

File I

Implementation

1 l3draw implementation

```

1 <*initex | package>
2 <@@=draw>
3 <*package>
4 \ProvidesExplPackage{l3draw}{2020-07-17}{}
5 {L3 Experimental core drawing support}
6 </package>
7 \RequirePackage { l3color }

```

1.1 Internal auxiliaries

```

\s__draw_mark Internal scan marks.
\s__draw_stop 8 \scan_new:N \s__draw_mark
               9 \scan_new:N \s__draw_stop

(End definition for \s__draw_mark and \s__draw_stop.)

```

```

\q__draw_recursion_tail Internal recursion quarks.
\q__draw_recursion_stop 10 \quark_new:N \q__draw_recursion_tail
                        11 \quark_new:N \q__draw_recursion_stop

(End definition for \q__draw_recursion_tail and \q__draw_recursion_stop.)

```

```

\__draw_if_recursion_tail_stop_do:Nn Functions to query recursion quarks.
                                   12 \__kernel_quark_new_test:N \__draw_if_recursion_tail_stop_do:Nn

(End definition for \__draw_if_recursion_tail_stop_do:Nn.)
    Everything else is in the sub-files!
13 </initex | package>

```

2 l3draw-boxes implementation

```

14 <*initex | package>
15 <@@=draw>

```

Inserting boxes requires us to “interrupt” the drawing state, so is closely linked to scoping. At the same time, there are a few additional features required to make text work in a flexible way.

```

\l__draw_tmp_box
16 \box_new:N \l__draw_tmp_box

(End definition for \l__draw_tmp_box.)

```

`\draw_box_use:N` Before inserting a box, we need to make sure that the bounding box is being updated correctly. As drawings track transformations as a whole, rather than as separate operations, we do the insertion using an almost-row matrix. The process is split into two so that coffins are also supported.

```

17 \cs_new_protected:Npn \draw_box_use:N #1
18 {
19   \__draw_box_use:Nnnnn #1
20   { Opt } { -\box_dp:N #1 } { \box_wd:N #1 } { \box_ht:N #1 }
21 }
22 \cs_new_protected:Npn \__draw_box_use:Nnnnn #1#2#3#4#5
23 {
24   \bool_if:NT \l_draw_bb_update_bool
25   {
26     \__draw_point_process:nn
27     { \__draw_path_update_limits:nn }
28     { \draw_point_transform:n { #2 , #3 } }
29     \__draw_point_process:nn
30     { \__draw_path_update_limits:nn }
31     { \draw_point_transform:n { #4 , #3 } }
32     \__draw_point_process:nn
33     { \__draw_path_update_limits:nn }
34     { \draw_point_transform:n { #4 , #5 } }
35     \__draw_point_process:nn
36     { \__draw_path_update_limits:nn }
37     { \draw_point_transform:n { #2 , #5 } }
38   }
39   \group_begin:
40   \hbox_set:Nn \l__draw_tmp_box
41   {
42     \use:x
43     {
44       \__draw_backend_box_use:Nnnnn #1
45       { \fp_use:N \l__draw_matrix_a_fp }
46       { \fp_use:N \l__draw_matrix_b_fp }
47       { \fp_use:N \l__draw_matrix_c_fp }
48       { \fp_use:N \l__draw_matrix_d_fp }
49     }
50   }
51   \hbox_set:Nn \l__draw_tmp_box
52   {
53     \tex_kern:D \l__draw_xshift_dim
54     \box_move_up:nn { \l__draw_yshift_dim }
55     { \box_use_drop:N \l__draw_tmp_box }
56   }
57   \box_set_ht:Nn \l__draw_tmp_box { Opt }
58   \box_set_dp:Nn \l__draw_tmp_box { Opt }
59   \box_set_wd:Nn \l__draw_tmp_box { Opt }
60   \box_use_drop:N \l__draw_tmp_box
61   \group_end:
62 }

```

(End definition for `\draw_box_use:N` and `__draw_box_use:Nnnnn`. This function is documented on page ??.)

`\draw_coffin_use:Nnn` Slightly more than a shortcut: we have to allow for the fact that coffins have no apparent width before the reference point.

```

63 \cs_new_protected:Npn \draw_coffin_use:Nnn #1#2#3
64 {
65   \group_begin:
66     \hbox_set:Nn \l__draw_tmp_box
67     { \coffin_typeset:Nnnnn #1 {#2} {#3} { Opt } { Opt } }
68     \__draw_box_use:Nnnnn \l__draw_tmp_box
69     { \box_wd:N \l__draw_tmp_box - \coffin_wd:N #1 }
70     { -\box_dp:N \l__draw_tmp_box }
71     { \box_wd:N \l__draw_tmp_box }
72     { \box_ht:N \l__draw_tmp_box }
73   \group_end:
74 }

```

(End definition for `\draw_coffin_use:Nnn`. This function is documented on page ??.)

```

75 </initex | package>

```

3 l3draw-layers implementation

```

76 <*initex | package>

```

```

77 <@@=draw>

```

3.1 User interface

`\draw_layer_new:n`

```

78 \cs_new_protected:Npn \draw_layer_new:n #1
79 {
80   \str_if_eq:nnTF {#1} { main }
81   { \msg_error:nnn { draw } { main-reserved } }
82   {
83     \box_new:c { g__draw_layer_ #1 _box }
84     \box_new:c { l__draw_layer_ #1 _box }
85   }
86 }

```

(End definition for `\draw_layer_new:n`. This function is documented on page ??.)

`\l__draw_layer_tl` The name of the current layer: we start off with `main`.

```

87 \tl_new:N \l__draw_layer_tl
88 \tl_set:Nn \l__draw_layer_tl { main }

```

(End definition for `\l__draw_layer_tl`.)

`\l__draw_layer_close_bool` Used to track if a layer needs to be closed.

```

89 \bool_new:N \l__draw_layer_close_bool

```

(End definition for `\l__draw_layer_close_bool`.)

`\l_draw_layers_clist` The list of layers to use starts off with just the `main` one.

```

\g__draw_layers_clist
90 \clist_new:N \l_draw_layers_clist
91 \clist_set:Nn \l_draw_layers_clist { main }
92 \clist_new:N \g__draw_layers_clist

```

(End definition for `\l_draw_layers_clist` and `\g__draw_layers_clist`. This variable is documented on page ??.)

`\draw_layer_begin:n` Layers may be called multiple times and have to work when nested. That drives a bit of grouping to get everything in order. Layers have to be zero width, so they get set as we go along.

```

93 \cs_new_protected:Npn \draw_layer_begin:n #1
94 {
95   \group_begin:
96   \box_if_exist:cTF { g__draw_layer_ #1 _box }
97   {
98     \str_if_eq:VnTF \l__draw_layer_tl {#1}
99     { \bool_set_false:N \l__draw_layer_close_bool }
100    {
101      \bool_set_true:N \l__draw_layer_close_bool
102      \tl_set:Nn \l__draw_layer_tl {#1}
103      \box_gset_wd:cn { g__draw_layer_ #1 _box } { Opt }
104      \hbox_gset:cw { g__draw_layer_ #1 _box }
105      \box_use_drop:c { g__draw_layer_ #1 _box }
106      \group_begin:
107    }
108    \draw_linewidth:n { \l_draw_default_linewidth_dim }
109  }
110  {
111    \str_if_eq:nnTF {#1} { main }
112    { \msg_error:nnn { draw } { unknown-layer } {#1} }
113    { \msg_error:nnn { draw } { main-layer } }
114  }
115 }
116 \cs_new_protected:Npn \draw_layer_end:
117 {
118   \bool_if:NT \l__draw_layer_close_bool
119   {
120     \group_end:
121     \hbox_gset_end:
122   }
123   \group_end:
124 }
```

(End definition for `\draw_layer_begin:n` and `\draw_layer_end:`. These functions are documented on page ??.)

3.2 Internal cross-links

`__draw_layers_insert:` The main layer is special, otherwise just dump the layer box inside a scope.

```

125 \cs_new_protected:Npn \__draw_layers_insert:
126 {
127   \clist_map_inline:Nn \l_draw_layers_clist
128   {
129     \str_if_eq:nnTF {##1} { main }
130     {
131       \box_set_wd:Nn \l__draw_layer_main_box { Opt }
132       \box_use_drop:N \l__draw_layer_main_box
133     }
134 }
```

```

134         {
135             \__draw_backend_scope_begin:
136             \box_gset_wd:cn { g__draw_layer_ ##1 _box } { Opt }
137             \box_use_drop:c { g__draw_layer_ ##1 _box }
138             \__draw_backend_scope_end:
139         }
140     }
141 }

```

(End definition for __draw_layers_insert:.)

__draw_layers_save: Simple save/restore functions.

```

\__draw_layers_restore:
142 \cs_new_protected:Npn \__draw_layers_save:
143 {
144     \clist_map_inline:Nn \l_draw_layers_clist
145     {
146         \str_if_eq:nnF {##1} { main }
147         {
148             \box_set_eq:cc { l__draw_layer_ ##1 _box }
149             { g__draw_layer_ ##1 _box }
150         }
151     }
152 }
153 \cs_new_protected:Npn \__draw_layers_restore:
154 {
155     \clist_map_inline:Nn \l_draw_layers_clist
156     {
157         \str_if_eq:nnF {##1} { main }
158         {
159             \box_gset_eq:cc { g__draw_layer_ ##1 _box }
160             { l__draw_layer_ ##1 _box }
161         }
162     }
163 }

```

(End definition for __draw_layers_save: and __draw_layers_restore:.)

```

164 \msg_new:nnnn { draw } { main-layer }
165 { Material~cannot~be~added~to~'main'~layer. }
166 { The~main~layer~may~only~be~accessed~at~the~top~level. }
167 \msg_new:nnn { draw } { main-reserved }
168 { The~'main'~layer~is~reserved. }
169 \msg_new:nnnn { draw } { unknown-layer }
170 { Layer~'#1'~has~not~been~created. }
171 { You~have~tried~to~use~layer~'#1',~but~it~was~never~set~up. }
172 % \end{macrocode}
173 %
174 % \begin{macrocode}
175 </initex | package>

```

4 l3draw-paths implementation

```

176 <*initex | package>
177 <@@=draw>

```

This sub-module covers more-or-less the same ideas as `pgfcorepathconstruct.code.tex`, though using the expandable FPU means that the implementation often varies. At present, equivalents of the following are currently absent:

- `\pgfpatharcto`, `\pgfpatharctoprecomputed`: These are extremely specialised and are very complex in implementation. If the functionality is required, it is likely that it will be set up from scratch here.
- `\pgfpathparabola`: Seems to be unused other than defining a TikZ interface, which itself is then not used further.
- `\pgfpathsine`, `\pgfpathcosine`: Need to see exactly how these need to work, in particular whether a wider input range is needed and what approximation to make.
- `\pgfpathcurvebetweentime`, `\pgfpathcurvebetweentimecontinue`: These don't seem to be used at all.

`\l__draw_path_tmp_tl` Scratch space.

```

\l__draw_path_tmpa_fp 178 \tl_new:N \l__draw_path_tmp_tl
\l__draw_path_tmpb_fp 179 \fp_new:N \l__draw_path_tmpa_fp
180 \fp_new:N \l__draw_path_tmpb_fp

```

(End definition for `\l__draw_path_tmp_tl`, `\l__draw_path_tmpa_fp`, and `\l__draw_path_tmpb_fp`.)

4.1 Tracking paths

`\g__draw_path_lastx_dim` The last point visited on a path.

```

\g__draw_path_lasty_dim 181 \dim_new:N \g__draw_path_lastx_dim
182 \dim_new:N \g__draw_path_lasty_dim

```

(End definition for `\g__draw_path_lastx_dim` and `\g__draw_path_lasty_dim`.)

`\g__draw_path_xmax_dim` The limiting size of a path.

```

\g__draw_path_xmin_dim 183 \dim_new:N \g__draw_path_xmax_dim
\g__draw_path_ymax_dim 184 \dim_new:N \g__draw_path_xmin_dim
\g__draw_path_ymin_dim 185 \dim_new:N \g__draw_path_ymax_dim
186 \dim_new:N \g__draw_path_ymin_dim

```

(End definition for `\g__draw_path_xmax_dim` and others.)

`__draw_path_update_limits:nn` Track the limits of a path and (perhaps) of the picture as a whole. (At present the latter is always true: that will change as more complex functionality is added.)

```

\__draw_path_reset_limits: 187 \cs_new_protected:Npn \__draw_path_update_limits:nn #1#2
188 {
189   \dim_gset:Nn \g__draw_path_xmax_dim
190     { \dim_max:nn \g__draw_path_xmax_dim {#1} }
191   \dim_gset:Nn \g__draw_path_xmin_dim
192     { \dim_min:nn \g__draw_path_xmin_dim {#1} }
193   \dim_gset:Nn \g__draw_path_ymax_dim
194     { \dim_max:nn \g__draw_path_ymax_dim {#2} }
195   \dim_gset:Nn \g__draw_path_ymin_dim
196     { \dim_min:nn \g__draw_path_ymin_dim {#2} }
197   \bool_if:NT \l_draw_bb_update_bool
198     {
199     \dim_gset:Nn \g__draw_xmax_dim

```

```

200         { \dim_max:nn \g__draw_xmax_dim {#1} }
201         \dim_gset:Nn \g__draw_xmin_dim
202         { \dim_min:nn \g__draw_xmin_dim {#1} }
203         \dim_gset:Nn \g__draw_ymax_dim
204         { \dim_max:nn \g__draw_ymax_dim {#2} }
205         \dim_gset:Nn \g__draw_ymin_dim
206         { \dim_min:nn \g__draw_ymin_dim {#2} }
207     }
208 }
209 \cs_new_protected:Npn \__draw_path_reset_limits:
210 {
211     \dim_gset:Nn \g__draw_path_xmax_dim { -\c_max_dim }
212     \dim_gset:Nn \g__draw_path_xmin_dim { \c_max_dim }
213     \dim_gset:Nn \g__draw_path_ymax_dim { -\c_max_dim }
214     \dim_gset:Nn \g__draw_path_ymin_dim { \c_max_dim }
215 }

```

(End definition for __draw_path_update_limits:nn and __draw_path_reset_limits:.)

__draw_path_update_last:nn A simple auxiliary to avoid repetition.

```

216 \cs_new_protected:Npn \__draw_path_update_last:nn #1#2
217 {
218     \dim_gset:Nn \g__draw_path_lastx_dim {#1}
219     \dim_gset:Nn \g__draw_path_lasty_dim {#2}
220 }

```

(End definition for __draw_path_update_last:nn.)

4.2 Corner arcs

At the level of path *construction*, rounded corners are handled by inserting a marker into the path: that is then picked up once the full path is constructed. Thus we need to set up the appropriate data structures here, such that this can be applied every time it is relevant.

\l__draw_corner_xarc_dim The two arcs in use.

```

\l__draw_corner_yarc_dim
221 \dim_new:N \l__draw_corner_xarc_dim
222 \dim_new:N \l__draw_corner_yarc_dim

```

(End definition for \l__draw_corner_xarc_dim and \l__draw_corner_yarc_dim.)

\l__draw_corner_arc_bool A flag to speed up the repeated checks.

```

223 \bool_new:N \l__draw_corner_arc_bool

```

(End definition for \l__draw_corner_arc_bool.)

\draw_path_corner_arc:nn Calculate the arcs, check they are non-zero.

```

224 \cs_new_protected:Npn \draw_path_corner_arc:nn #1#2
225 {
226     \dim_set:Nn \l__draw_corner_xarc_dim {#1}
227     \dim_set:Nn \l__draw_corner_yarc_dim {#2}
228     \bool_lazy_and:nnTF
229     { \dim_compare_p:nNn \l__draw_corner_xarc_dim = { 0pt } }
230     { \dim_compare_p:nNn \l__draw_corner_yarc_dim = { 0pt } }

```

```

231     { \bool_set_false:N \l__draw_corner_arc_bool }
232     { \bool_set_true:N \l__draw_corner_arc_bool }
233   }

```

(End definition for \draw_path_corner_arc:nn. This function is documented on page ??.)

__draw_path_mark_corner: Mark up corners for arc post-processing.

```

234 \cs_new_protected:Npn \__draw_path_mark_corner:
235 {
236   \bool_if:NT \l__draw_corner_arc_bool
237   {
238     \__draw_softpath_roundpoint:VW
239     \l__draw_corner_xarc_dim
240     \l__draw_corner_yarc_dim
241   }
242 }

```

(End definition for __draw_path_mark_corner:.)

4.3 Basic path constructions

\draw_path_moveto:n At present, stick to purely linear transformation support and skip the soft path business:
\draw_path_lineto:n that will likely need to be revisited later.

```

\__draw_path_moveto:nn 243 \cs_new_protected:Npn \draw_path_moveto:n #1
\__draw_path_lineto:nn 244 {
\draw_path_curveto:nnn 245   \__draw_point_process:nn
\__draw_path_curveto:nnnnnn 246   { \__draw_path_moveto:nn }
247   { \draw_point_transform:n {#1} }
248 }
249 \cs_new_protected:Npn \__draw_path_moveto:nn #1#2
250 {
251   \__draw_path_update_limits:nn {#1} {#2}
252   \__draw_softpath_moveto:nn {#1} {#2}
253   \__draw_path_update_last:nn {#1} {#2}
254 }
255 \cs_new_protected:Npn \draw_path_lineto:n #1
256 {
257   \__draw_point_process:nn
258   { \__draw_path_lineto:nn }
259   { \draw_point_transform:n {#1} }
260 }
261 \cs_new_protected:Npn \__draw_path_lineto:nn #1#2
262 {
263   \__draw_path_mark_corner:
264   \__draw_path_update_limits:nn {#1} {#2}
265   \__draw_softpath_lineto:nn {#1} {#2}
266   \__draw_path_update_last:nn {#1} {#2}
267 }
268 \cs_new_protected:Npn \draw_path_curveto:nnn #1#2#3
269 {
270   \__draw_point_process:nnnn
271   {
272     \__draw_path_mark_corner:
273     \__draw_path_curveto:nnnnnn

```



```

274     }
275     { \draw_point_transform:n {#1} }
276     { \draw_point_transform:n {#2} }
277     { \draw_point_transform:n {#3} }
278   }
279   \cs_new_protected:Npn \__draw_path_curveto:nnnnnn #1#2#3#4#5#6
280   {
281     \__draw_path_update_limits:nn {#1} {#2}
282     \__draw_path_update_limits:nn {#3} {#4}
283     \__draw_path_update_limits:nn {#5} {#6}
284     \__draw_softpath_curveto:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
285     \__draw_path_update_last:nn {#5} {#6}
286   }

```

(End definition for `\draw_path_moveto:n` and others. These functions are documented on page ??.)

