

45 Node coordinates

```

\node@coor

3144 \def\node@coor#1;#2\@nil{%
3145 \pst@getnode{#1}\pst@tempg
3146 \edef\pst@coor{%
3147 \pst@nodedict
3148 tx@NodeDict \pst@tempg known
3149 { \pst@tempg load \tx@GetCenter }
3150 { 0 0 }
3151 ifelse

```

```

3152     end }}

\Node@coord
3153 \def\Node@coord[#1]#2;#3\@nil{%
3154   \begingroup
3155     \psset{#1}%
3156     \pst@getnode{#2}\pst@tempg
3157     \xdef\pst@tempg{%
3158       \pst@nodedict
3159       tx@NodeDict \pst@tempg known
3160       { \psk@offsetA \psk@angleA \psk@nodesepA \pst@tempg load \tx@GetEdge }
3161       { 0 0 }
3162       ifelse
3163     end }%
3164   \endgroup
3165   \let\pst@coord\pst@tempg}

3166 \pst@ATH<end>
3167 \catcode'\@=\TheAtCode\relax
3168 \endinput

```

Index

The **bold** numbers denote the pages where the entries are defined, and all other numbers indicate the *lines of code* where the entries are used.

| Symbols | | | |
|-------------------|----|-----------------|----|
| \@ehpa | 3 | \dotsize | 46 |
| \@ehpb | 3 | \dotstyle | 41 |
| \@ehpc | 3 | E | |
| \@newcolor | 11 | \Ellipse | 63 |
| \@none | 14 | \end@ClosedObj | 31 |
| \@pstrickserr | 3 | \end@CustomObj | 37 |
| | | \end@OpenObj | 31 |
| A | | \end@psput | 69 |
| \addto@par | 30 | \end@SpecialObj | 32 |
| \addto@pscode | 32 | \EndArrow | 26 |
| \altcolormode | 12 | \endclip | 57 |
| \AltCurve | 45 | \EndDot | 47 |
| \Aput | 90 | \endoverlaybox | 74 |
| \aput | 90 | \everypsbox | 54 |
| \Arcto | 42 | F | |
| \arrowsscale | 26 | \fillstyle | 25 |
| | | \Frame | 49 |
| B | | \framearc | 49 |
| \begin@AltOpenObj | 31 | \framesep | 56 |
| \begin@ClosedObj | 31 | G | |
| \begin@CustomObj | 37 | \getcoor@c | 16 |
| \begin@OpenObj | 31 | \GetOnodePos | 80 |
| \begin@psput | 69 | \gray | 13 |
| \begin@SpecialObj | 32 | \green | 13 |
| \BeginArrow | 26 | \Grid | 52 |
| \BeginOverlay | 73 | \gridcolor | 52 |
| \black | 13 | \griddots | 52 |
| \blue | 13 | \gridlabelcolor | 52 |
| \boxsep | 56 | \gridlabels | 52 |
| \Bput | 90 | \gridwidth | 51 |
| \bput | 90 | H | |
| | | \hatchangle | 23 |
| C | | \hatchcolor | 23 |
| \caddto@pscode | 38 | \hatchsep | 23 |
| \Cartesian | 18 | \hatchwidth | 23 |
| \circlenode | 78 | I | |
| \ClosedCurve | 45 | \if@star | 4 |
| \cnode | 78 | \ifpsdoubleline | 19 |
| \cornersize | 49 | \ifpsmathbox | 54 |
| \cput | 70 | \ifpsshadow | 19 |
| \curvature | 45 | \ifpsswapaxes | 19 |
| \cyan | 13 | \ifshowpoints | 19 |
| | | \init@pscode | 32 |
| D | | \InitOL | 74 |
| \darkgray | 13 | \InitRNode | 79 |
| \degrees | 16 | \InitRnode | 79 |
| \dimen | 50 | | |
| \dotangle | 47 | | |
| \dotsep | 21 | | |