`\draw_path_close:` A simple wrapper.

```

287   \cs_new_protected:Npn \draw_path_close:
288   {
289     \__draw_path_mark_corner:
290     \__draw_softpath_closepath:
291   }

```

(End definition for `\draw_path_close:`. This function is documented on page ??.)

4.4 Canvas path constructions

`\draw_path_canvas_moveto:n` Operations with no application of the transformation matrix.

```

\draw_path_canvas_lineto:n
\draw_path_canvas_curveto:nnn
292   \cs_new_protected:Npn \draw_path_canvas_moveto:n #1
293   { \__draw_point_process:nn { \__draw_path_moveto:nn } {#1} }
294   \cs_new_protected:Npn \draw_path_canvas_lineto:n #1
295   { \__draw_point_process:nn { \__draw_path_lineto:nn } {#1} }
296   \cs_new_protected:Npn \draw_path_canvas_curveto:nnn #1#2#3
297   {
298     \__draw_point_process:nnnn
299     {
300       \__draw_path_mark_corner:
301       \__draw_path_curveto:nnnnnn
302     }
303     {#1} {#2} {#3}
304   }

```

(End definition for `\draw_path_canvas_moveto:n`, `\draw_path_canvas_lineto:n`, and `\draw_path_canvas_curveto:nnn`. These functions are documented on page ??.)

4.5 Computed curves

More complex operations need some calculations. To assist with those, various constants are pre-defined.

`\draw_path_curveto:nn` A quadratic curve with one control point (x_c, y_c) . The two required control points are then

$$x_1 = \frac{1}{3}x_s + \frac{2}{3}x_c \quad y_1 = \frac{1}{3}y_s + \frac{2}{3}y_c$$

`__draw_path_curveto:nnnn`
`\c__draw_path_curveto_a_fp`
`\c__draw_path_curveto_b_fp`

and

$$x_2 = \frac{1}{3}x_e + \frac{2}{3}x_c \quad x_2 = \frac{1}{3}y_e + \frac{2}{3}y_c$$

using the start (last) point (x_s, y_s) and the end point (x_s, y_s) .

```

305 \cs_new_protected:Npn \draw_path_curveto:nn #1#2
306 {
307   \__draw_point_process:nnn
308   { \__draw_path_curveto:nnnn }
309   { \draw_point_transform:n {#1} }
310   { \draw_point_transform:n {#2} }
311 }
312 \cs_new_protected:Npn \__draw_path_curveto:nnnn #1#2#3#4
313 {
314   \fp_set:Nn \l__draw_path_tmpa_fp { \c__draw_path_curveto_b_fp * #1 }
315   \fp_set:Nn \l__draw_path_tmpb_fp { \c__draw_path_curveto_b_fp * #2 }
316   \use:x
317   {
318     \__draw_path_mark_corner:
319     \__draw_path_curveto:nnnnnn
320     {
321       \fp_to_dim:n
322       {
323         \c__draw_path_curveto_a_fp * \g__draw_path_lastx_dim
324         + \l__draw_path_tmpa_fp
325       }
326     }
327     {
328       \fp_to_dim:n
329       {
330         \c__draw_path_curveto_a_fp * \g__draw_path_lasty_dim
331         + \l__draw_path_tmpb_fp
332       }
333     }
334     {
335       \fp_to_dim:n
336       { \c__draw_path_curveto_a_fp * #3 + \l__draw_path_tmpa_fp }
337     }
338     {
339       \fp_to_dim:n
340       { \c__draw_path_curveto_a_fp * #4 + \l__draw_path_tmpb_fp }
341     }
342     {#3}
343     {#4}
344   }
345 }
346 \fp_const:Nn \c__draw_path_curveto_a_fp { 1 / 3 }
347 \fp_const:Nn \c__draw_path_curveto_b_fp { 2 / 3 }

```

(End definition for `\draw_path_curveto:nn` and others. This function is documented on page ??.)

<pre> \draw_path_arc:nnn \draw_path_arc:nnnn __draw_path_arc:nnnn __draw_path_arc:nnNnn _draw_path_arc_auxi:nnnnNnn _draw_path_arc_auxi:fnnnNnn _draw_path_arc_auxi:fnfnNnn _draw_path_arc_auxii:nnnNnnnn __draw_path_arc_auxiii:nn __draw_path_arc_auxiv:nnnn __draw_path_arc_auxv:nn __draw_path_arc_auxvi:nn __draw_path_arc_add:nnnn \l__draw_path_arc_delta_fp </pre>	<p>Drawing an arc means dividing the total curve required into sections: using Bézier curves we can cover at most 90° at once. To allow for later manipulations, we aim to have roughly equal last segments to the line, with the split set at a final part of 115°.</p> <pre> 348 \cs_new_protected:Npn \draw_path_arc:nnn #1#2#3 </pre>
---	---

```

349 { \draw_path_arc:nnnn {#1} {#2} {#3} {#3} }
350 \cs_new_protected:Npn \draw_path_arc:nnnn #1#2#3#4
351 {
352   \use:x
353   {
354     \__draw_path_arc:nnnn
355     { \fp_eval:n {#1} }
356     { \fp_eval:n {#2} }
357     { \fp_to_dim:n {#3} }
358     { \fp_to_dim:n {#4} }
359   }
360 }
361 \cs_new_protected:Npn \__draw_path_arc:nnnn #1#2#3#4
362 {
363   \fp_compare:nNnTF {#1} > {#2}
364   { \__draw_path_arc:nnNnn {#1} {#2} - {#3} {#4} }
365   { \__draw_path_arc:nnNnn {#1} {#2} + {#3} {#4} }
366 }
367 \cs_new_protected:Npn \__draw_path_arc:nnNnn #1#2#3#4#5
368 {
369   \fp_set:Nn \l__draw_path_arc_start_fp {#1}
370   \fp_set:Nn \l__draw_path_arc_delta_fp { abs( #1 - #2 ) }
371   \fp_while_do:nNnn { \l__draw_path_arc_delta_fp } > { 90 }
372   {
373     \fp_compare:nNnTF \l__draw_path_arc_delta_fp > { 115 }
374     {
375       \__draw_path_arc_auxi:ffnnNnn
376       { \fp_to_decimal:N \l__draw_path_arc_start_fp }
377       { \fp_eval:n { \l__draw_path_arc_start_fp #3 90 } }
378       { 90 } {#2}
379       #3 {#4} {#5}
380     }
381     {
382       \__draw_path_arc_auxi:ffnnNnn
383       { \fp_to_decimal:N \l__draw_path_arc_start_fp }
384       { \fp_eval:n { \l__draw_path_arc_start_fp #3 60 } }
385       { 60 } {#2}
386       #3 {#4} {#5}
387     }
388   }
389   \__draw_path_mark_corner:
390   \__draw_path_arc_auxi:fnfnNnn
391   { \fp_to_decimal:N \l__draw_path_arc_start_fp }
392   {#2}
393   { \fp_eval:n { abs( \l__draw_path_arc_start_fp - #2 ) } }
394   {#2}
395   #3 {#4} {#5}
396 }

```

The auxiliary is responsible for calculating the required points. The “magic” number required to determine the length of the control vectors is well-established for a right-angle: $\frac{4}{3}(\sqrt{2} - 1) = 0.552\,284\,75$. For other cases, we follow the calculation used by pgf but with the second common case of 60° pre-calculated for speed.

```

397 \cs_new_protected:Npn \__draw_path_arc_auxi:nnnnNnn #1#2#3#4#5#6#7

```

```

398 {
399   \use:x
400   {
401     \__draw_path_arc_auxii:nnnNnnnn
402     {#1} {#2} {#4} #5 {#6} {#7}
403     {
404       \fp_to_dim:n
405       {
406         \cs_if_exist_use:cF
407         { c__draw_path_arc_ #3 _fp }
408         { 4/3 * tand( 0.25 * #3 ) }
409         * #6
410       }
411     }
412     {
413       \fp_to_dim:n
414       {
415         \cs_if_exist_use:cF
416         { c__draw_path_arc_ #3 _fp }
417         { 4/3 * tand( 0.25 * #3 ) }
418         * #7
419       }
420     }
421   }
422 }
423 \cs_generate_variant:Nn \__draw_path_arc_auxi:nnnnNnn { fnf , ff }

```

We can now calculate the required points. As everything here is non-expandable, that is best done by using x-type expansion to build up the tokens. The three points are calculated out-of-order, since finding the second control point needs the position of the end point. Once the points are found, fire-off the fundamental path operation and update the record of where we are up to. The final point has to be

```

424 \cs_new_protected:Npn \__draw_path_arc_auxii:nnnNnnnn #1#2#3#4#5#6#7#8
425 {
426   \tl_clear:N \l__draw_path_tmp_tl
427   \__draw_point_process:nn
428   { \__draw_path_arc_auxiii:nn }
429   {
430     \__draw_point_transform_noshift:n
431     { \draw_point_polar:nnn {#7} {#8} { #1 #4 90 } }
432   }
433   \__draw_point_process:nnn
434   { \__draw_path_arc_auxiv:nnnn }
435   {
436     \draw_point_transform:n
437     { \draw_point_polar:nnn {#5} {#6} {#1} }
438   }
439   {
440     \draw_point_transform:n
441     { \draw_point_polar:nnn {#5} {#6} {#2} }
442   }
443   \__draw_point_process:nn
444   { \__draw_path_arc_auxv:nn }
445   {

```

```

446     \__draw_point_transform_noshift:n
447     { \draw_point_polar:nnn {#7} {#8} { #2 #4 -90 } }
448   }
449   \exp_after:wN \__draw_path_curveto:nnnnnn \l__draw_path_tmp_tl
450   \fp_set:Nn \l__draw_path_arc_delta_fp { abs ( #2 - #3 ) }
451   \fp_set:Nn \l__draw_path_arc_start_fp {#2}
452 }

```

The first control point.

```

453 \cs_new_protected:Npn \__draw_path_arc_auxiii:nn #1#2
454 {
455   \__draw_path_arc_aux_add:nn
456   { \g__draw_path_lastx_dim + #1 }
457   { \g__draw_path_lasty_dim + #2 }
458 }

```

The end point: simple arithmetic.

```

459 \cs_new_protected:Npn \__draw_path_arc_auxiv:nnnn #1#2#3#4
460 {
461   \__draw_path_arc_aux_add:nn
462   { \g__draw_path_lastx_dim - #1 + #3 }
463   { \g__draw_path_lasty_dim - #2 + #4 }
464 }

```

The second control point: extract the last point, do some rearrangement and record.

```

465 \cs_new_protected:Npn \__draw_path_arc_auxv:nn #1#2
466 {
467   \exp_after:wN \__draw_path_arc_auxvi:nn
468   \l__draw_path_tmp_tl {#1} {#2}
469 }
470 \cs_new_protected:Npn \__draw_path_arc_auxvi:nn #1#2#3#4#5#6
471 {
472   \tl_set:Nn \l__draw_path_tmp_tl { {#1} {#2} }
473   \__draw_path_arc_aux_add:nn
474   { #5 + #3 }
475   { #6 + #4 }
476   \tl_put_right:Nn \l__draw_path_tmp_tl { {#3} {#4} }
477 }
478 \cs_new_protected:Npn \__draw_path_arc_aux_add:nn #1#2
479 {
480   \tl_put_right:Nx \l__draw_path_tmp_tl
481   { { \fp_to_dim:n {#1} } { \fp_to_dim:n {#2} } }
482 }
483 \fp_new:N \l__draw_path_arc_delta_fp
484 \fp_new:N \l__draw_path_arc_start_fp
485 \fp_const:cn { c__draw_path_arc_90_fp } { 4/3 * (sqrt(2) - 1) }
486 \fp_const:cn { c__draw_path_arc_60_fp } { 4/3 * tand(15) }

```

(End definition for \draw_path_arc:nnn and others. These functions are documented on page ??.)

\draw_path_arc_axes:nnnn A simple wrapper.

```

487 \cs_new_protected:Npn \draw_path_arc_axes:nnnn #1#2#3#4
488 {
489   \draw_transform_triangle:nnn { 0cm , 0cm } {#3} {#4}
490   \draw_path_arc:nnn {#1} {#2} { 1pt }
491 }

```

(End definition for \draw_path_arc_axes:nnnn. This function is documented on page ??.)

```

\draw_path_ellipse:nnn Drawing an ellipse is an optimised version of drawing an arc, in particular reusing the
\__draw_path_ellipse:nnnnnn same constant. We need to deal with the ellipse in four parts and also deal with moving
  \_draw_path_ellipse_arci:nnnnnn to the right place, closing it and ending up back at the center. That is handled on a
  \_draw_path_ellipse_arcii:nnnnnn per-arc basis, each in a separate auxiliary for readability.
  \_draw_path_ellipse_arciiii:nnnnnn
  \_draw_path_ellipse_arciv:nnnnnn
\c__draw_path_ellipse_fp
492 \cs_new_protected:Npn \draw_path_ellipse:nnn #1#2#3
493 {
494   \__draw_point_process:nnnn
495   { \__draw_path_ellipse:nnnnnn }
496   { \draw_point_transform:n {#1} }
497   { \__draw_point_transform_noshift:n {#2} }
498   { \__draw_point_transform_noshift:n {#3} }
499 }
500 \cs_new_protected:Npn \__draw_path_ellipse:nnnnnn #1#2#3#4#5#6
501 {
502   \use:x
503   {
504     \__draw_path_moveto:nn
505     { \fp_to_dim:n { #1 + #3 } } { \fp_to_dim:n { #2 + #4 } }
506     \__draw_path_ellipse_arci:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
507     \__draw_path_ellipse_arcii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
508     \__draw_path_ellipse_arciiii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
509     \__draw_path_ellipse_arciv:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
510   }
511   \__draw_softpath_closepath:
512   \__draw_path_moveto:nn {#1} {#2}
513 }
514 \cs_new:Npn \__draw_path_ellipse_arci:nnnnnn #1#2#3#4#5#6
515 {
516   \__draw_path_curveto:nnnnnn
517   { \fp_to_dim:n { #1 + #3 + #5 * \c__draw_path_ellipse_fp } }
518   { \fp_to_dim:n { #2 + #4 + #6 * \c__draw_path_ellipse_fp } }
519   { \fp_to_dim:n { #1 + #3 * \c__draw_path_ellipse_fp + #5 } }
520   { \fp_to_dim:n { #2 + #4 * \c__draw_path_ellipse_fp + #6 } }
521   { \fp_to_dim:n { #1 + #5 } }
522   { \fp_to_dim:n { #2 + #6 } }
523 }
524 \cs_new:Npn \__draw_path_ellipse_arcii:nnnnnn #1#2#3#4#5#6
525 {
526   \__draw_path_curveto:nnnnnn
527   { \fp_to_dim:n { #1 - #3 * \c__draw_path_ellipse_fp + #5 } }
528   { \fp_to_dim:n { #2 - #4 * \c__draw_path_ellipse_fp + #6 } }
529   { \fp_to_dim:n { #1 - #3 + #5 * \c__draw_path_ellipse_fp } }
530   { \fp_to_dim:n { #2 - #4 + #6 * \c__draw_path_ellipse_fp } }
531   { \fp_to_dim:n { #1 - #3 } }
532   { \fp_to_dim:n { #2 - #4 } }
533 }
534 \cs_new:Npn \__draw_path_ellipse_arciiii:nnnnnn #1#2#3#4#5#6
535 {
536   \__draw_path_curveto:nnnnnn
537   { \fp_to_dim:n { #1 - #3 - #5 * \c__draw_path_ellipse_fp } }
538   { \fp_to_dim:n { #2 - #4 - #6 * \c__draw_path_ellipse_fp } }
539   { \fp_to_dim:n { #1 - #3 * \c__draw_path_ellipse_fp - #5 } }

```

```

540     { \fp_to_dim:n { #2 - #4 * \c__draw_path_ellipse_fp - #6 } }
541     { \fp_to_dim:n { #1 - #5 } }
542     { \fp_to_dim:n { #2 - #6 } }
543   }
544   \cs_new:Npn \__draw_path_ellipse_arciv:nnnnnn #1#2#3#4#5#6
545   {
546     \__draw_path_curveto:nnnnnn
547     { \fp_to_dim:n { #1 + #3 * \c__draw_path_ellipse_fp - #5 } }
548     { \fp_to_dim:n { #2 + #4 * \c__draw_path_ellipse_fp - #6 } }
549     { \fp_to_dim:n { #1 + #3 - #5 * \c__draw_path_ellipse_fp } }
550     { \fp_to_dim:n { #2 + #4 - #6 * \c__draw_path_ellipse_fp } }
551     { \fp_to_dim:n { #1 + #3 } }
552     { \fp_to_dim:n { #2 + #4 } }
553   }
554   \fp_const:Nn \c__draw_path_ellipse_fp { \fp_use:c { c__draw_path_arc_90_fp } }

```

(End definition for \draw_path_ellipse:nnn and others. This function is documented on page ??.)

\draw_path_circle:nn A shortcut.

```

555   \cs_new_protected:Npn \draw_path_circle:nn #1#2
556   { \draw_path_ellipse:nnn {#1} { #2 , 0pt } { 0pt , #2 } }

```

(End definition for \draw_path_circle:nn. This function is documented on page ??.)

4.6 Rectangles

\draw_path_rectangle:nn Building a rectangle can be a single operation, or for rounded versions will involve step-by-step construction.

__draw_path_rectangle:nnnn

__draw_path_rectangle_rounded:nnnn

```

557   \cs_new_protected:Npn \draw_path_rectangle:nn #1#2
558   {
559     \__draw_point_process:nnn
560     {
561       \bool_lazy_or:nnTF
562       { \l__draw_corner_arc_bool }
563       { \l__draw_matrix_active_bool }
564       { \__draw_path_rectangle_rounded:nnnn }
565       { \__draw_path_rectangle:nnnn }
566     }
567     { \draw_point_transform:n {#1} }
568     {#2}
569   }
570   \cs_new_protected:Npn \__draw_path_rectangle:nnnn #1#2#3#4
571   {
572     \__draw_path_update_limits:nn {#1} {#2}
573     \__draw_path_update_limits:nn { #1 + #3 } { #2 + #4 }
574     \__draw_softpath_rectangle:nnnn {#1} {#2} {#3} {#4}
575     \__draw_path_update_last:nn {#1} {#2}
576   }
577   \cs_new_protected:Npn \__draw_path_rectangle_rounded:nnnn #1#2#3#4
578   {
579     \draw_path_moveto:n { #1 + #3 , #2 + #4 }
580     \draw_path_lineto:n { #1 , #2 + #4 }
581     \draw_path_lineto:n { #1 , #2 }
582     \draw_path_lineto:n { #1 + #3 , #2 }

```

```

583 \draw_path_close:
584 \draw_path_moveto:n { #1 , #2 }
585 }

```

(End definition for `\draw_path_rectangle:nn`, `__draw_path_rectangle:nnnn`, and `__draw_path_rectangle_rounded:nnnn`. This function is documented on page ??.)

```

\draw_path_rectangle_corners:nn
\__draw_path_rectangle_corners:nnnn

```

Another shortcut wrapper.

```

586 \cs_new_protected:Npn \draw_path_rectangle_corners:nn #1#2
587 {
588   \__draw_point_process:nnn
589   { \__draw_path_rectangle_corners:nnnnn {#1} }
590   {#1} {#2}
591 }
592 \cs_new_protected:Npn \__draw_path_rectangle_corners:nnnnn #1#2#3#4#5
593 { \draw_path_rectangle:nn {#1} { #4 - #2 , #5 - #3 } }

```

(End definition for `\draw_path_rectangle_corners:nn` and `__draw_path_rectangle_corners:nnnn`. This function is documented on page ??.)

4.7 Grids

```

\draw_path_grid:nnnn
\__draw_path_grid_auxi:nnnnnn
\__draw_path_grid_auxi:ffnnnn
\__draw_path_grid_auxii:nnnnnn
\__draw_path_grid_auxiii:nnnnnn
\__draw_path_grid_auxiiii:ffnnnn
\__draw_path_grid_auxiv:nnnnnnnn
\__draw_path_grid_auxiv:ffnnnnnn

```

The main complexity here is lining up the grid correctly. To keep it simple, we tidy up the argument ordering first.

```

594 \cs_new_protected:Npn \draw_path_grid:nnnn #1#2#3#4
595 {
596   \__draw_point_process:nnn
597   {
598     \__draw_path_grid_auxi:ffnnnn
599     { \dim_eval:n { \dim_abs:n {#1} } }
600     { \dim_eval:n { \dim_abs:n {#2} } }
601   }
602   {#3} {#4}
603 }
604 \cs_new_protected:Npn \__draw_path_grid_auxi:nnnnnn #1#2#3#4#5#6
605 {
606   \dim_compare:nNnTF {#3} > {#5}
607   { \__draw_path_grid_auxii:nnnnnn {#1} {#2} {#5} {#4} {#3} {#6} }
608   { \__draw_path_grid_auxii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6} }
609 }
610 \cs_generate_variant:Nn \__draw_path_grid_auxi:nnnnnn { ff }
611 \cs_new_protected:Npn \__draw_path_grid_auxii:nnnnnn #1#2#3#4#5#6
612 {
613   \dim_compare:nNnTF {#4} > {#6}
614   { \__draw_path_grid_auxiii:nnnnnn {#1} {#2} {#3} {#6} {#5} {#4} }
615   { \__draw_path_grid_auxiii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6} }
616 }
617 \cs_new_protected:Npn \__draw_path_grid_auxiii:nnnnnn #1#2#3#4#5#6
618 {
619   \__draw_path_grid_auxiv:ffnnnnnn
620   { \fp_to_dim:n { #1 * trunc(#{3}/(#{1})) } }
621   { \fp_to_dim:n { #2 * trunc(#{4}/(#{2})) } }
622   {#1} {#2} {#3} {#4} {#5} {#6}
623 }

```



```

624 \cs_new_protected:Npn \__draw_path_grid_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
625 {
626   \dim_step_inline:nnnn
627   {#1}
628   {#3}
629   {#7}
630   {
631     \draw_path_moveto:n { ##1 , #6 }
632     \draw_path_lineto:n { ##1 , #8 }
633   }
634   \dim_step_inline:nnnn
635   {#2}
636   {#4}
637   {#8}
638   {
639     \draw_path_moveto:n { #5 , ##1 }
640     \draw_path_lineto:n { #7 , ##1 }
641   }
642 }
643 \cs_generate_variant:Nn \__draw_path_grid_auxiv:nnnnnnnn { ff }

```

(End definition for `\draw_path_grid:nnnn` and others. This function is documented on page ??.)

4.8 Using paths

Actions to pass to the driver.

```

\l__draw_path_use_clip_bool
\l__draw_path_use_fill_bool
\l__draw_path_use_stroke_bool
644 \bool_new:N \l__draw_path_use_clip_bool
645 \bool_new:N \l__draw_path_use_fill_bool
646 \bool_new:N \l__draw_path_use_stroke_bool

```

(End definition for `\l__draw_path_use_clip_bool`, `\l__draw_path_use_fill_bool`, and `\l__draw_path_use_stroke_bool`.)

Actions handled at the macro layer.

```

\l__draw_path_use_bb_bool
\l__draw_path_use_clear_bool
647 \bool_new:N \l__draw_path_use_bb_bool
648 \bool_new:N \l__draw_path_use_clear_bool

```

(End definition for `\l__draw_path_use_bb_bool` and `\l__draw_path_use_clear_bool`.)

There are a range of actions which can apply to a path: they are handled in a single function which can carry out several of them. The first step is to deal with the special case of clearing the path.

```

\draw_path_use:n
\draw_path_use_clear:n
\__draw_path_use:n
\__draw_path_use_action_draw:
\__draw_path_use_action_fillstroke:
\__draw_path_use_stroke_bb:
\__draw_path_use_stroke_bb_aux:NnN
649 \cs_new_protected:Npn \draw_path_use:n #1
650 {
651   \tl_if_blank:nF {#1}
652   { \__draw_path_use:n {#1} }
653 }
654 \cs_new_protected:Npn \draw_path_use_clear:n #1
655 {
656   \bool_lazy_or:nnTF
657   { \tl_if_blank_p:n {#1} }
658   { \str_if_eq_p:nn {#1} { clear } }
659   {
660     \__draw_softpath_clear:

```

```

661     \__draw_path_reset_limits:
662   }
663   { \__draw_path_use:n { #1 , clear } }
664 }

```

Map over the actions and set up the data: mainly just booleans, but with the possibility to cover more complex cases. The business end of the function is a series of checks on the various flags, then taking the appropriate action(s).

```

665 \cs_new_protected:Npn \__draw_path_use:n #1
666 {
667   \bool_set_false:N \l__draw_path_use_clip_bool
668   \bool_set_false:N \l__draw_path_use_fill_bool
669   \bool_set_false:N \l__draw_path_use_stroke_bool
670   \clist_map_inline:nn {#1}
671   {
672     \cs_if_exist:cTF { l__draw_path_use_ ##1 _ bool }
673     { \bool_set_true:c { l__draw_path_use_ ##1 _ bool } }
674     {
675       \cs_if_exist_use:cF { __draw_path_use_action_ ##1 : }
676       { \msg_error:nnn { draw } { invalid-path-action } {##1} }
677     }
678   }
679   \__draw_softpath_round_corners:
680   \bool_lazy_and:nnT
681   { \l_draw_bb_update_bool }
682   { \l__draw_path_use_stroke_bool }
683   { \__draw_path_use_stroke_bb: }
684   \__draw_softpath_use:
685   \bool_if:NT \l__draw_path_use_clip_bool
686   {
687     \__draw_backend_clip:
688     \bool_set_false:N \l_draw_bb_update_bool
689     \bool_lazy_or:nnF
690     { \l__draw_path_use_fill_bool }
691     { \l__draw_path_use_stroke_bool }
692     { \__draw_backend_discardpath: }
693   }
694   \bool_lazy_or:nnT
695   { \l__draw_path_use_fill_bool }
696   { \l__draw_path_use_stroke_bool }
697   {
698     \use:c
699     {
700       __draw_backend_
701       \bool_if:NT \l__draw_path_use_fill_bool { fill }
702       \bool_if:NT \l__draw_path_use_stroke_bool { stroke }
703       :
704     }
705   }
706   \bool_if:NT \l__draw_path_use_clear_bool
707   { \__draw_softpath_clear: }
708 }
709 \cs_new_protected:Npn \__draw_path_use_action_draw:
710 {

```

```

711     \bool_set_true:N \l__draw_path_use_stroke_bool
712   }
713   \cs_new_protected:Npn \__draw_path_use_action_fillstroke:
714   {
715     \bool_set_true:N \l__draw_path_use_fill_bool
716     \bool_set_true:N \l__draw_path_use_stroke_bool
717   }

```

Where the path is relevant to size and is stroked, we need to allow for the part which overlaps the edge of the bounding box.

```

718   \cs_new_protected:Npn \__draw_path_use_stroke_bb:
719   {
720     \__draw_path_use_stroke_bb_aux:NnN x { max } +
721     \__draw_path_use_stroke_bb_aux:NnN y { max } +
722     \__draw_path_use_stroke_bb_aux:NnN x { min } -
723     \__draw_path_use_stroke_bb_aux:NnN y { min } -
724   }
725   \cs_new_protected:Npn \__draw_path_use_stroke_bb_aux:NnN #1#2#3
726   {
727     \dim_compare:nNnF { \dim_use:c { g__draw_ #1#2 _dim } } = { #3 -\c_max_dim }
728     {
729       \dim_gset:cn { g__draw_ #1#2 _dim }
730       {
731         \use:c { dim_ #2 :nn }
732         { \dim_use:c { g__draw_ #1#2 _dim } }
733         {
734           \dim_use:c { g__draw_path_ #1#2 _dim }
735           #3 0.5 \g__draw_linewidth_dim
736         }
737       }
738     }
739   }

```

(End definition for `\draw_path_use:n` and others. These functions are documented on page ??.)

4.9 Scoping paths

`\l__draw_path_lastx_dim` Local storage for global data. There is already a `\l__draw_softpath_main_tl` for path manipulation, so we can reuse that (it is always grouped when the path is being reconstructed).

```

\l__draw_path_xmax_dim
\l__draw_path_xmin_dim
\l__draw_path_ymax_dim
\l__draw_path_ymin_dim
\l__draw_softpath_corners_bool
740 \dim_new:N \l__draw_path_lastx_dim
741 \dim_new:N \l__draw_path_lasty_dim
742 \dim_new:N \l__draw_path_xmax_dim
743 \dim_new:N \l__draw_path_xmin_dim
744 \dim_new:N \l__draw_path_ymax_dim
745 \dim_new:N \l__draw_path_ymin_dim
746 \dim_new:N \l__draw_softpath_lastx_dim
747 \dim_new:N \l__draw_softpath_lasty_dim
748 \bool_new:N \l__draw_softpath_corners_bool

```

(End definition for `\l__draw_path_lastx_dim` and others.)

`\draw_path_scope_begin:` Scoping a path is a bit more involved, largely as there are a number of variables to keep hold of.

```

749 \cs_new_protected:Npn \draw_path_scope_begin:
750 {
751   \group_begin:
752     \dim_set_eq:NN \l__draw_path_lastx_dim \g__draw_path_lastx_dim
753     \dim_set_eq:NN \l__draw_path_lasty_dim \g__draw_path_lasty_dim
754     \dim_set_eq:NN \l__draw_path_xmax_dim \g__draw_path_xmax_dim
755     \dim_set_eq:NN \l__draw_path_xmin_dim \g__draw_path_xmin_dim
756     \dim_set_eq:NN \l__draw_path_ymax_dim \g__draw_path_ymax_dim
757     \dim_set_eq:NN \l__draw_path_ymin_dim \g__draw_path_ymin_dim
758     \dim_set_eq:NN \l__draw_softpath_lastx_dim \g__draw_softpath_lastx_dim
759     \dim_set_eq:NN \l__draw_softpath_lasty_dim \g__draw_softpath_lasty_dim
760     \__draw_path_reset_limits:
761     \tl_build_get:NN \g__draw_softpath_main_tl \l__draw_softpath_main_tl
762     \bool_set_eq:NN
763       \l__draw_softpath_corners_bool
764       \g__draw_softpath_corners_bool
765     \__draw_softpath_clear:
766   }
767 \cs_new_protected:Npn \draw_path_scope_end:
768 {
769   \__draw_softpath_clear:
770   \bool_gset_eq:NN
771     \g__draw_softpath_corners_bool
772     \l__draw_softpath_corners_bool
773   \__draw_softpath_add:o \l__draw_softpath_main_tl
774   \dim_gset_eq:NN \g__draw_softpath_lastx_dim \l__draw_softpath_lastx_dim
775   \dim_gset_eq:NN \g__draw_softpath_lasty_dim \l__draw_softpath_lasty_dim
776   \dim_gset_eq:NN \g__draw_path_xmax_dim \l__draw_path_xmax_dim
777   \dim_gset_eq:NN \g__draw_path_xmin_dim \l__draw_path_xmin_dim
778   \dim_gset_eq:NN \g__draw_path_ymax_dim \l__draw_path_ymax_dim
779   \dim_gset_eq:NN \g__draw_path_ymin_dim \l__draw_path_ymin_dim
780   \dim_gset_eq:NN \g__draw_path_lastx_dim \l__draw_path_lastx_dim
781   \dim_gset_eq:NN \g__draw_path_lasty_dim \l__draw_path_lasty_dim
782   \group_end:
783 }

```

(End definition for `\draw_path_scope_begin:` and `\draw_path_scope_end:`. These functions are documented on page ??.)

```

784 \msg_new:nnnn { draw } { invalid-path-action }
785 { Invalid-action-~'#1'~for-path. }
786 { Paths-can-be-used-with-actions-~'draw',~'clip',~'fill'~or~'stroke'. }
787 % \end{macrocode}
788 %
789 % \begin{macrocode}
790 </initex | package>

```

5 l3draw-points implementation

```

791 <*initex | package>
792 <@@=draw>

```

This sub-module covers more-or-less the same ideas as `pgfcorepoints.code.tex`, though the approach taken to returning values is different: point expressions here are

processed by expansion and return a co-ordinate pair in the form $\{\langle x \rangle\}\{\langle y \rangle\}$. Equivalents of following pgf functions are deliberately omitted:

- `\pgfpointorigin`: Can be given explicitly as `0pt,0pt`.
- `\pgfpointadd`, `\pgfpointdiff`, `\pgfpointscale`: Can be given explicitly.
- `\pgfextractx`, `\pgfextracty`: Available by applying `\use_i:nn/\use_ii:nn` or similar to the `x`-type expansion of a point expression.
- `\pgfgetlastxy`: Unused in the entire pgf core, may be emulated by `x`-type expansion of a point expression, then using the result.

In addition, equivalents of the following *may* be added in future but are currently absent:

- `\pgfpointcylindrical`, `\pgfpointspherical`: The usefulness of these commands is not currently clear.
- `\pgfpointborderrectangle`, `\pgfpointborderellipse`: To be revisited once the semantics and use cases are clear.
- `\pgfqpoint`, `\pgfqpointscale`, `\pgfqpointpolar`, `\pgfqpointxy`, `\pgfqpointxyz`: The expandable approach taken in the code here, along with the absolute requirement for ε -TeX, means it is likely many use cases for these commands may be covered in other ways. This may be revisited as higher-level structures are constructed.

5.1 Support functions

Execute whatever code is passed to extract the x and y co-ordinates. The first argument here should itself absorb two arguments. There is also a version to deal with two co-ordinates: common enough to justify a separate function.

```

\__draw_point_process:nn
  \__draw_point_process_auxi:nn
  \__draw_point_process_auxii:nw
\__draw_point_process:nnn
  \__draw_point_process_auxiii:nnn
  \__draw_point_process_auxiv:nw
\__draw_point_process:nnnn
  \__draw_point_process_auxv:nnnn
  \__draw_point_process_auxvi:nw
\__draw_point_process:nnnnn
  \__draw_point_process_auxvii:nnnnn
  \__draw_point_process_auxviii:nw
793 \cs_new:Npn \__draw_point_process:nn #1#2
794 {
795   \exp_args:Nf \__draw_point_process_auxi:nn
796   { \__draw_point_to_dim:n {#2} }
797   {#1}
798 }
799 \cs_new:Npn \__draw_point_process_auxi:nn #1#2
800 { \__draw_point_process_auxii:nw {#2} #1 \s__draw_stop }
801 \cs_new:Npn \__draw_point_process_auxii:nw #1 #2 , #3 \s__draw_stop
802 { #1 {#2} {#3} }
803 \cs_new:Npn \__draw_point_process:nnn #1#2#3
804 {
805   \exp_args:Nff \__draw_point_process_auxiii:nnn
806   { \__draw_point_to_dim:n {#2} }
807   { \__draw_point_to_dim:n {#3} }
808   {#1}
809 }
810 \cs_new:Npn \__draw_point_process_auxiii:nnn #1#2#3
811 { \__draw_point_process_auxiv:nw {#3} #1 \s__draw_mark #2 \s__draw_stop }
812 \cs_new:Npn \__draw_point_process_auxiv:nw #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_stop
813 { #1 {#2} {#3} {#4} {#5} }
814 \cs_new:Npn \__draw_point_process:nnnn #1#2#3#4
815 {

```

```

816 \exp_args:Nfff \__draw_point_process_auxv:nnnn
817 { \__draw_point_to_dim:n {#2} }
818 { \__draw_point_to_dim:n {#3} }
819 { \__draw_point_to_dim:n {#4} }
820 {#1}
821 }
822 \cs_new:Npn \__draw_point_process_auxv:nnnn #1#2#3#4
823 { \__draw_point_process_auxvi:nw {#4} #1 \s__draw_mark #2 \s__draw_mark #3 \s__draw_stop }
824 \cs_new:Npn \__draw_point_process_auxvi:nw
825 #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_mark #6 , #7 \s__draw_stop
826 { #1 {#2} {#3} {#4} {#5} {#6} {#7} }
827 \cs_new:Npn \__draw_point_process:nnnnn #1#2#3#4#5
828 {
829 \exp_args:Nffff \__draw_point_process_auxvii:nnnnn
830 { \__draw_point_to_dim:n {#2} }
831 { \__draw_point_to_dim:n {#3} }
832 { \__draw_point_to_dim:n {#4} }
833 { \__draw_point_to_dim:n {#5} }
834 {#1}
835 }
836 \cs_new:Npn \__draw_point_process_auxvii:nnnnn #1#2#3#4#5
837 {
838 \__draw_point_process_auxviii:nw
839 {#5} #1 \s__draw_mark #2 \s__draw_mark #3 \s__draw_mark #4 \s__draw_stop
840 }
841 \cs_new:Npn \__draw_point_process_auxviii:nw
842 #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_mark #6 , #7 \s__draw_mark #8 , #9 \s__draw_stop
843 { #1 {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9} }

```

(End definition for __draw_point_process:nn and others.)

```

\__draw_point_to_dim:n Co-ordinates are always returned as two dimensions.
\__draw_point_to_dim_aux:n
\__draw_point_to_dim_aux:f
\__draw_point_to_dim_aux:w
844 \cs_new:Npn \__draw_point_to_dim:n #1
845 { \__draw_point_to_dim_aux:f { \fp_eval:n {#1} } }
846 \cs_new:Npn \__draw_point_to_dim_aux:n #1
847 { \__draw_point_to_dim_aux:w #1 }
848 \cs_generate_variant:Nn \__draw_point_to_dim_aux:n { f }
849 \cs_new:Npn \__draw_point_to_dim_aux:w ( #1 , ~ #2 ) { #1pt , #2pt }

```

5.2 Polar co-ordinates

Polar co-ordinates may have either one or two lengths, so there is a need to do a simple split before the calculation. As the angle gets used twice, save on any expression evaluation there and force expansion.

```

\draw_point_polar:nn
\draw_point_polar:nnn
\__draw_draw_polar:nnn
\__draw_draw_polar:fnn
850 \cs_new:Npn \draw_point_polar:nn #1#2
851 { \draw_point_polar:nnn {#1} {#1} {#2} }
852 \cs_new:Npn \draw_point_polar:nnn #1#2#3
853 { \__draw_draw_polar:fnn { \fp_eval:n {#3} } {#1} {#2} }
854 \cs_new:Npn \__draw_draw_polar:nnn #1#2#3
855 { \__draw_point_to_dim:n { cosd(#1) * (#2) , sind(#1) * (#3) } }
856 \cs_generate_variant:Nn \__draw_draw_polar:nnn { f }

```

5.3 Point expression arithmetic

These functions all take point expressions as arguments.