| | | | | |
|----------------------|----------|--|---------------------------|----|
| | L | | \psas@oo | 48 |
| \lightgray | 13 | | \psbezier | 50 |
| \Line | 41 | | \psbordercolor | 19 |
| \linearc | 48 | | \psccurve | 46 |
| \Lput | 90 | | \pscircle | 62 |
| \lput | 89 | | \psclip | 57 |
| \LPutCoor | 89 | | \psclipbox | 58 |
| | | | \pscclosepath | 38 |
| | M | | \pscurve | 46 |
| \magenta | 13 | | \pscustom | 37 |
| \mixed@coor | 17 | | \psdblframebox | 57 |
| \Mput | 90 | | \psdots | 47 |
| \mput | 90 | | \psdoublecolor | 19 |
| \multips | 64 | | \psdoublesep | 19 |
| \multirput | 64 | | \psecurve | 46 |
| | | | \psellipse | 63 |
| | N | | \psfillcolor | 23 |
| \NArray | 41 | | \psframe | 50 |
| \newcmkcolor | 13 | | \psframebox | 56 |
| \newgray | 12 | | \psfs@crosshatch | 25 |
| \newsbcolor | 12 | | \psfs@hlines | 23 |
| \newsobject | 30 | | \psfs@none | 23 |
| \newsstyle | 14 | | \psfs@solid | 23 |
| \newrgbcolor | 12 | | \psfs@vlines | 24 |
| \Node@coor | 18, 91 | | \psgrid | 54 |
| \node@coor | 18, 90 | | \psgroup | 38 |
| \NormalCoor | 16 | | \psk@angleA | 82 |
| | | | \psk@angleB | 82 |
| | O | | \psk@arcangleA | 83 |
| \OpenCurve | 44 | | \psk@arcangleB | 83 |
| \ovalnode | 80 | | \psk@arcsepA | 60 |
| \overlaybox | 74 | | \psk@arcsepB | 60 |
| | | | \psk@armA | 82 |
| | P | | \psk@armB | 82 |
| \parabola | 51 | | \psk@arrowA | 25 |
| \pcdiag | 87 | | \psk@arrowB | 25 |
| \pcdiagg | 87 | | \psk@arrowinset | 27 |
| \pnode | 78 | | \psk@arrowlength | 27 |
| \Polar | 18 | | \psk@arrowsize | 27 |
| \polar@coor | 17 | | \psk@border | 19 |
| \Polygon | 42 | | \psk@bracketlength | 28 |
| \psaddtolength | 14 | | \psk@dash | 21 |
| \psarc | 61 | | \psk@ncurvA | 83 |
| \psarcn | 62 | | \psk@ncurvB | 83 |
| \psas@ | 29 | | \psk@offsetA | 82 |
| \psas@(..... | 29 | | \psk@offsetB | 82 |
| \psas@* | 48 | | \psk@origin | 18 |
| \psas@** | 48 | | \psk@rbracketlength | 29 |
| \psas@< | 28 | | \psk@shadowangle | 20 |
| \psas@<< | 28 | | \psk@shadowsize | 19 |
| \psas@] | 29 | | \psk@tbarsize | 28 |
| \psas@C | 29 | | \pslabelsep | 70 |
| \psas@c | 29 | | \pslbrace | 11 |
| \psas@cc | 29 | | \psline | 48 |
| \psas@o | 48 | | | |