The outcome is the normalised vector from (0,0) in the direction of the point, *i.e.*

```
\draw_point_unit_vector:n
\_draw_point_unit_vector:nn
  \_draw_point_unit_vector:nnn
```

$$P_x = \frac{x}{\sqrt{x^2 + y^2}} \quad P_y = \frac{y}{\sqrt{x^2 + y^2}}$$

except where the length is zero, in which case a vertical vector is returned.

```
857 \cs_new:Npn \draw_point_unit_vector:n #1
858 { \_draw_point_process:nn { \_draw_point_unit_vector:nn } {#1} }
859 \cs_new:Npn \_draw_point_unit_vector:nn #1#2
860 {
861   \exp_args:Nf \_draw_point_unit_vector:nnn
862     { \fp_eval:n { (sqrt(#1 * #1 + #2 * #2)) } }
863     {#1} {#2}
864 }
865 \cs_new:Npn \_draw_point_unit_vector:nnn #1#2#3
866 {
867   \fp_compare:nNnTF {#1} = \c_zero_fp
868     { Opt, 1pt }
869     {
870       \_draw_point_to_dim:n
871         { ( #2 , #3 ) / #1 }
872     }
873 }
```

5.4 Intersection calculations

The intersection point P between a line joining points (x_1, y_1) and (x_2, y_2) with a second line joining points (x_3, y_3) and (x_4, y_4) can be calculated using the formulae

```
\draw_point_intersect_lines:nnnn
\_draw_point_intersect_lines:nnnnnn
\_draw_point_intersect_lines:nnnnnnnn
\_draw_point_intersect_lines_aux:nnnnnn
\_draw_point_intersect_lines_aux:fffff
```

$$P_x = \frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_3y_4 - y_3x_4)(x_1 - x_2)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}$$

and

$$P_y = \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (x_3y_4 - y_3x_4)(y_1 - y_2)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}$$

The work therefore comes down to expanding the incoming data, then pre-calculating as many parts as possible before the final work to find the intersection. (Expansion and argument re-ordering is much less work than additional floating point calculations.)

```
874 \cs_new:Npn \draw_point_intersect_lines:nnnn #1#2#3#4
875 {
876   \_draw_point_process:nnnnn
877     { \_draw_point_intersect_lines:nnnnnnnn }
878     {#1} {#2} {#3} {#4}
879 }
```

At this stage we have all of the information we need, fully expanded:

```
#1 x_1
#2 y_1
```

#3 x_2
#4 y_2
#5 x_3
#6 y_3
#7 x_4
#8 y_4

so now just have to do all of the calculation.

```

880 \cs_new:Npn \__draw_point_intersect_lines:nnnnnnnn #1#2#3#4#5#6#7#8
881 {
882   \__draw_point_intersect_lines_aux:ffffff
883   { \fp_eval:n { #1 * #4 - #2 * #3 } }
884   { \fp_eval:n { #5 * #8 - #6 * #7 } }
885   { \fp_eval:n { #1 - #3 } }
886   { \fp_eval:n { #5 - #7 } }
887   { \fp_eval:n { #2 - #4 } }
888   { \fp_eval:n { #6 - #8 } }
889 }
890 \cs_new:Npn \__draw_point_intersect_lines_aux:nnnnnn #1#2#3#4#5#6
891 {
892   \__draw_point_to_dim:n
893   {
894     ( #2 * #3 - #1 * #4 , #2 * #5 - #1 * #6 )
895     / ( #4 * #5 - #6 * #3 )
896   }
897 }
898 \cs_generate_variant:Nn \__draw_point_intersect_lines_aux:nnnnnn { fffffff }

```

Another long expansion chain to get the values in the right places. We have two circles, the first with center (a, b) and radius r , the second with center (c, d) and radius s . We use the intermediate values

$$\begin{aligned}
e &= c - a \\
f &= d - b \\
p &= \sqrt{e^2 + f^2} \\
k &= \frac{p^2 + r^2 - s^2}{2p}
\end{aligned}$$

in either

$$\begin{aligned}
P_x &= a + \frac{ek}{p} + \frac{f}{p}\sqrt{r^2 - k^2} \\
P_y &= b + \frac{fk}{p} - \frac{e}{p}\sqrt{r^2 - k^2}
\end{aligned}$$

or

$$\begin{aligned}
P_x &= a + \frac{ek}{p} - \frac{f}{p}\sqrt{r^2 - k^2} \\
P_y &= b + \frac{fk}{p} + \frac{e}{p}\sqrt{r^2 - k^2}
\end{aligned}$$

depending on which solution is required. The rest of the work is simply forcing the appropriate expansion and shuffling arguments.

```

899 \cs_new:Npn \draw_point_intersect_circles:nnnnn #1#2#3#4#5
900 {
901   \__draw_point_process:nnn
902   { \__draw_point_intersect_circles_auxi:nnnnnnn {#2} {#4} {#5} }
903   {#1} {#3}
904 }
905 \cs_new:Npn \__draw_point_intersect_circles_auxi:nnnnnnn #1#2#3#4#5#6#7
906 {
907   \__draw_point_intersect_circles_auxii:ffnnnnnn
908   { \fp_eval:n {#1} } { \fp_eval:n {#2} } {#4} {#5} {#6} {#7} {#3}
909 }

```

At this stage we have all of the information we need, fully expanded:

```

#1 r
#2 s
#3 a
#4 b
#5 c
#6 d
#7 n

```

Once we evaluate e and f , the co-ordinate (c, d) is no longer required: handy as we will need various intermediate values in the following.

```

910 \cs_new:Npn \__draw_point_intersect_circles_auxii:nnnnnnn #1#2#3#4#5#6#7
911 {
912   \__draw_point_intersect_circles_auxiii:ffnnnnnn
913   { \fp_eval:n { #5 - #3 } }
914   { \fp_eval:n { #6 - #4 } }
915   {#1} {#2} {#3} {#4} {#7}
916 }
917 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxii:nnnnnnn { ff }
918 \cs_new:Npn \__draw_point_intersect_circles_auxiii:nnnnnnn #1#2#3#4#5#6#7
919 {
920   \__draw_point_intersect_circles_auxiv:fnnnnnnn
921   { \fp_eval:n { sqrt( #1 * #1 + #2 * #2 ) } }
922   {#1} {#2} {#3} {#4} {#5} {#6} {#7}
923 }
924 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxiii:nnnnnnn { ff }

```

We now have p : we pre-calculate $1/p$ as it is needed a few times and is relatively expensive. We also need r^2 twice so deal with that here too.

```

925 \cs_new:Npn \__draw_point_intersect_circles_auxiv:fnnnnnnn #1#2#3#4#5#6#7#8
926 {
927   \__draw_point_intersect_circles_auxv:ffnnnnnnnn
928   { \fp_eval:n { 1 / #1 } }
929   { \fp_eval:n { #4 * #4 } }
930   {#1} {#2} {#3} {#5} {#6} {#7} {#8}

```

```

931 }
932 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxiv:nnnnnnnn { f }
933 \cs_new:Npn \__draw_point_intersect_circles_auxv:nnnnnnnn #1#2#3#4#5#6#7#8#9
934 {
935   \__draw_point_intersect_circles_auxvi:fnnnnnnn
936   { \fp_eval:n { 0.5 * #1 * ( #2 + #3 * #3 - #6 * #6 ) } }
937   {#1} {#2} {#4} {#5} {#7} {#8} {#9}
938 }
939 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxv:nnnnnnnn { ff }

```

We now have all of the intermediate values we require, with one division carried out up-front to avoid doing this expensive step twice:

```

#1  $k$ 
#2  $1/p$ 
#3  $r^2$ 
#4  $e$ 
#5  $f$ 
#6  $a$ 
#7  $b$ 
#8  $n$ 

```

There are some final pre-calculations, k/p , $\frac{\sqrt{r^2-k^2}}{p}$ and the usage of n , then we can yield a result.

```

940 \cs_new:Npn \__draw_point_intersect_circles_auxvi:nnnnnnnn #1#2#3#4#5#6#7#8
941 {
942   \__draw_point_intersect_circles_auxvii:ffnnnnn
943   { \fp_eval:n { #1 * #2 } }
944   { \int_if_odd:nTF {#8} { 1 } { -1 } }
945   { \fp_eval:n { sqrt ( #3 - #1 * #1 ) * #2 } }
946   {#4} {#5} {#6} {#7}
947 }
948 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxvi:nnnnnnnn { f }
949 \cs_new:Npn \__draw_point_intersect_circles_auxvii:nnnnnnnn #1#2#3#4#5#6#7
950 {
951   \__draw_point_to_dim:n
952   { #6 + #4 * #1 + #2 * #3 * #5 , #7 + #5 * #1 + -1 * #2 * #3 * #4 }
953 }
954 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxvii:nnnnnnnn { fff }

```

5.5 Interpolation on a line (vector) or arc

Simple maths after expansion.

```

\draw_point_interpolate_line:nnn
\__draw_point_interpolate_line_aux:nnnnn
\__draw_point_interpolate_line_aux:fnnnn
\__draw_point_interpolate_line_aux:nnnnnn
\__draw_point_interpolate_line_aux:fnnnnn
\__draw_point_interpolate_line_aux:fnnnnn
955 \cs_new:Npn \draw_point_interpolate_line:nnn #1#2#3
956 {
957   \__draw_point_process:nnn
958   { \__draw_point_interpolate_line_aux:fnnnn { \fp_eval:n {#1} } }
959   {#2} {#3}
960 }

```

```

961 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnn #1#2#3#4#5
962 {
963   \__draw_point_interpolate_line_aux:fnnnnn { \fp_eval:n { 1 - #1 } }
964   {#1} {#2} {#3} {#4} {#5}
965 }
966 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnn { f }
967 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnnn #1#2#3#4#5#6
968 { \__draw_point_to_dim:n { #2 * #3 + #1 * #5 , #2 * #4 + #1 * #6 } }
969 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnnn { f }

```

Same idea but using the normalised length to obtain the scale factor. The start point is needed twice, so we force evaluation, but the end point is needed only the once.

```

\draw_point_interpolate_distance:nnm
\__draw_point_interpolate_distance:nnnnn
\__draw_point_interpolate_distance:nnnnnn
\__draw_point_interpolate_distance:fnnnnn
970 \cs_new:Npn \draw_point_interpolate_distance:nnn #1#2#3
971 {
972   \__draw_point_process:nn
973   { \__draw_point_interpolate_distance:nnnn {#1} {#3} }
974   {#2}
975 }
976 \cs_new:Npn \__draw_point_interpolate_distance:nnnn #1#2#3#4
977 {
978   \__draw_point_process:nn
979   {
980     \__draw_point_interpolate_distance:fnnnn
981     { \fp_eval:n {#1} } {#3} {#4}
982   }
983   { \draw_point_unit_vector:n { ( #2 ) - ( #3 , #4 ) } }
984 }
985 \cs_new:Npn \__draw_point_interpolate_distance:nnnnn #1#2#3#4#5
986 { \__draw_point_to_dim:n { #2 + #1 * #4 , #3 + #1 * #5 } }
987 \cs_generate_variant:Nn \__draw_point_interpolate_distance:nnnnn { f }

```

(End definition for __draw_point_to_dim:n and others. These functions are documented on page ??.)

```

\draw_point_interpolate_arcaxes:nnnnnn
\draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn
\draw_point_interpolate_arcaxes_auxii:nnnnnnnnnn
\draw_point_interpolate_arcaxes_auxii:fnnnnnnnnnn
\draw_point_interpolate_arcaxes_auxiii:nnnnnnnnnn
\draw_point_interpolate_arcaxes_auxiii:fnnnnnnnnnn
\draw_point_interpolate_arcaxes_auxiv:nnnnnnnnnn
\draw_point_interpolate_arcaxes_auxiv:ffnnnnnnnn

```

Finding a point on an ellipse arc is relatively easy: find the correct angle between the two given, use the sine and cosine of that angle, apply to the axes. We just have to work a bit with the co-ordinate expansion.

```

988 \cs_new:Npn \draw_point_interpolate_arcaxes:nnnnnn #1#2#3#4#5#6
989 {
990   \__draw_point_process:nnnn
991   { \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn {#1} {#5} {#6} }
992   {#2} {#3} {#4}
993 }
994 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
995 {
996   \__draw_point_interpolate_arcaxes_auxii:fnnnnnnnnnn
997   { \fp_eval:n {#1} } {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
998 }

```

At this stage, the three co-ordinate pairs are fully expanded but somewhat re-ordered:

```

#1 p
#2  $\theta_1$ 
#3  $\theta_2$ 

```

#4 x_c

#5 y_c

#6 x_{a1}

#7 y_{a1}

#8 x_{a2}

#9 y_{a2}

We are now in a position to find the target angle, and from that the sine and cosine required.

```

999 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnn #1#2#3#4#5#6#7#8#9
1000 {
1001   \__draw_point_interpolate_arcaxes_auxiii:fnnnnnnn
1002   { \fp_eval:n { #1 * (#3) + ( 1 - #1 ) * (#2) } }
1003   {#4} {#5} {#6} {#7} {#8} {#9}
1004 }
1005 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnn { f }
1006 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnnn #1#2#3#4#5#6#7
1007 {
1008   \__draw_point_interpolate_arcaxes_auxiv:ffnnnnnnn
1009   { \fp_eval:n { cosd (#1) } }
1010   { \fp_eval:n { sind (#1) } }
1011   {#2} {#3} {#4} {#5} {#6} {#7}
1012 }
1013 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnnnn { f }
1014 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnnn #1#2#3#4#5#6#7#8
1015 {
1016   \__draw_point_to_dim:n
1017   { #3 + #1 * #5 + #2 * #7 , #4 + #1 * #6 + #2 * #8 }
1018 }
1019 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnnn { ff }

```

(End definition for `\draw_point_interpolate_arcaxes:nnnnnn` and others. This function is documented on page ??.)

\draw_point_interpolate_curve:nnnnn
\draw_point_interpolate_curve_auxi:nnnnnnnnn
\draw_point_interpolate_curve_auxii:nnnnnnnnn
\draw_point_interpolate_curve_auxiii:fnnnnnnnnn
\draw_point_interpolate_curve_auxiiii:nnnnnnn
\draw_point_interpolate_curve_auxiiii:fnnnnn
\draw_point_interpolate_curve_auxiv:nnnnnnn
\draw_point_interpolate_curve_auxv:nnw
\draw_point_interpolate_curve_auxv:ffw
\draw_point_interpolate_curve_auxvi:n
\draw_point_interpolate_curve_auxvii:nnnnnnnnn
\draw_point_interpolate_curve_auxviii:nnnnnnn
\draw_point_interpolate_curve_auxviii:ffnnnnn

Here we start with a proportion of the curve (p) and four points

1. The initial point (x_1, y_1)
2. The first control point (x_2, y_2)
3. The second control point (x_3, y_3)
4. The final point (x_4, y_4)

The first phase is to expand out all of these values.

```

1020 \cs_new:Npn \draw_point_interpolate_curve:nnnnnn #1#2#3#4#5
1021 {
1022   \__draw_point_process:nnnnn
1023   { \__draw_point_interpolate_curve_auxi:nnnnnnnnn {#1} }
1024   {#2} {#3} {#4} {#5}
1025 }

```

```

1026 \cs_new:Npn \__draw_point_interpolate_curve_auxi:nnnnnnnnn #1#2#3#4#5#6#7#8#9
1027 {
1028   \__draw_point_interpolate_curve_auxii:fnnnnnnnn
1029   { \fp_eval:n {#1} }
1030   {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1031 }

```

At this stage, everything is fully expanded and back in the input order. The approach to finding the required point is iterative. We carry out three phases. In phase one, we need all of the input co-ordinates

$$\begin{aligned}
x'_1 &= (1-p)x_1 + px_2 \\
y'_1 &= (1-p)y_1 + py_2 \\
x'_2 &= (1-p)x_2 + px_3 \\
y'_2 &= (1-p)y_2 + py_3 \\
x'_3 &= (1-p)x_3 + px_4 \\
y'_3 &= (1-p)y_3 + py_4
\end{aligned}$$

In the second stage, we can drop the final point

$$\begin{aligned}
x''_1 &= (1-p)x'_1 + px'_2 \\
y''_1 &= (1-p)y'_1 + py'_2 \\
x''_2 &= (1-p)x'_2 + px'_3 \\
y''_2 &= (1-p)y'_2 + py'_3
\end{aligned}$$

and for the final stage only need one set of calculations

$$\begin{aligned}
P_x &= (1-p)x''_1 + px''_2 \\
P_y &= (1-p)y''_1 + py''_2
\end{aligned}$$

Of course, this does mean a lot of calculations and expansion!

```

1032 \cs_new:Npn \__draw_point_interpolate_curve_auxii:nnnnnnnnn
1033   #1#2#3#4#5#6#7#8#9
1034 {
1035   \__draw_point_interpolate_curve_auxiii:fnnnnnn
1036   { \fp_eval:n { 1 - #1 } }
1037   {#1}
1038   { {#2} {#3} } { {#4} {#5} } { {#6} {#7} } { {#8} {#9} }
1039 }
1040 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxii:nnnnnnnnn { f }
1041 % \begin{macrocode}
1042 % We need to do the first cycle, but haven't got enough arguments to keep
1043 % everything in play at once. So here we use a but of argument re-ordering
1044 % and a single auxiliary to get the job done.
1045 % \begin{macrocode}
1046 \cs_new:Npn \__draw_point_interpolate_curve_auxiii:nnnnnn #1#2#3#4#5#6
1047 {
1048   \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #3 #4
1049   \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #4 #5
1050   \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #5 #6
1051   \prg_do_nothing:
1052   \__draw_point_interpolate_curve_auxvi:n { {#1} {#2} }

```

```

1053 }
1054 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxiii:nnnnnn { f }
1055 \cs_new:Npn \__draw_point_interpolate_curve_auxiv:nnnnnn #1#2#3#4#5#6
1056 {
1057   \__draw_point_interpolate_curve_auxv:ffw
1058   { \fp_eval:n { #1 * #3 + #2 * #5 } }
1059   { \fp_eval:n { #1 * #4 + #2 * #6 } }
1060 }
1061 \cs_new:Npn \__draw_point_interpolate_curve_auxv:nnw
1062 #1#2#3 \prg_do_nothing: #4#5
1063 {
1064   #3
1065   \prg_do_nothing:
1066   #4 { #5 {#1} {#2} }
1067 }
1068 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxv:nnw { ff }
1069 % \begin{macrocode}
1070 % Get the arguments back into the right places and to the second and
1071 % third cycles directly.
1072 % \begin{macrocode}
1073 \cs_new:Npn \__draw_point_interpolate_curve_auxvi:n #1
1074 { \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1 }
1075 \cs_new:Npn \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1#2#3#4#5#6#7#8
1076 {
1077   \__draw_point_interpolate_curve_auxviii:ffffnn
1078   { \fp_eval:n { #1 * #5 + #2 * #3 } }
1079   { \fp_eval:n { #1 * #6 + #2 * #4 } }
1080   { \fp_eval:n { #1 * #7 + #2 * #5 } }
1081   { \fp_eval:n { #1 * #8 + #2 * #6 } }
1082   {#1} {#2}
1083 }
1084 \cs_new:Npn \__draw_point_interpolate_curve_auxviii:nnnnnn #1#2#3#4#5#6
1085 {
1086   \__draw_point_to_dim:n
1087   { #5 * #3 + #6 * #1 , #5 * #4 + #6 * #2 }
1088 }
1089 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxviii:nnnnnn { ffff }

```

(End definition for \draw_point_interpolate_curve:nnnnn and others. These functions are documented on page ??.)

5.6 Vector support

As well as co-ordinates relative to the drawing

```

\l__draw_xvec_x_dim Base vectors to map to the underlying two-dimensional drawing space.
\l__draw_xvec_y_dim
\l__draw_yvec_x_dim 1090 \dim_new:N \l__draw_xvec_x_dim
\l__draw_yvec_y_dim 1091 \dim_new:N \l__draw_xvec_y_dim
\l__draw_zvec_x_dim 1092 \dim_new:N \l__draw_yvec_x_dim
\l__draw_zvec_y_dim 1093 \dim_new:N \l__draw_yvec_y_dim
1094 \dim_new:N \l__draw_zvec_x_dim
1095 \dim_new:N \l__draw_zvec_y_dim

```

(End definition for \l__draw_xvec_x_dim and others.)

```

\draw_xvec:n Calculate the underlying position and store it.
\draw_yvec:n 1096 \cs_new_protected:Npn \draw_xvec:n #1
\draw_zvec:n 1097 { \__draw_vec:nn { x } {#1} }
\__draw_vec:nn 1098 \cs_new_protected:Npn \draw_yvec:n #1
\__draw_vec:nnn 1099 { \__draw_vec:nn { y } {#1} }
1100 \cs_new_protected:Npn \draw_zvec:n #1
1101 { \__draw_vec:nn { z } {#1} }
1102 \cs_new_protected:Npn \__draw_vec:nn #1#2
1103 {
1104   \__draw_point_process:nn { \__draw_vec:nnn {#1} } {#2}
1105 }
1106 \cs_new_protected:Npn \__draw_vec:nnn #1#2#3
1107 {
1108   \dim_set:cn { l__draw_ #1 vec_x_dim } {#2}
1109   \dim_set:cn { l__draw_ #1 vec_y_dim } {#3}
1110 }

```

(End definition for \draw_xvec:n and others. These functions are documented on page ??.)

Initialise the vectors.

```

1111 \draw_xvec:n { 1cm , 0cm }
1112 \draw_yvec:n { 0cm , 1cm }
1113 \draw_zvec:n { -0.385cm , -0.385cm }

```

```

\draw_point_vec:nn Force a single evaluation of each factor, then use these to work out the underlying point.
\__draw_point_vec:nn 1114 \cs_new:Npn \draw_point_vec:nn #1#2
\__draw_point_vec:ff 1115 { \__draw_point_vec:ff { \fp_eval:n {#1} } { \fp_eval:n {#2} } }
\draw_point_vec:nnn 1116 \cs_new:Npn \__draw_point_vec:nn #1#2
\__draw_point_vec:nnn 1117 {
\__draw_point_vec:fff 1118   \__draw_point_to_dim:n
1119   {
1120     #1 * \l__draw_xvec_x_dim + #2 * \l__draw_yvec_x_dim ,
1121     #1 * \l__draw_xvec_y_dim + #2 * \l__draw_yvec_y_dim
1122   }
1123 }
1124 \cs_generate_variant:Nn \__draw_point_vec:nn { ff }
1125 \cs_new:Npn \draw_point_vec:nnn #1#2#3
1126 {
1127   \__draw_point_vec:fff
1128   { \fp_eval:n {#1} } { \fp_eval:n {#2} } { \fp_eval:n {#3} }
1129 }
1130 \cs_new:Npn \__draw_point_vec:nnn #1#2#3
1131 {
1132   \__draw_point_to_dim:n
1133   {
1134     #1 * \l__draw_xvec_x_dim
1135     + #2 * \l__draw_yvec_x_dim
1136     + #3 * \l__draw_zvec_x_dim
1137     ,
1138     #1 * \l__draw_xvec_y_dim
1139     + #2 * \l__draw_yvec_y_dim
1140     + #3 * \l__draw_zvec_y_dim
1141   }
1142 }
1143 \cs_generate_variant:Nn \__draw_point_vec:nnn { fff }

```

(End definition for `\draw_point_vec:nn` and others. These functions are documented on page ??.)

```

\draw_point_vec_polar:nn Much the same as the core polar approach.
\draw_point_vec_polar:nnn
\__draw_point_vec_polar:nnn
\__draw_point_vec_polar:fnn
1144 \cs_new:Npn \draw_point_vec_polar:nn #1#2
1145 { \draw_point_vec_polar:nnn {#1} {#1} {#2} }
1146 \cs_new:Npn \draw_point_vec_polar:nnn #1#2#3
1147 { \__draw_draw_vec_polar:fnn { \fp_eval:n {#3} } {#1} {#2} }
1148 \cs_new:Npn \__draw_draw_vec_polar:nnn #1#2#3
1149 {
1150   \__draw_point_to_dim:n
1151   {
1152     cosd(#1) * (#2) * \l__draw_xvec_x_dim ,
1153     sind(#1) * (#3) * \l__draw_yvec_y_dim
1154   }
1155 }
1156 \cs_generate_variant:Nn \__draw_draw_vec_polar:nnn { f }

```

(End definition for `\draw_point_vec_polar:nn`, `\draw_point_vec_polar:nnn`, and `__draw_point_vec_polar:nnn`. These functions are documented on page ??.)

5.7 Transformations

`\draw_point_transform:n` Applies a transformation matrix to a point: see `l3draw-transforms` for the business end. Where possible, we avoid the relatively expensive multiplication step.

```

1157 \cs_new:Npn \draw_point_transform:n #1
1158 {
1159   \__draw_point_process:nn
1160   { \__draw_point_transform:nn } {#1}
1161 }
1162 \cs_new:Npn \__draw_point_transform:nn #1#2
1163 {
1164   \bool_if:NTF \l__draw_matrix_active_bool
1165   {
1166     \__draw_point_to_dim:n
1167     {
1168       (
1169         \l__draw_matrix_a_fp * #1
1170         + \l__draw_matrix_c_fp * #2
1171         + \l__draw_xshift_dim
1172       )
1173       ,
1174       (
1175         \l__draw_matrix_b_fp * #1
1176         + \l__draw_matrix_d_fp * #2
1177         + \l__draw_yshift_dim
1178       )
1179     }
1180   }
1181   {
1182     \__draw_point_to_dim:n
1183     {
1184       (#1, #2)
1185       + ( \l__draw_xshift_dim , \l__draw_yshift_dim )
1186     }

```



```

1187     }
1188 }

```

(End definition for `\draw_point_transform:n` and `__draw_point_transform:nn`. This function is documented on page ??.)

```

\__draw_point_transform_noshift:n
\__draw_point_transform_noshift:nn

```

A version with no shift: used for internal purposes.

```

1189 \cs_new:Npn \__draw_point_transform_noshift:n #1
1190 {
1191   \__draw_point_process:nn
1192   { \__draw_point_transform_noshift:nn } {#1}
1193 }
1194 \cs_new:Npn \__draw_point_transform_noshift:nn #1#2
1195 {
1196   \bool_if:NTF \l__draw_matrix_active_bool
1197   {
1198     \__draw_point_to_dim:n
1199     {
1200       (
1201         \l__draw_matrix_a_fp * #1
1202         + \l__draw_matrix_c_fp * #2
1203       )
1204       ,
1205       (
1206         \l__draw_matrix_b_fp * #1
1207         + \l__draw_matrix_d_fp * #2
1208       )
1209     }
1210   }
1211   { \__draw_point_to_dim:n { (#1, #2) } }
1212 }

```

(End definition for `__draw_point_transform_noshift:n` and `__draw_point_transform_noshift:nn`.)

```

1213 </initex | package>

```

6 l3draw-scopes implementation

```

1214 <*initex | package>

```

```

1215 <@@=draw>

```

6.1 Drawing environment

```

\g__draw_xmax_dim
\g__draw_xmin_dim
\g__draw_ymax_dim
\g__draw_ymin_dim

```

Used to track the overall (official) size of the image created: may not actually be the natural size of the content.

```

1216 \dim_new:N \g__draw_xmax_dim
1217 \dim_new:N \g__draw_xmin_dim
1218 \dim_new:N \g__draw_ymax_dim
1219 \dim_new:N \g__draw_ymin_dim

```

(End definition for `\g__draw_xmax_dim` and others.)

```

\l_draw_bb_update_bool

```

Flag to indicate that a path (or similar) should update the bounding box of the drawing.

```

1220 \bool_new:N \l_draw_bb_update_bool

```

(End definition for `\l_draw_bb_update_bool`. This variable is documented on page ??.)

`\l__draw_layer_main_box` Box for setting the drawing itself and the top-level layer.

```
1221 \box_new:N \l__draw_main_box
1222 \box_new:N \l__draw_layer_main_box
```

(End definition for `\l__draw_layer_main_box`.)

`\g__draw_id_int` The drawing number.

```
1223 \int_new:N \g__draw_id_int
```

(End definition for `\g__draw_id_int`.)

`__draw_reset_bb:` A simple auxiliary.

```
1224 \cs_new_protected:Npn \__draw_reset_bb:
1225 {
1226   \dim_gset:Nn \g__draw_xmax_dim { -\c_max_dim }
1227   \dim_gset:Nn \g__draw_xmin_dim { \c_max_dim }
1228   \dim_gset:Nn \g__draw_ymax_dim { -\c_max_dim }
1229   \dim_gset:Nn \g__draw_ymin_dim { \c_max_dim }
1230 }
```

(End definition for `__draw_reset_bb:`.)

`\draw_begin:` Drawings are created by setting them into a box, then adjusting the box before inserting
`\draw_end:` into the surroundings. Color is set here using the drawing mechanism largely as it then sets up the internal data structures. It may be that a coffin construct is better here in the longer term: that may become clearer as the code is completed. As we need to avoid any insertion of baseline skips, the outer box here has to be an `hbox`. To allow for layers, there is some box nesting: notice that we

```
1231 \cs_new_protected:Npn \draw_begin:
1232 {
1233   \group_begin:
1234   \int_gincr:N \g__draw_id_int
1235   \hbox_set:Nw \l__draw_main_box
1236   \__draw_backend_begin:
1237   \__draw_reset_bb:
1238   \__draw_path_reset_limits:
1239   \bool_set_true:N \l_draw_bb_update_bool
1240   \draw_transform_matrix_reset:
1241   \draw_transform_shift_reset:
1242   \__draw_softpath_clear:
1243   \draw_linewidth:n { \l_draw_default_linewidth_dim }
1244   \draw_color:n { . }
1245   \draw_nonzero_rule:
1246   \draw_cap_but:
1247   \draw_join_miter:
1248   \draw_miterlimit:n { 10 }
1249   \draw_dash_pattern:nn { } { 0cm }
1250   \hbox_set:Nw \l__draw_layer_main_box
1251 }
1252 \cs_new_protected:Npn \draw_end:
1253 {
1254   \exp_args:NNNV \hbox_set_end:
```

```

1255         \clist_set:Nn \l_draw_layers_clist \l_draw_layers_clist
1256         \__draw_layers_insert:
1257         \__draw_backend_end:
1258     \hbox_set_end:
1259     \dim_compare:nNnT \g__draw_xmin_dim = \c_max_dim
1260     {
1261         \dim_gzero:N \g__draw_xmax_dim
1262         \dim_gzero:N \g__draw_xmin_dim
1263         \dim_gzero:N \g__draw_ymax_dim
1264         \dim_gzero:N \g__draw_ymin_dim
1265     }
1266     \hbox_set:Nn \l__draw_main_box
1267     {
1268         \skip_horizontal:n { -\g__draw_xmin_dim }
1269         \box_move_down:nn { \g__draw_ymin_dim }
1270         { \box_use_drop:N \l__draw_main_box }
1271     }
1272     \box_set_ht:Nn \l__draw_main_box
1273     { \g__draw_ymax_dim - \g__draw_ymin_dim }
1274     \box_set_dp:Nn \l__draw_main_box { Opt }
1275     \box_set_wd:Nn \l__draw_main_box
1276     { \g__draw_xmax_dim - \g__draw_xmin_dim }
1277     \mode_leave_vertical:
1278     \box_use_drop:N \l__draw_main_box
1279     \group_end:
1280 }

```

(End definition for `\draw_begin:` and `\draw_end:`. These functions are documented on page ??.)

6.2 Scopes

<code>\l__draw_linewidth_dim</code>	Storage for local variables.
<code>\l__draw_fill_color_tl</code>	1281 <code>\dim_new:N \l__draw_linewidth_dim</code>
<code>\l__draw_stroke_color_tl</code>	1282 <code>\tl_new:N \l__draw_fill_color_tl</code>
	1283 <code>\tl_new:N \l__draw_stroke_color_tl</code>

(End definition for `\l__draw_linewidth_dim`, `\l__draw_fill_color_tl`, and `\l__draw_stroke_color_tl`.)

`\draw_scope_begin:` As well as the graphics (and T_EX) scope, also deal with global data structures.

<code>\draw_scope_begin:</code>	1284 <code>\cs_new_protected:Npn \draw_scope_begin:</code>
	1285 {
	1286 <code>__draw_backend_scope_begin:</code>
	1287 <code>\group_begin:</code>
	1288 <code>\dim_set_eq:NN \l__draw_linewidth_dim \g__draw_linewidth_dim</code>
	1289 <code>\draw_path_scope_begin:</code>
	1290 }
	1291 <code>\cs_new_protected:Npn \draw_scope_end:</code>
	1292 {
	1293 <code>\draw_path_scope_end:</code>
	1294 <code>\dim_gset_eq:NN \g__draw_linewidth_dim \l__draw_linewidth_dim</code>
	1295 <code>\group_end:</code>
	1296 <code>__draw_backend_scope_end:</code>
	1297 }

(End definition for \draw_scope_begin:. This function is documented on page ??.)

```
\l__draw_xmax_dim Storage for the bounding box.
\l__draw_xmin_dim 1298 \dim_new:N \l__draw_xmax_dim
\l__draw_ymax_dim 1299 \dim_new:N \l__draw_xmin_dim
\l__draw_ymin_dim 1300 \dim_new:N \l__draw_ymax_dim
1301 \dim_new:N \l__draw_ymin_dim
```

(End definition for \l__draw_xmax_dim and others.)

__draw_scope_bb_begin: The bounding box is simple: a straight group-based save and restore approach.

```
\__draw_scope_bb_end: 1302 \cs_new_protected:Npn \__draw_scope_bb_begin:
1303 {
1304   \group_begin:
1305     \dim_set_eq:NN \l__draw_xmax_dim \g__draw_xmax_dim
1306     \dim_set_eq:NN \l__draw_xmin_dim \g__draw_xmin_dim
1307     \dim_set_eq:NN \l__draw_ymax_dim \g__draw_ymax_dim
1308     \dim_set_eq:NN \l__draw_ymin_dim \g__draw_ymin_dim
1309     \__draw_reset_bb:
1310   }
1311 \cs_new_protected:Npn \__draw_scope_bb_end:
1312 {
1313   \dim_gset_eq:NN \g__draw_xmax_dim \l__draw_xmax_dim
1314   \dim_gset_eq:NN \g__draw_xmin_dim \l__draw_xmin_dim
1315   \dim_gset_eq:NN \g__draw_ymax_dim \l__draw_ymax_dim
1316   \dim_gset_eq:NN \g__draw_ymin_dim \l__draw_ymin_dim
1317   \group_end:
1318 }
```

(End definition for __draw_scope_bb_begin: and __draw_scope_bb_end:.)

\draw_suspend_begin: Suspend all parts of a drawing.

```
\draw_suspend_end: 1319 \cs_new_protected:Npn \draw_suspend_begin:
1320 {
1321   \__draw_scope_bb_begin:
1322   \draw_path_scope_begin:
1323   \draw_transform_matrix_reset:
1324   \draw_transform_shift_reset:
1325   \__draw_layers_save:
1326 }
1327 \cs_new_protected:Npn \draw_suspend_end:
1328 {
1329   \__draw_layers_restore:
1330   \draw_path_scope_end:
1331   \__draw_scope_bb_end:
1332 }
```

(End definition for \draw_suspend_begin: and \draw_suspend_end:.. These functions are documented on page ??.)

```
1333 </initex | package>
```

7 l3draw-softpath implementation

1334 $\langle *initex | package \rangle$

1335 $\langle @@=draw \rangle$

7.1 Managing soft paths

There are two linked aims in the code here. The most significant is to provide a way to modify paths, for example to shorten the ends or round the corners. This means that the path cannot be written piecemeal as specials, but rather needs to be held in macros. The second aspect that follows from this is performance: simply adding to a single macro a piece at a time will have poor performance as the list gets long so we use `\tl_build...` functions.