| | | | |
|------------------------------------|----|--------------------------------|----|
| <code>\pslinecolor</code> | 20 | <code>\psset@xunit</code> | 15 |
| <code>\pslinetype</code> | 38 | <code>\psset@yunit</code> | 15 |
| <code>\pslinewidth</code> | 20 | <code>\pssetlength</code> | 14 |
| <code>\pslongbox</code> | 56 | <code>\pssetxlength</code> | 14 |
| <code>\psls@dashed</code> | 21 | <code>\pssetylength</code> | 14 |
| <code>\psls@dotted</code> | 22 | <code>\psshadowbox</code> | 58 |
| <code>\psls@none</code> | 20 | <code>\psshadowcolor</code> | 20 |
| <code>\psls@solid</code> | 21 | <code>\pst@@dimtonum</code> | 5 |
| <code>\psmove</code> | 38 | <code>\pst@@getlength</code> | 15 |
| <code>\psovalbox</code> | 60 | <code>\pst@activearrows</code> | 25 |
| <code>\pspicture</code> | 72 | <code>\pst@addarrowdef</code> | 35 |
| <code>\pspolygon</code> | 49 | <code>\pst@addborder</code> | 34 |
| <code>\psput@cartesian</code> | 69 | <code>\pst@angle</code> | 15 |
| <code>\psput@lput</code> | 89 | <code>\pst@angleunit</code> | 16 |
| <code>\psput@special</code> | 69 | <code>\pst@arrowdef</code> | 35 |
| <code>\psrawfile</code> | 39 | <code>\pst@arrowtable</code> | 25 |
| <code>\psrbrace</code> | 11 | <code>\pst@arrowtype</code> | 35 |
| <code>\psset</code> | 13 | <code>\pst@ATH</code> | 6 |
| <code>\psset@angle</code> | 82 | <code>\pst@checknum</code> | 9 |
| <code>\psset@arcangle</code> | 83 | <code>\pst@closedshadow</code> | 33 |
| <code>\psset@arcsep</code> | 60 | <code>\pst@color</code> | 11 |
| <code>\psset@arm</code> | 82 | <code>\pst@configerr</code> | 5 |
| <code>\psset@arrowinset</code> | 27 | <code>\pst@coor</code> | 15 |
| <code>\psset@arrowlength</code> | 27 | <code>\pst@coors</code> | 15 |
| <code>\psset@arrows</code> | 25 | <code>\pst@cp</code> | 38 |
| <code>\psset@arrowsize</code> | 27 | <code>\pst@def</code> | 6 |
| <code>\psset@border</code> | 19 | <code>\pst@dict</code> | 7 |
| <code>\psset@bordercolor</code> | 19 | <code>\pst@dimtonum</code> | 5 |
| <code>\psset@bracketlength</code> | 28 | <code>\pst@divide</code> | 5 |
| <code>\psset@dash</code> | 21 | <code>\pst@doublestroke</code> | 35 |
| <code>\psset@dotscale</code> | 47 | <code>\pst@endcolor</code> | 11 |
| <code>\psset@doublecolor</code> | 19 | <code>\pst@endoverlay</code> | 74 |
| <code>\psset@doubleline</code> | 19 | <code>\pst@expandafter</code> | 4 |
| <code>\psset@doublesep</code> | 19 | <code>\pst@fill</code> | 35 |
| <code>\psset@fillcolor</code> | 23 | <code>\pst@Getangle</code> | 47 |
| <code>\psset@labelsep</code> | 70 | <code>\pst@getangle</code> | 15 |
| <code>\psset@liftpen</code> | 38 | <code>\pst@getarrows</code> | 31 |
| <code>\psset@linecolor</code> | 20 | <code>\pst@getcolor</code> | 14 |
| <code>\psset@linestyle</code> | 22 | <code>\pst@getcoor</code> | 15 |
| <code>\psset@linetype</code> | 38 | <code>\pst@getcoors</code> | 15 |
| <code>\psset@linewidth</code> | 20 | <code>\pst@getdimnum</code> | 10 |
| <code>\psset@loopsize</code> | 88 | <code>\pst@getint</code> | 11 |
| <code>\psset@ncurv</code> | 83 | <code>\pst@getlabelsep</code> | 72 |
| <code>\psset@offset</code> | 82 | <code>\pst@getlength</code> | 15 |
| <code>\psset@origin</code> | 18 | <code>\pst@getlputrot</code> | 89 |
| <code>\psset@rbracketlength</code> | 29 | <code>\pst@getnode</code> | 76 |
| <code>\psset@shadow</code> | 19 | <code>\pst@getnumii</code> | 10 |
| <code>\psset@shadowangle</code> | 20 | <code>\pst@getnumiii</code> | 10 |
| <code>\psset@shadowcolor</code> | 20 | <code>\pst@getnumiv</code> | 10 |
| <code>\psset@shadowsize</code> | 19 | <code>\pst@getref</code> | 67 |
| <code>\psset@showpoints</code> | 19 | <code>\pst@getrefangle</code> | 70 |
| <code>\psset@swapaxes</code> | 19 | <code>\pst@getrot</code> | 68 |
| <code>\psset@tbarsize</code> | 28 | <code>\pst@getrputrot</code> | 68 |
| <code>\psset@unit</code> | 15 | <code>\pst@getscale</code> | 10 |