Each marker (operation) token takes two arguments, which makes processing more straight-forward. As such, some operations have dummy arguments, whilst others have to be split over several tokens. As the code here is at a low level, all dimension arguments are assumed to be explicit and fully-expanded.

```
\g__draw_softpath_main_tl The soft path itself.
1336 \tl_new:N \g__draw_softpath_main_tl
(End definition for \g__draw_softpath_main_tl.)

\l__draw_softpath_internal_tl The soft path itself.
1337 \tl_new:N \l__draw_softpath_internal_tl
(End definition for \l__draw_softpath_internal_tl.)

\g__draw_softpath_corners_bool Allow for optimised path use.
1338 \bool_new:N \g__draw_softpath_corners_bool
(End definition for \g__draw_softpath_corners_bool.)

\__draw_softpath_add:n
\__draw_softpath_add:o 1339 \cs_new_protected:Npn \__draw_softpath_add:n
\__draw_softpath_add:x 1340 { \tl_build_gput_right:Nn \g__draw_softpath_main_tl }
1341 \cs_generate_variant:Nn \__draw_softpath_add:n { o, x }
(End definition for \__draw_softpath_add:n.)

\__draw_softpath_use: Using and clearing is trivial.
\__draw_softpath_clear: 1342 \cs_new_protected:Npn \__draw_softpath_use:
1343 {
1344   \tl_build_get:NN \g__draw_softpath_main_tl \l__draw_softpath_internal_tl
1345   \l__draw_softpath_internal_tl
1346 }
1347 \cs_new_protected:Npn \__draw_softpath_clear:
1348 {
1349   \tl_build_gclear:N \g__draw_softpath_main_tl
1350   \bool_gset_false:N \g__draw_softpath_corners_bool
1351 }
(End definition for \__draw_softpath_use: and \__draw_softpath_clear:.)
```

<pre> \g__draw_softpath_lastx_dim \g__draw_softpath_lasty_dim </pre>	<p>For tracking the end of the path (to close it).</p> <pre> 1352 \dim_new:N \g__draw_softpath_lastx_dim 1353 \dim_new:N \g__draw_softpath_lasty_dim </pre> <p>(End definition for \g__draw_softpath_lastx_dim and \g__draw_softpath_lasty_dim.)</p>
<pre> \g__draw_softpath_move_bool </pre>	<p>Track if moving a point should update the close position.</p> <pre> 1354 \bool_new:N \g__draw_softpath_move_bool 1355 \bool_gset_true:N \g__draw_softpath_move_bool </pre> <p>(End definition for \g__draw_softpath_move_bool.)</p>
<pre> __draw_softpath_curveto:nnnnnn __draw_softpath_lineto:nn __draw_softpath_moveto:nn __draw_softpath_rectangle:nnnn __draw_softpath_roundpoint:nn __draw_softpath_roundpoint:VV </pre>	<p>The various parts of a path expressed as the appropriate soft path functions.</p> <pre> 1356 \cs_new_protected:Npn __draw_softpath_closepath: 1357 { 1358 __draw_softpath_add:x 1359 { 1360 __draw_softpath_close_op:nn 1361 { \dim_use:N \g__draw_softpath_lastx_dim } 1362 { \dim_use:N \g__draw_softpath_lasty_dim } 1363 } 1364 } 1365 \cs_new_protected:Npn __draw_softpath_curveto:nnnnnn #1#2#3#4#5#6 1366 { 1367 __draw_softpath_add:n 1368 { 1369 __draw_softpath_curveto_opi:nn {#1} {#2} 1370 __draw_softpath_curveto_opii:nn {#3} {#4} 1371 __draw_softpath_curveto_opiii:nn {#5} {#6} 1372 } 1373 } 1374 \cs_new_protected:Npn __draw_softpath_lineto:nn #1#2 1375 { 1376 __draw_softpath_add:n 1377 { __draw_softpath_lineto_op:nn {#1} {#2} } 1378 } 1379 \cs_new_protected:Npn __draw_softpath_moveto:nn #1#2 1380 { 1381 __draw_softpath_add:n 1382 { __draw_softpath_moveto_op:nn {#1} {#2} } 1383 \bool_if:NT \g__draw_softpath_move_bool 1384 { 1385 \dim_gset:Nn \g__draw_softpath_lastx_dim {#1} 1386 \dim_gset:Nn \g__draw_softpath_lasty_dim {#2} 1387 } 1388 } 1389 \cs_new_protected:Npn __draw_softpath_rectangle:nnnn #1#2#3#4 1390 { 1391 __draw_softpath_add:n 1392 { 1393 __draw_softpath_rectangle_opi:nn {#1} {#2} 1394 __draw_softpath_rectangle_opii:nn {#3} {#4} 1395 } 1396 } </pre>

```

1397 \cs_new_protected:Npn \__draw_softpath_roundpoint:nn #1#2
1398 {
1399     \__draw_softpath_add:n
1400     { \__draw_softpath_roundpoint_op:nn {#1} {#2} }
1401     \bool_gset_true:N \g__draw_softpath_corners_bool
1402 }
1403 \cs_generate_variant:Nn \__draw_softpath_roundpoint:nn { VV }

```

(End definition for __draw_softpath_curveto:nnnnnn and others.)

__draw_softpath_close_op:nn The markers for operations: all the top-level ones take two arguments. The support tokens for curves have to be different in meaning to a round point, hence being quark-like.

```

\__draw_softpath_curveto_opi:nn
\__draw_softpath_curveto_opii:nn
\__draw_softpath_curveto_opiii:nn
\__draw_softpath_lineto_op:nn
\__draw_softpath_moveto_op:nn
\__draw_softpath_roundpoint_op:nn
\__draw_softpath_rectangle_opi:nn
\__draw_softpath_rectangle_opii:nn
\__draw_softpath_curveto_opi:nnNnnNnn
\__draw_softpath_rectangle_opi:nnNnn
1404 \cs_new_protected:Npn \__draw_softpath_close_op:nn #1#2
1405 { \__draw_backend_closepath: }
1406 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nn #1#2
1407 { \__draw_softpath_curveto_opi:nnNnnNnn {#1} {#2} }
1408 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nnNnnNnn #1#2#3#4#5#6#7#8
1409 { \__draw_backend_curveto:nnnnnn {#1} {#2} {#4} {#5} {#7} {#8} }
1410 \cs_new_protected:Npn \__draw_softpath_curveto_opii:nn #1#2
1411 { \__draw_softpath_curveto_opii:nn }
1412 \cs_new_protected:Npn \__draw_softpath_curveto_opiii:nn #1#2
1413 { \__draw_softpath_curveto_opiii:nn }
1414 \cs_new_protected:Npn \__draw_softpath_lineto_op:nn #1#2
1415 { \__draw_backend_lineto:nn {#1} {#2} }
1416 \cs_new_protected:Npn \__draw_softpath_moveto_op:nn #1#2
1417 { \__draw_backend_moveto:nn {#1} {#2} }
1418 \cs_new_protected:Npn \__draw_softpath_roundpoint_op:nn #1#2 { }
1419 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nn #1#2
1420 { \__draw_softpath_rectangle_opi:nnNnn {#1} {#2} }
1421 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nnNnn #1#2#3#4#5
1422 { \__draw_backend_rectangle:nnnn {#1} {#2} {#4} {#5} }
1423 \cs_new_protected:Npn \__draw_softpath_rectangle_opii:nn #1#2 { }

```

(End definition for __draw_softpath_close_op:nn and others.)

7.2 Rounding soft path corners

The aim here is to find corner rounding points and to replace them with arcs of appropriate length. The approach is exactly that in pgf: step through, find the corners, find the supporting data, do the rounding.

\l__draw_softpath_main_tl For constructing the updated path.

```

1424 \tl_new:N \l__draw_softpath_main_tl

```

(End definition for \l__draw_softpath_main_tl.)

\l__draw_softpath_part_tl Data structures.

```

1425 \tl_new:N \l__draw_softpath_part_tl
1426 \tl_new:N \l__draw_softpath_curve_end_tl

```

(End definition for \l__draw_softpath_part_tl.)

`\l__draw_softpath_lastx_fp` Position tracking: the token list data may be entirely empty or set to a co-ordinate.

```

\l__draw_softpath_lasty_fp 1427 \fp_new:N \l__draw_softpath_lastx_fp
    \l__draw_softpath_corneri_dim 1428 \fp_new:N \l__draw_softpath_lasty_fp
    \l__draw_softpath_cornerii_dim 1429 \dim_new:N \l__draw_softpath_corneri_dim
\l__draw_softpath_first_tl 1430 \dim_new:N \l__draw_softpath_cornerii_dim
\l__draw_softpath_move_tl 1431 \tl_new:N \l__draw_softpath_first_tl
    1432 \tl_new:N \l__draw_softpath_move_tl

```

(End definition for `\l__draw_softpath_lastx_fp` and others.)

`\c__draw_softpath_arc_fp` The magic constant.

```
1433 \fp_const:Nn \c__draw_softpath_arc_fp { 4/3 * (sqrt(2) - 1) }
```

(End definition for `\c__draw_softpath_arc_fp`.)

```

    \__draw_softpath_round_corners: 1434 \cs_new_protected:Npn \__draw_softpath_round_corners:
    \__draw_softpath_round_loop:Nnn 1435 {
    \__draw_softpath_round_action:nn 1436   \bool_if:NT \g__draw_softpath_corners_bool
    \__draw_softpath_round_action:Nnn 1437   {
    \__draw_softpath_round_action_curveto:NnnNnn 1438     \group_begin:
    \__draw_softpath_round_roundpoint:NnnNnnNnn 1439       \tl_clear:N \l__draw_softpath_main_tl
    \__draw_softpath_round_calc:NnnNnn 1440       \tl_clear:N \l__draw_softpath_part_tl
    \__draw_softpath_round_calc:nnnnnn 1441       \fp_zero:N \l__draw_softpath_lastx_fp
    \__draw_softpath_round_calc:fVnnnn 1442       \fp_zero:N \l__draw_softpath_lasty_fp
    \__draw_softpath_round_calc:nnnnw 1443       \tl_clear:N \l__draw_softpath_first_tl
    \__draw_softpath_round_close:nn 1444       \tl_clear:N \l__draw_softpath_move_tl
    \__draw_softpath_round_close:w 1445       \tl_build_get:NN \g__draw_softpath_main_tl \l__draw_softpath_internal_tl
    \__draw_softpath_round_end: 1446       \exp_after:wN \__draw_softpath_round_loop:Nnn
    1447         \l__draw_softpath_internal_tl
    1448         \q__draw_recursion_tail ? ?
    1449         \q__draw_recursion_stop
    1450       \group_end:
    1451     }
    1452   \bool_gset_false:N \g__draw_softpath_corners_bool
    1453 }

```

The loop can take advantage of the fact that all soft path operations are made up of a token followed by two arguments. At this stage, there is a simple split: have we round a round point. If so, is there any actual rounding to be done: if the arcs have come through zero, just ignore it. In cases where we are not at a corner, we simply move along the path, allowing for any new part starting due to a moveto.

```

1454 \cs_new_protected:Npn \__draw_softpath_round_loop:Nnn #1#2#3
1455 {
1456   \__draw_if_recursion_tail_stop_do:Nn #1 { \__draw_softpath_round_end: }
1457   \token_if_eq_meaning:NNTF #1 \__draw_softpath_roundpoint_op:nn
1458   { \__draw_softpath_round_action:nn {#2} {#3} }
1459   {
1460     \tl_if_empty:NT \l__draw_softpath_first_tl
1461     { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1462     \fp_set:Nn \l__draw_softpath_lastx_fp {#2}
1463     \fp_set:Nn \l__draw_softpath_lasty_fp {#3}

```



```

1464     \token_if_eq_meaning:NNTF #1 \__draw_softpath_moveto_op:nn
1465     {
1466         \tl_put_right:No \l__draw_softpath_main_tl
1467         \l__draw_softpath_move_tl
1468         \tl_put_right:No \l__draw_softpath_main_tl
1469         \l__draw_softpath_part_tl
1470         \tl_set:Nn \l__draw_softpath_move_tl { #1 {#2} {#3} }
1471         \tl_clear:N \l__draw_softpath_first_tl
1472         \tl_clear:N \l__draw_softpath_part_tl
1473     }
1474     { \tl_put_right:Nn \l__draw_softpath_part_tl { #1 {#2} {#3} } }
1475     \__draw_softpath_round_loop:Nnn
1476 }
1477 }
1478 \cs_new_protected:Npn \__draw_softpath_round_action:nn #1#2
1479 {
1480     \dim_set:Nn \l__draw_softpath_corneri_dim {#1}
1481     \dim_set:Nn \l__draw_softpath_cornerii_dim {#2}
1482     \bool_lazy_and:nnTF
1483     { \dim_compare_p:nNn \l__draw_softpath_corneri_dim = { Opt } }
1484     { \dim_compare_p:nNn \l__draw_softpath_cornerii_dim = { Opt } }
1485     { \__draw_softpath_round_loop:Nnn }
1486     { \__draw_softpath_round_action:Nnn }
1487 }

```

We now have a round point to work on and have grabbed the next item in the path. There are only a few cases where we have to do anything. Each of them is picked up by looking for the appropriate action.

```

1488 \cs_new_protected:Npn \__draw_softpath_round_action:Nnn #1#2#3
1489 {
1490     \tl_if_empty:NT \l__draw_softpath_first_tl
1491     { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1492     \token_if_eq_meaning:NNTF #1 \__draw_softpath_curveto_opi:nn
1493     { \__draw_softpath_round_action_curveto:NnnNnn }
1494     {
1495         \token_if_eq_meaning:NNTF #1 \__draw_softpath_close_op:nn
1496         { \__draw_softpath_round_action_close: }
1497         {
1498             \token_if_eq_meaning:NNTF #1 \__draw_softpath_lineto_op:nn
1499             { \__draw_softpath_round_lookahead:NnnNnn }
1500             { \__draw_softpath_round_loop:Nnn }
1501         }
1502     }
1503     #1 {#2} {#3}
1504 }

```

For a curve, we collect the two control points then move on to grab the end point and add the curve there: the second control point becomes our starter.

```

1505 \cs_new_protected:Npn \__draw_softpath_round_action_curveto:NnnNnn
1506 #1#2#3#4#5#6
1507 {
1508     \tl_put_right:Nn \l__draw_softpath_part_tl
1509     { #1 {#2} {#3} #4 {#5} {#6} }
1510     \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1511     \fp_set:Nn \l__draw_softpath_lasty_fp {#6}

```

```

1512     \__draw_softpath_round_lookahead:NnnNnn
1513   }
1514 \cs_new_protected:Npn \__draw_softpath_round_action_close:
1515 {
1516   \bool_lazy_and:nnTF
1517     { ! \tl_if_empty_p:N \l__draw_softpath_first_tl }
1518     { ! \tl_if_empty_p:N \l__draw_softpath_move_tl }
1519   {
1520     \exp_after:wN \__draw_softpath_round_close:nn
1521     \l__draw_softpath_first_tl
1522   }
1523   { \__draw_softpath_round_loop:Nnn }
1524 }

```

At this stage we have a current (sub)operation (#1) and the next operation (#4), and can therefore decide whether to round or not. In the case of yet another rounding marker, we have to look a bit further ahead.

```

1525 \cs_new_protected:Npn \__draw_softpath_round_lookahead:NnnNnn #1#2#3#4#5#6
1526 {
1527   \bool_lazy_any:nTF
1528   {
1529     { \token_if_eq_meaning_p:NN #4 \__draw_softpath_lineto_op:nn }
1530     { \token_if_eq_meaning_p:NN #4 \__draw_softpath_curveto_opi:nn }
1531     { \token_if_eq_meaning_p:NN #4 \__draw_softpath_close_op:nn }
1532   }
1533   {
1534     \__draw_softpath_round_calc:NnnNnn
1535     \__draw_softpath_round_loop:Nnn
1536     {#5} {#6}
1537   }
1538   {
1539     \token_if_eq_meaning:NNTF #4 \__draw_softpath_roundpoint_op:nn
1540     { \__draw_softpath_round_roundpoint:NnnNnnNnn }
1541     { \__draw_softpath_round_loop:Nnn }
1542   }
1543   #1 {#2} {#3}
1544   #4 {#5} {#6}
1545 }
1546 \cs_new_protected:Npn \__draw_softpath_round_roundpoint:NnnNnnNnn
1547 #1#2#3#4#5#6#7#8#9
1548 {
1549   \__draw_softpath_round_calc:NnnNnn
1550   \__draw_softpath_round_loop:Nnn
1551   {#8} {#9}
1552   #1 {#2} {#3}
1553   #4 {#5} {#6} #7 {#8} {#9}
1554 }

```

We now have all of the data needed to construct a rounded corner: all that is left to do is to work out the detail! At this stage, we have details of where the corner itself is (#5, #6), and where the next point is (#2, #3). There are two types of calculations to do. First, we need to interpolate from those two points in the direction of the corner, in order to work out where the curve we are adding will start and end. From those, plus the points we already have, we work out where the control points will lie. All of this is done

in an expansion to avoid multiple calls to `\tl_put_right:Nx`. The end point of the line is worked out up-front and saved: we need that if dealing with a close-path operation.

```

1555 \cs_new_protected:Npn \__draw_softpath_round_calc:NnnNnn #1#2#3#4#5#6
1556 {
1557   \tl_set:Nx \l__draw_softpath_curve_end_tl
1558   {
1559     \draw_point_interpolate_distance:nnn
1560     \l__draw_softpath_cornerii_dim
1561     { #5 , #6 } { #2 , #3 }
1562   }
1563   \tl_put_right:Nx \l__draw_softpath_part_tl
1564   {
1565     \exp_not:N #4
1566     \__draw_softpath_round_calc:fVnnnn
1567     {
1568       \draw_point_interpolate_distance:nnn
1569       \l__draw_softpath_corneri_dim
1570       { #5 , #6 }
1571       {
1572         \l__draw_softpath_lastx_fp ,
1573         \l__draw_softpath_lasty_fp
1574       }
1575     }
1576     \l__draw_softpath_curve_end_tl
1577     {#5} {#6} {#2} {#3}
1578   }
1579   \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1580   \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1581   #1
1582 }

```

At this stage we have the two curve end points, but they are in co-ordinate form. So we split them up (with some more reordering).

```

1583 \cs_new:Npn \__draw_softpath_round_calc:nnnnnn #1#2#3#4#5#6
1584 {
1585   \__draw_softpath_round_calc:nnnnw {#3} {#4} {#5} {#6}
1586   #1 \s__draw_mark #2 \s__draw_stop
1587 }
1588 \cs_generate_variant:Nn \__draw_softpath_round_calc:nnnnnn { fV }

```

The calculations themselves are relatively straight-forward, as we use a quadratic Bézier curve.

```

1589 \cs_new:Npn \__draw_softpath_round_calc:nnnnw
1590 #1#2#3#4 #5 , #6 \s__draw_mark #7 , #8 \s__draw_stop
1591 {
1592   {#5} {#6}
1593   \exp_not:N \__draw_softpath_curveto_opi:nn
1594   {
1595     \fp_to_dim:n
1596     { #5 + \c__draw_softpath_arc_fp * ( #1 - #5 ) }
1597   }
1598   {
1599     \fp_to_dim:n
1600     { #6 + \c__draw_softpath_arc_fp * ( #2 - #6 ) }

```

```

1601     }
1602     \exp_not:N \__draw_softpath_curveto_opii:nn
1603     {
1604         \fp_to_dim:n
1605         { #7 + \c__draw_softpath_arc_fp * ( #1 - #7 ) }
1606     }
1607     {
1608         \fp_to_dim:n
1609         { #8 + \c__draw_softpath_arc_fp * ( #2 - #8 ) }
1610     }
1611     \exp_not:N \__draw_softpath_curveto_opiii:nn
1612     {#7} {#8}
1613 }

```

To deal with a close-path operation, we need to do some manipulation. It needs to be treated as a line operation for rounding, and then have the close path operation re-added at the point where the curve ends. That means saving the end point in the calculation step (see earlier), and shuffling a lot.

```

1614 \cs_new_protected:Npn \__draw_softpath_round_close:nn #1#2
1615 {
1616     \use:x
1617     {
1618         \__draw_softpath_round_calc:NnnNnn
1619         {
1620             \tl_set:Nx \exp_not:N \l__draw_softpath_move_tl
1621             {
1622                 \__draw_softpath_moveto_op:nn
1623                 \exp_not:N \exp_after:wN
1624                 \exp_not:N \__draw_softpath_round_close:w
1625                 \exp_not:N \l__draw_softpath_curve_end_tl
1626                 \s__draw_stop
1627             }
1628             \use:x
1629             {
1630                 \exp_not:N \exp_not:N \exp_not:N \use_i:nnnn
1631                 {
1632                     \__draw_softpath_round_loop:Nnn
1633                     \__draw_softpath_close_op:nn
1634                     \exp_not:N \exp_after:wN
1635                     \exp_not:N \__draw_softpath_round_close:w
1636                     \exp_not:N \l__draw_softpath_curve_end_tl
1637                     \s__draw_stop
1638                 }
1639             }
1640             }
1641             {#1} {#2}
1642             \__draw_softpath_lineto_op:nn
1643             \exp_after:wN \use_none:n \l__draw_softpath_move_tl
1644         }
1645     }
1646 \cs_new:Npn \__draw_softpath_round_close:w #1 , #2 \s__draw_stop { {#1} {#2} }

```

Tidy up the parts of the path, complete the built token list and put it back into action.

```

1647 \cs_new_protected:Npn \__draw_softpath_round_end:
1648 {

```

```

1649 \tl_put_right:No \l__draw_softpath_main_tl
1650 \l__draw_softpath_move_tl
1651 \tl_put_right:No \l__draw_softpath_main_tl
1652 \l__draw_softpath_part_tl
1653 \tl_build_gclear:N \g__draw_softpath_main_tl
1654 \__draw_softpath_add:o \l__draw_softpath_main_tl
1655 }

```

(End definition for `__draw_softpath_round_corners:` and others.)

```

1656 </initex | package>

```

8 l3draw-state implementation

```

1657 <*initex | package>

```

```

1658 <@@=draw>

```

This sub-module covers more-or-less the same ideas as `pgfcoregraphicstate.code.tex`.

At present, equivalents of the following are currently absent:

- `\pgfsetinnerlinewidth`, `\pgfinnerlinewidth`, `\pgfsetinnerstrokecolor`, `\pgfsetinnerstrokecolor`

Likely to be added on further work is done on paths/stroking.

`\g__draw_linewidth_dim` Linewidth for strokes: global as the scope for this relies on the graphics state. The inner line width is used for places where two lines are used.

```

1659 \dim_new:N \g__draw_linewidth_dim

```

(End definition for `\g__draw_linewidth_dim`.)

`\l_draw_default_linewidth_dim` A default: this is used at the start of every drawing.

```

1660 \dim_new:N \l_draw_default_linewidth_dim
1661 \dim_set:Nn \l_draw_default_linewidth_dim { 0.4pt }

```

(End definition for `\l_draw_default_linewidth_dim`. This variable is documented on page ??.)

`\draw_linewidth:n` Set the linewidth: we need a wrapper as this has to pass to the driver layer.

```

1662 \cs_new_protected:Npn \draw_linewidth:n #1
1663 {
1664   \dim_gset:Nn \g__draw_linewidth_dim { \fp_to_dim:n {#1} }
1665   \__draw_backend_linewidth:n \g__draw_linewidth_dim
1666 }

```

(End definition for `\draw_linewidth:n`. This function is documented on page ??.)

`\draw_dash_pattern:nn` Evaluated all of the list and pass it to the driver layer.

```

\l__draw_tmp_seq
1667 \cs_new_protected:Npn \draw_dash_pattern:nn #1#2
1668 {
1669   \group_begin:
1670     \seq_set_from_clist:Nn \l__draw_tmp_seq {#1}
1671     \seq_set_map:NNn \l__draw_tmp_seq \l__draw_tmp_seq
1672       { \fp_to_dim:n {##1} }
1673     \use:x
1674     {
1675       \__draw_backend_dash_pattern:nn
1676       { \seq_use:Nn \l__draw_tmp_seq { , } }

```

```

1677         { \fp_to_dim:n {#2} }
1678     }
1679     \group_end:
1680 }
1681 \seq_new:N \l__draw_tmp_seq

```

(End definition for \draw_dash_pattern:nn and \l__draw_tmp_seq. This function is documented on page ??.)

\draw_miterlimit:n Pass through to the driver layer.

```

1682 \cs_new_protected:Npn \draw_miterlimit:n #1
1683 { \exp_args:Nx \__draw_backend_miterlimit:n { \fp_eval:n {#1} } }

```

(End definition for \draw_miterlimit:n. This function is documented on page ??.)

\draw_cap_but: All straight wrappers.

```

\draw_cap_rectangle: 1684 \cs_new_protected:Npn \draw_cap_but: { \__draw_backend_cap_but: }
\draw_cap_round:      1685 \cs_new_protected:Npn \draw_cap_rectangle: { \__draw_backend_cap_rectangle: }
\draw_evenodd_rule:   1686 \cs_new_protected:Npn \draw_cap_round: { \__draw_backend_cap_round: }
\draw_nonzero_rule:   1687 \cs_new_protected:Npn \draw_evenodd_rule: { \__draw_backend_evenodd_rule: }
\draw_join_bevel:     1688 \cs_new_protected:Npn \draw_nonzero_rule: { \__draw_backend_nonzero_rule: }
\draw_join_miter:     1689 \cs_new_protected:Npn \draw_join_bevel: { \__draw_backend_join_bevel: }
\draw_join_round:     1690 \cs_new_protected:Npn \draw_join_miter: { \__draw_backend_join_miter: }
1691 \cs_new_protected:Npn \draw_join_round: { \__draw_backend_join_round: }

```

(End definition for \draw_cap_but: and others. These functions are documented on page ??.)

\l__draw_color_tmp_tl Scratch space.

```

1692 \tl_new:N \l__draw_color_tmp_tl

```

(End definition for \l__draw_color_tmp_tl.)

\draw_color:n Much the same as for core color support but calling the relevant driver-level function.

```

\draw_color_fill:n    1693 \cs_new_eq:NN \draw_color:n \color_select:n
\draw_color_stroke:n  1694 \cs_new_protected:Npn \draw_color_fill:n #1
\__draw_color:nn      1695 { \__draw_color:nn { fill } {#1} }
\__draw_color:nnn     1696 \cs_new_protected:Npn \draw_color_stroke:n #1
1697 { \__draw_color:nn { stroke } {#1} }
1698 \cs_new_protected:Npn \__draw_color:nn #1#2
1699 {
1700     \color_export:nnN {#2} { backend } \l__draw_color_tmp_tl
1701     \exp_after:wN \__draw_color:nnn \l__draw_color_tmp_tl {#1}
1702 }
1703 \cs_new_protected:Npn \__draw_color:nnn #1#2#3
1704 { \use:c { __draw_backend_color_ #3 _ #1 :n } {#2} }

```

(End definition for \draw_color:n and others. These functions are documented on page ??.)

```

1705 </initex | package>

```

9 l3draw-transforms implementation

1706 \langle *initex | package \rangle

1707 \langle @@=draw \rangle

This sub-module covers more-or-less the same ideas as `pgfcoretransformations.code.tex`. At present, equivalents of the following are currently absent:

- `\pgfgettransform`, `\pgfgettransformentries`: Awaiting use cases.
- `\pgftransformlineatime`, `\pgftransformarcaxesatime`, `\pgftransformcurveatime`: Need to look at the use cases for these to fully understand them.
- `\pgftransformarrow`: Likely to be done when other arrow functions are added.
- `\pgfflowlevelsynccm`, `\pgfflowlevel`: Likely to be added when use cases are encountered in other parts of the code.

`\l__draw_matrix_active_bool` An internal flag to avoid redundant calculations.

1708 `\bool_new:N \l__draw_matrix_active_bool`

(End definition for `\l__draw_matrix_active_bool`.)

`\l__draw_matrix_a_fp` The active matrix and shifts.

`\l__draw_matrix_b_fp` 1709 `\fp_new:N \l__draw_matrix_a_fp`

`\l__draw_matrix_c_fp` 1710 `\fp_new:N \l__draw_matrix_b_fp`

`\l__draw_xshift_dim` 1711 `\fp_new:N \l__draw_matrix_c_fp`

`\l__draw_yshift_dim` 1712 `\fp_new:N \l__draw_matrix_d_fp`

1713 `\dim_new:N \l__draw_xshift_dim`

1714 `\dim_new:N \l__draw_yshift_dim`

(End definition for `\l__draw_matrix_a_fp` and others.)

`\draw_transform_matrix_reset:` Fast resetting.

`\draw_transform_shift_reset:` 1715 `\cs_new_protected:Npn \draw_transform_matrix_reset:`

1716 `{`

1717 `\fp_set:Nn \l__draw_matrix_a_fp { 1 }`

1718 `\fp_zero:N \l__draw_matrix_b_fp`

1719 `\fp_zero:N \l__draw_matrix_c_fp`

1720 `\fp_set:Nn \l__draw_matrix_d_fp { 1 }`

1721 `}`

1722 `\cs_new_protected:Npn \draw_transform_shift_reset:`

1723 `{`

1724 `\dim_zero:N \l__draw_xshift_dim`

1725 `\dim_zero:N \l__draw_yshift_dim`

1726 `}`

1727 `\draw_transform_matrix_reset:`

1728 `\draw_transform_shift_reset:`

(End definition for `\draw_transform_matrix_reset:` and `\draw_transform_shift_reset:`. These functions are documented on page ??.)

`\draw_transform_matrix_absolute:nnnn` Setting the transform matrix is straight-forward, with just a bit of expansion to sort out.
`\draw_transform_shift_absolute:n` With the mechanism active, the identity matrix is set.

```

1729 \cs_new_protected:Npn \draw_transform_matrix_absolute:nnnn #1#2#3#4
1730 {
1731   \fp_set:Nn \l__draw_matrix_a_fp {#1}
1732   \fp_set:Nn \l__draw_matrix_b_fp {#2}
1733   \fp_set:Nn \l__draw_matrix_c_fp {#3}
1734   \fp_set:Nn \l__draw_matrix_d_fp {#4}
1735   \bool_lazy_all:nTF
1736   {
1737     { \fp_compare_p:nNn \l__draw_matrix_a_fp = \c_one_fp }
1738     { \fp_compare_p:nNn \l__draw_matrix_b_fp = \c_zero_fp }
1739     { \fp_compare_p:nNn \l__draw_matrix_c_fp = \c_zero_fp }
1740     { \fp_compare_p:nNn \l__draw_matrix_d_fp = \c_one_fp }
1741   }
1742   { \bool_set_false:N \l__draw_matrix_active_bool }
1743   { \bool_set_true:N \l__draw_matrix_active_bool }
1744 }
1745 \cs_new_protected:Npn \draw_transform_shift_absolute:n #1
1746 {
1747   \__draw_point_process:nn
1748   { \__draw_transform_shift_absolute:nn } {#1}
1749 }
1750 \cs_new_protected:Npn \__draw_transform_shift_absolute:nn #1#2
1751 {
1752   \dim_set:Nn \l__draw_xshift_dim {#1}
1753   \dim_set:Nn \l__draw_yshift_dim {#2}
1754 }

```

(End definition for `\draw_transform_matrix_absolute:nnnn`, `\draw_transform_shift_absolute:n`, and `__draw_transform_shift_absolute:nn`. These functions are documented on page ??.)

`\draw_transform_matrix:nnnn` Much the same story for adding to an existing matrix, with a bit of pre-expansion so
`__draw_transform:nnnn` that the calculation uses “frozen” values.

```

1755 \cs_new_protected:Npn \draw_transform_matrix:nnnn #1#2#3#4
1756 {
1757   \use:x
1758   {
1759     \__draw_transform:nnnn
1760     { \fp_eval:n {#1} }
1761     { \fp_eval:n {#2} }
1762     { \fp_eval:n {#3} }
1763     { \fp_eval:n {#4} }
1764   }
1765 }
1766 \cs_new_protected:Npn \__draw_transform:nnnn #1#2#3#4
1767 {
1768   \use:x
1769   {
1770     \draw_transform_matrix_absolute:nnnn
1771     { #1 * \l__draw_matrix_a_fp + #2 * \l__draw_matrix_c_fp }
1772     { #1 * \l__draw_matrix_b_fp + #2 * \l__draw_matrix_d_fp }
1773     { #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_c_fp }
1774     { #3 * \l__draw_matrix_b_fp + #4 * \l__draw_matrix_d_fp }

```



```

1775     }
1776   }
1777   \cs_new_protected:Npn \draw_transform_shift:n #1
1778   {
1779     \__draw_point_process:nn
1780     { \__draw_transform_shift:nn } {#1}
1781   }
1782   \cs_new_protected:Npn \__draw_transform_shift:nn #1#2
1783   {
1784     \dim_set:Nn \l__draw_xshift_dim { \l__draw_xshift_dim + #1 }
1785     \dim_set:Nn \l__draw_yshift_dim { \l__draw_yshift_dim + #2 }
1786   }

```

(End definition for \draw_transform_matrix:nnnn and others. These functions are documented on page ??.)

<pre> \draw_transform_matrix_invert: __draw_transform_invert:n __draw_transform_invert:f \draw_transform_shift_invert: </pre>	<p>Standard mathematics: calculate the inverse matrix and use that, then undo the shifts.</p> <pre> 1787 \cs_new_protected:Npn \draw_transform_matrix_invert: 1788 { 1789 \bool_if:NT \l__draw_matrix_active_bool 1790 { 1791 __draw_transform_invert:f 1792 { 1793 \fp_eval:n 1794 { 1795 1 / 1796 (1797 \l__draw_matrix_a_fp * \l__draw_matrix_d_fp 1798 - \l__draw_matrix_b_fp * \l__draw_matrix_c_fp 1799) 1800 } 1801 } 1802 } 1803 } 1804 \cs_new_protected:Npn __draw_transform_invert:n #1 1805 { 1806 \fp_set:Nn \l__draw_matrix_a_fp 1807 { \l__draw_matrix_d_fp * #1 } 1808 \fp_set:Nn \l__draw_matrix_b_fp 1809 { -\l__draw_matrix_b_fp * #1 } 1810 \fp_set:Nn \l__draw_matrix_c_fp 1811 { -\l__draw_matrix_c_fp * #1 } 1812 \fp_set:Nn \l__draw_matrix_d_fp 1813 { \l__draw_matrix_a_fp * #1 } 1814 } 1815 \cs_generate_variant:Nn __draw_transform_invert:n { f } 1816 \cs_new_protected:Npn \draw_transform_shift_invert: 1817 { 1818 \dim_set:Nn \l__draw_xshift_dim { -\l__draw_xshift_dim } 1819 \dim_set:Nn \l__draw_yshift_dim { -\l__draw_yshift_dim } 1820 } </pre>
---	--

(End definition for \draw_transform_matrix_invert:, __draw_transform_invert:n, and \draw_transform_shift_invert:. These functions are documented on page ??.)

`\draw_transform_triangle:nnn` Simple maths to move the canvas origin to #1 and the two axes to #2 and #3.

```

1821 \cs_new_protected:Npn \draw_transform_triangle:nnn #1#2#3
1822 {
1823   \__draw_point_process:nnn
1824   {
1825     \__draw_point_process:nn
1826     { \__draw_tranform_triangle:nnnnnn }
1827     {#1}
1828   }
1829   {#2} {#3}
1830 }
1831 \cs_new_protected:Npn \__draw_tranform_triangle:nnnnnn #1#2#3#4#5#6
1832 {
1833   \use:x
1834   {
1835     \draw_transform_matrix_absolute:nnnn
1836     { #3 - #1 }
1837     { #4 - #2 }
1838     { #5 - #1 }
1839     { #6 - #2 }
1840     \draw_transform_shift_absolute:n { #1 , #2 }
1841   }
1842 }

```

(End definition for `\draw_transform_triangle:nnn`. This function is documented on page ??.)

`\draw_transform_scale:n` Lots of shortcuts.

```

\draw_transform_xscale:n 1843 \cs_new_protected:Npn \draw_transform_scale:n #1
\draw_transform_yscale:n 1844 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { #1 } }
\draw_transform_xshift:n 1845 \cs_new_protected:Npn \draw_transform_xscale:n #1
\draw_transform_yshift:n 1846 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { 1 } }
\draw_transform_xslant:n 1847 \cs_new_protected:Npn \draw_transform_yscale:n #1
\draw_transform_yslant:n 1848 { \draw_transform_matrix:nnnn { 1 } { 0 } { 0 } { #1 } }
1849 \cs_new_protected:Npn \draw_transform_xshift:n #1
1850 { \draw_transform_shift:n { #1 , Opt } }
1851 \cs_new_protected:Npn \draw_transform_yshift:n #1
1852 { \draw_transform_shift:n { Opt , #1 } }
1853 \cs_new_protected:Npn \draw_transform_xslant:n #1
1854 { \draw_transform_matrix:nnnn { 1 } { 0 } { #1 } { 1 } }
1855 \cs_new_protected:Npn \draw_transform_yslant:n #1
1856 { \draw_transform_matrix:nnnn { 1 } { #1 } { 0 } { 1 } }

```

(End definition for `\draw_transform_scale:n` and others. These functions are documented on page ??.)