| | | | |
|---------------------------------------|----------|-----------------------------|----|
| <code>\pst@grestore</code> | 12 | <code>\rotateleft</code> | 66 |
| <code>\pst@ifstar</code> | 4 | <code>\rotateright</code> | 66 |
| <code>\pst@initoverlay</code> | 74 | <code>\RoundBracket</code> | 29 |
| <code>\pst@killglue</code> | 32 | <code>\Rput</code> | 72 |
| <code>\pst@linetype</code> | 20 | <code>\rput</code> | 69 |
| <code>\pst@longbox</code> | 55 | | |
| <code>\pst@makelongbox</code> | 55 | | |
| <code>\pst@makenotverbbox</code> | 54 | | |
| <code>\pst@makesmall</code> | 67 | | |
| <code>\pst@makeverbbox</code> | 55 | | |
| <code>\pst@misplaced</code> | 4 | | |
| <code>\pst@newnode</code> | 77 | | |
| <code>\pst@nodedict</code> | 76 | | |
| <code>\pst@number</code> | 9 | | |
| <code>\pst@object</code> | 30 | | |
| <code>\pst@openshadow</code> | 34 | | |
| <code>\pst@OpenShowPoints</code> | 36 | | |
| <code>\pst@oplineto</code> | 38 | | |
| <code>\pst@optcp</code> | 38 | | |
| <code>\pst@overlay</code> | 74 | | |
| <code>\pst@par</code> | 30 | | |
| <code>\pst@pyth</code> | 5 | | |
| <code>\pst@rawfile</code> | 39 | | |
| <code>\pst@refangletable</code> | 70 | | |
| <code>\pst@repeatarrows</code> | 36 | | |
| <code>\pst@rotate</code> | 68 | | |
| <code>\pst@rotable</code> | 68 | | |
| <code>\pst@setdoublesep</code> | 33 | | |
| <code>\pst@setrepeatarrowsflag</code> | 20 | | |
| <code>\pst@stroke</code> | 34 | | |
| <code>\pst@useboxpar</code> | 56 | | |
| <code>\pst@usecolor</code> | 12 | | |
| <code>\pst@useheader</code> | 6 | | |
| <code>\pst@Verb</code> | 8 | | |
| <code>\PSTricksOff</code> | 6 | | |
| <code>\psunit</code> | 14 | | |
| <code>\psverbboxfalse</code> | 55 | | |
| <code>\psverbboxtrue</code> | 55 | | |
| <code>\pswedge</code> | 63 | | |
| <code>\psxunit</code> | 14 | | |
| <code>\psyunit</code> | 14 | | |
| <code>\putoverlaybox</code> | 74 | | |
| | | | |
| | Q | | |
| <code>\qdisk</code> | 63 | | |
| <code>\qline</code> | 48 | | |
| | | | |
| | R | | |
| <code>\radians</code> | 16 | | |
| <code>\raw@coor</code> | 17 | | |
| <code>\red</code> | 13 | | |
| <code>\Rnode</code> | 80 | | |
| <code>\rnode</code> | 79 | | |
| <code>\rotatedown</code> | 66 | | |
| | | S | |
| | | <code>\scalebox</code> | 65 |
| | | <code>\scaleboxto</code> | 65 |
| | | <code>\SD</code> | 40 |
| | | <code>\solid@star</code> | 33 |
| | | <code>\special@angle</code> | 18 |
| | | <code>\SpecialCoor</code> | 16 |
| | | <code>\specialcoor</code> | 16 |
| | | <code>\subgridcolor</code> | 52 |
| | | <code>\subgriddiv</code> | 52 |
| | | <code>\subgriddots</code> | 52 |
| | | <code>\subgridwidth</code> | 52 |
| | | | |
| | | T | |
| | | <code>\Tbar</code> | 28 |
| | | <code>\tx@@Bracket</code> | 28 |
| | | <code>\tx@Arc</code> | 60 |
| | | <code>\tx@Arrow</code> | 27 |
| | | <code>\tx@Atan</code> | 8 |
| | | <code>\tx@BAC</code> | 44 |
| | | <code>\tx@BOC</code> | 44 |
| | | <code>\tx@Bracket</code> | 28 |
| | | <code>\tx@CC</code> | 43 |
| | | <code>\tx@CCA</code> | 43 |
| | | <code>\tx@DashLine</code> | 21 |
| | | <code>\tx@Div</code> | 8 |
| | | <code>\tx@DotLine</code> | 22 |
| | | <code>\tx@EAC</code> | 44 |
| | | <code>\tx@EOC</code> | 44 |
| | | <code>\tx@GetAngle</code> | 80 |
| | | <code>\tx@GetCenter</code> | 80 |
| | | <code>\tx@GetEdge</code> | 81 |
| | | <code>\tx@GetPos</code> | 81 |
| | | <code>\tx@IC</code> | 44 |
| | | <code>\tx@InitCnode</code> | 78 |
| | | <code>\tx@InitPnode</code> | 78 |
| | | <code>\tx@LineFill</code> | 23 |
| | | <code>\tx@LoopMP</code> | 87 |
| | | <code>\tx@MRestore</code> | 38 |
| | | <code>\tx@MSave</code> | 38 |
| | | <code>\tx@NAC</code> | 44 |
| | | <code>\tx@NC</code> | 44 |
| | | <code>\tx@NCCircle</code> | 88 |
| | | <code>\tx@NCCoor</code> | 83 |
| | | <code>\tx@NCLine</code> | 83 |
| | | <code>\tx@NET</code> | 8 |
| | | <code>\tx@NewNode</code> | 77 |
| | | <code>\tx@PathLength</code> | 8 |

| | | | |
|------------------|----|-------------------|----|
| \tx@PtoC | 8 | \use@par | 30 |
| \tx@Pyth | 8 | \use@pscode | 32 |
| \tx@Rot | 66 | | |
| \tx@Shadow | 33 | | |
| \tx@STP | 9 | | |
| \tx@STV | 9 | | |
| \tx@Uput | 71 | | |
| | | W | |
| | | \white | 13 |
| | | | |
| | | Y | |
| U | | | |
| \uput | 70 | \yellow | 13 |