`\draw_transform_rotate:n` Slightly more involved: evaluate the angle only once, and the sine and cosine only once.

```

\__draw_transform_rotate:n 1857 \cs_new_protected:Npn \draw_transform_rotate:n #1
\__draw_transform_rotate:f 1858 { \__draw_transform_rotate:f { \fp_eval:n {#1} } }
\__draw_transform_rotate:nn 1859 \cs_new_protected:Npn \__draw_transform_rotate:n #1
\__draw_transform_rotate:ff 1860 {
1861   \__draw_transform_rotate:ff
1862   { \fp_eval:n { cosd(#1) } }
1863   { \fp_eval:n { sind(#1) } }
1864 }
1865 \cs_generate_variant:Nn \__draw_transform_rotate:n { f }

```

```

1866 \cs_new_protected:Npn \__draw_transform_rotate:nn #1#2
1867   { \draw_transform_matrix:nnnn {#1} {#2} { -#2 } { #1 } }
1868 \cs_generate_variant:Nn \__draw_transform_rotate:nn { ff }

(End definition for \draw_transform_rotate:n, \__draw_transform_rotate:n, and \__draw_transform_
rotate:nn. This function is documented on page ??.)

1869 </initex | package>

```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

B

`\begin` . . . 174, 789, 1041, 1045, 1069, 1072

bool commands:

`\bool_gset_eq:Nn` 770

`\bool_gset_false:N` 1350, 1452

`\bool_gset_true:N` 1355, 1401

`\bool_if:NTF`

. 24, 118, 197, 236, 685, 701,

702, 706, 1164, 1196, 1383, 1436, 1789

`\bool_lazy_all:nTF` 1735

`\bool_lazy_and:nnTF`

. 228, 680, 1482, 1516

`\bool_lazy_any:nTF` 1527

`\bool_lazy_or:nnTF` . 561, 656, 689, 694

`\bool_new:N` . 89, 223, 644, 645, 646,

647, 648, 748, 1220, 1338, 1354, 1708

`\bool_set_eq:Nn` 762

`\bool_set_false:N`

. 99, 231, 667, 668, 669, 688, 1742

`\bool_set_true:N` 101,

232, 673, 711, 715, 716, 1239, 1743

box commands:

`\box_dp:N` 20, 70

`\box_gset_eq:Nn` 159

`\box_gset_wd:Nn` 103, 136

`\box_ht:N` 20, 72

`\box_if_exist:NTF` 96

`\box_move_down:nn` 1269

`\box_move_up:nn` 54

`\box_new:N` 16, 83, 84, 1221, 1222

`\box_set_dp:Nn` 58, 1274

`\box_set_eq:Nn` 148

`\box_set_ht:Nn` 57, 1272

`\box_set_wd:Nn` 59, 131, 1275

`\box_use_drop:N`

. 55, 60, 105, 132, 137, 1270, 1278

`\box_wd:N` 20, 69, 71

C

clist commands:

`\clist_map_inline:Nn` . . 127, 144, 155

`\clist_map_inline:nn` 670

`\clist_new:N` 90, 92

`\clist_set:Nn` 91, 1255

coffin commands:

`\coffin_typeset:Nnnnn` 67

`\coffin_wd:N` 69

color commands:

`\color_export:nnN` 1700

`\color_select:n` 1693

cs commands:

`\cs_generate_variant:Nn`

. 423, 610, 643, 848, 856, 898,

917, 924, 932, 939, 948, 954, 966,

969, 987, 1005, 1013, 1019, 1040,

1054, 1068, 1089, 1124, 1143, 1156,

1341, 1403, 1588, 1815, 1865, 1868

`\cs_if_exist:NTF` 672

`\cs_if_exist_use:NTF` . . 406, 415, 675

`\cs_new:Npn` 514, 524, 534, 544,

793, 799, 801, 803, 810, 812, 814,

822, 824, 827, 836, 841, 844, 846,

849, 850, 852, 854, 857, 859, 865,

874, 880, 890, 899, 905, 910, 918,

925, 933, 940, 949, 955, 961, 967,

970, 976, 985, 988, 994, 999, 1006,

1014, 1020, 1026, 1032, 1046, 1055,

1061, 1073, 1075, 1084, 1114, 1116,

1125, 1130, 1144, 1146, 1148, 1157,

1162, 1189, 1194, 1583, 1589, 1646

`\cs_new_eq:Nn` 1693

`\cs_new_protected:Npn` . 17, 22, 63,

78, 93, 116, 125, 142, 153, 187, 209,

216, 224, 234, 243, 249, 255, 261,

268, 279, 287, 292, 294, 296, 305,

312, 348, 350, 361, 367, 397, 424,

453, 459, 465, 470, 478, 487, 492,

500, 555, 557, 570, 577, 586, 592,

594, 604, 611, 617, 624, 649, 654,

665, 709, 713, 718, 725, 749, 767,

1096, 1098, 1100, 1102, 1106, 1224,

1231, 1252, 1284, 1291, 1302, 1311,

1319, 1327, 1339, 1342, 1347, 1356,

1365, 1374, 1379, 1389, 1397, 1404,

1406, 1408, 1410, 1412, 1414, 1416,

1418, 1419, 1421, 1423, 1434, 1454,

1478, 1488, 1505, 1514, 1525, 1546,

1555, 1614, 1647, 1662, 1667, 1682,

1684, 1685, 1686, 1687, 1688, 1689,

1690, 1691, 1694, 1696, 1698, 1703,

1715, 1722, 1729, 1745, 1750, 1755,

1766, 1777, 1782, 1787, 1804, 1816,

1821, 1831, 1843, 1845, 1847, 1849,

1851, 1853, 1855, 1857, 1859, 1866

D

dim commands:

`\dim_abs:n` 599, 600
`\dim_compare:nNnTF` 606, 613, 727, 1259
`\dim_compare_p:nNn` 229, 230, 1483, 1484
`\dim_eval:n` 599, 600
`\dim_gset:Nn` 189, 191,
193, 195, 199, 201, 203, 205, 211,
212, 213, 214, 218, 219, 729, 1226,
1227, 1228, 1229, 1385, 1386, 1664
`\dim_gset_eq:NN`
.... 774, 775, 776, 777, 778, 779,
780, 781, 1294, 1313, 1314, 1315, 1316
`\dim_gzero:N` .. 1261, 1262, 1263, 1264
`\dim_max:nn` 190, 194, 200, 204
`\dim_min:nn` 192, 196, 202, 206
`\dim_new:N` 181,
182, 183, 184, 185, 186, 221, 222,
740, 741, 742, 743, 744, 745, 746,
747, 1090, 1091, 1092, 1093, 1094,
1095, 1216, 1217, 1218, 1219, 1281,
1298, 1299, 1300, 1301, 1352, 1353,
1429, 1430, 1659, 1660, 1713, 1714
`\dim_set:Nn` 226,
227, 1108, 1109, 1480, 1481, 1661,
1752, 1753, 1784, 1785, 1818, 1819
`\dim_set_eq:NN`
.... 752, 753, 754, 755, 756, 757,
758, 759, 1288, 1305, 1306, 1307, 1308
`\dim_step_inline:nnnn` 626, 634
`\dim_use:N` .. 727, 732, 734, 1361, 1362
`\dim_zero:N` 1724, 1725
`\c_max_dim` 211, 212, 213,
214, 727, 1226, 1227, 1228, 1229, 1259

draw commands:

`\l_draw_bb_update_bool`
.... 24, 197, 681, 688, 1220, 1239
`\draw_begin:` 1231
`\draw_box_use:N` 17
`\draw_cap_but:` 1246, 1684
`\draw_cap_rectangle:` 1684
`\draw_cap_round:` 1684
`\draw_coffin_use:Nnn` 63
`\draw_color:n` 1244, 1693
`\draw_color_fill:n` 1693
`\draw_color_stroke:n` 1693
`\draw_dash_pattern:nn` ... 1249, 1667
`\l_draw_default_linewidth_dim` ...
.... 108, 1243, 1660
`\draw_end:` 1231
`\draw_evenodd_rule:` 1684
`\draw_join_bevel:` 1684
`\draw_join_miter:` 1247, 1684
`\draw_join_round:` 1684

`\draw_layer_begin:n` 93
`\draw_layer_end:` 93
`\draw_layer_new:n` 78
`\l_draw_layers_clist`
.... 90, 127, 144, 155, 1255
`\draw_linewidth:n` 108, 1243, 1662
`\draw_miterlimit:n` 1248, 1682
`\draw_nonzero_rule:` 1245, 1684
`\draw_path_arc:nnn` 348, 490
`\draw_path_arc:nnnn` 348
`\draw_path_arc_axes:nnnn` 487
`\draw_path_canvas_curveto:nnn` .. 292
`\draw_path_canvas_lineto:n` 292
`\draw_path_canvas_moveto:n` 292
`\draw_path_circle:nn` 555
`\draw_path_close:` 287, 583
`\draw_path_corner_arc:nn` 224
`\draw_path_curveto:nn` 305
`\draw_path_curveto:nnn` 243
`\draw_path_ellipse:nnn` 492, 556
`\draw_path_grid:nnnn` 594
`\draw_path_lineto:n`
.... 243, 580, 581, 582, 632, 640
`\draw_path_moveto:n`
.... 243, 579, 584, 631, 639
`\draw_path_rectangle:nn` 557, 593
`\draw_path_rectangle_corners:nn` 586
`\draw_path_scope_begin:`
.... 749, 1289, 1322
`\draw_path_scope_end:` 749, 1293, 1330
`\draw_path_use:n` 649
`\draw_path_use_clear:n` 649
`\draw_point_interpolate_arcaxes:nnnnnn`
..... 988
`\draw_point_interpolate_curve:nnnnnn`
..... 1020
`\draw_point_interpolate_curve:nnnnnnn`
..... 1020
`\draw_point_interpolate_curve_-`
auxi:nnnnnnnnn 1020
`\draw_point_interpolate_curve_-`
auxii:nnnnnnnnn 1020
`\draw_point_interpolate_curve_-`
auxiii:nnnnnnn 1020
`\draw_point_interpolate_curve_-`
auxiv:nnnnnnn 1020
`\draw_point_interpolate_curve_-`
auxv:nnw 1020
`\draw_point_interpolate_curve_-`
auxvi:n 1020
`\draw_point_interpolate_curve_-`
auxvii:nnnnnnnnn 1020
`\draw_point_interpolate_curve_-`
auxviii:nnnnnnn 1020

<code>\draw_point_interpolate_distance:nnn</code>	<code>__draw_backend_clip:</code>	687
..... 970, 1559, 1568	<code>__draw_backend_closepath: ...</code>	1405
<code>\draw_point_interpolate_line:nnn</code>	<code>__draw_backend_curveto:nnnnnn</code>	1409
955	<code>__draw_backend_dash_pattern:nn</code>	1675
<code>\draw_point_intersect_circles:nnnnn</code>	<code>__draw_backend_discardpath: ..</code>	692
..... 899	<code>__draw_backend_end:</code>	1257
<code>\draw_point_intersect_lines:nnnn</code>	<code>__draw_backend_evenodd_rule: .</code>	1687
874	<code>__draw_backend_join_bevel: ..</code>	1689
<code>\draw_point_polar:nn</code>	<code>__draw_backend_join_miter: ..</code>	1690
850	<code>__draw_backend_join_round: ..</code>	1691
<code>\draw_point_polar:nnn</code>	<code>__draw_backend_lineto:nn</code>	1415
..... 431, 437, 441, 447, 850	<code>__draw_backend_linewidth:n ..</code>	1665
<code>\draw_point_transform:n</code>	<code>__draw_backend_miterlimit:n .</code>	1683
28, 31, 34, 37, 247, 259, 275, 276,	<code>__draw_backend_moveto:nn</code>	1417
277, 309, 310, 436, 440, 496, 567, 1157	<code>__draw_backend_nonzero_rule: .</code>	1688
<code>\draw_point_unit_vector:n ..</code>	<code>__draw_backend_rectangle:nnnn</code>	1422
857, 983	<code>__draw_backend_scope_begin: ...</code>	
<code>\draw_point_vec:nn</code> 135, 1286	
1114	<code>__draw_backend_scope_end: ..</code>	138, 1296
<code>\draw_point_vec:nnn</code>	<code>__draw_box_use:Nnnnn</code>	17, 68
1114	<code>__draw_color:nn</code>	1693
<code>\draw_point_vec_polar:nn</code>	<code>__draw_color:nnn</code>	1693
1144	<code>\l__draw_color_tmp_tl</code>	1692, 1700, 1701
<code>\draw_point_vec_polar:nnn</code>	<code>\l__draw_corner_arc_bool</code>	
1144 223, 231, 232, 236, 562	
<code>\draw_scope_begin:</code>	<code>\l__draw_corner_xarc_dim</code>	
1284 221, 226, 229, 239	
<code>\draw_scope_end:</code>	<code>\l__draw_corner_yarc_dim</code>	
1291 221, 227, 230, 240	
<code>\draw_suspend_begin:</code>	<code>__draw_draw_polar:nnn</code>	850
1319	<code>__draw_draw_vec_polar:nnn</code>	
<code>\draw_suspend_end:</code> 1147, 1148, 1156	
1319	<code>\l__draw_fill_color_tl</code>	1281
<code>\draw_transform_matrix:nnnn</code>	<code>\g__draw_id_int</code>	1223, 1234
1755,	<code>__draw_if_recursion_tail_stop_-</code>	
1844, 1846, 1848, 1854, 1856, 1867	do:Nn	12, 1456
<code>\draw_transform_matrix_absolute:nnnn</code>	<code>\l__draw_layer_close_bool</code>	
..... 1729, 1770, 1835 89, 99, 101, 118	
<code>\draw_transform_matrix_invert: ..</code>	<code>\l__draw_layer_main_box</code>	
1787 131, 132, 1221, 1250	
<code>\draw_transform_matrix_reset: ...</code>	<code>\l__draw_layer_tl</code>	87, 98, 102
..... 1240, 1323, 1715	<code>\g__draw_layers_clist</code>	90
<code>\draw_transform_rotate:n</code>	<code>__draw_layers_insert:</code>	125, 1256
1857	<code>__draw_layers_restore: ...</code>	142, 1329
<code>\draw_transform_scale:n</code>	<code>__draw_layers_save:</code>	142, 1325
1843	<code>\g__draw_linewidth_dim</code>	
<code>\draw_transform_shift:n</code>	... 735, 1288, 1294, 1659, 1664, 1665	
..... 1755, 1850, 1852	<code>\l__draw_linewidth_dim</code>	
<code>\draw_transform_shift_absolute:n</code> 1281, 1288, 1294	
..... 1729, 1840	<code>\l__draw_main_box</code>	1221, 1235,
<code>\draw_transform_shift_invert: .</code>	1266, 1270, 1272, 1274, 1275, 1278	
1787	<code>\l__draw_matrix_a_fp</code>	
<code>\draw_transform_shift_reset: ...</code>	. 45, 1169, 1201, 1709, 1717, 1731,	
..... 1241, 1324, 1715	1737, 1771, 1773, 1797, 1806, 1813	
<code>\draw_transform_triangle:nnn ...</code>		
..... 489, 1821		
<code>\draw_transform_xscale:n</code>		
1843		
<code>\draw_transform_xshift:n</code>		
1843		
<code>\draw_transform_xslant:n</code>		
1843		
<code>\draw_transform_yscale:n</code>		
1843		
<code>\draw_transform_yshift:n</code>		
1843		
<code>\draw_transform_yslant:n</code>		
1843		
<code>\draw_xvec:n</code>		
1096, 1111		
<code>\draw_yvec:n</code>		
1096, 1112		
<code>\draw_zvec:n</code>		
1096, 1113		
draw internal commands:		
<code>__draw_backend_begin:</code>		1236
<code>__draw_backend_box_use:Nnnnn ...</code>		44
<code>__draw_backend_cap_but:</code>		1684
<code>__draw_backend_cap_rectangle: ..</code>		1685
<code>__draw_backend_cap_round: ...</code>		1686

\l__draw_matrix_active_bool	__draw_path_mark_corner:
563, 1164, 1196, <u>1708</u> , 1742, 1743, 1789 <u>234</u> , 263, 272, 289, 300, 318, 389
\l__draw_matrix_b_fp	__draw_path_moveto:nn
. 46, 1175, 1206, <u>1709</u> , 1718, 1732, <u>243</u> , 293, 504, 512
1738, 1772, 1774, 1798, 1808, 1809	__draw_path_rectangle:nnnn . . . <u>557</u>
\l__draw_matrix_c_fp	__draw_path_rectangle_corners:nnnn
. 47, 1170, 1202, <u>1709</u> , 1719, 1733, <u>586</u>
1739, 1771, 1773, 1798, 1810, 1811	__draw_path_rectangle_corners:nnnnn
\l__draw_matrix_d_fp 589, 592
. 48, 1176, 1207, 1712, 1720, 1734,	__draw_path_rectangle_rounded:nnnn
1740, 1772, 1774, 1797, 1807, 1812 <u>557</u>
__draw_path_arc:nnnn <u>348</u>	__draw_path_reset_limits:
__draw_path_arc:nnNnn <u>348</u> <u>187</u> , 661, 760, 1238
\c__draw_path_arc_60_fp <u>348</u>	\l__draw_path_tmp_tl
\c__draw_path_arc_90_fp <u>348</u> <u>178</u> , 426, 449, 468, 472, 476, 480
__draw_path_arc_add:nnnn <u>348</u>	\l__draw_path_tmpa_fp
__draw_path_arc_aux_add:nn <u>178</u> , 314, 324, 336
. 455, 461, 473, 478	\l__draw_path_tmpb_fp
__draw_path_arc_auxi:nnnnNnn <u>178</u> , 315, 331, 340
. <u>348</u> , 375, 382	__draw_path_update_last:nn
__draw_path_arc_auxii:nnnNnnnn <u>348</u> <u>216</u> , 253, 266, 285, 575
__draw_path_arc_auxiii:nn <u>348</u>	__draw_path_update_limits:nn . . .
__draw_path_arc_auxiv:nnnn <u>348</u> 27, 30, 33, 36,
__draw_path_arc_auxv:nn <u>348</u>	<u>187</u> , 251, 264, 281, 282, 283, 572, 573
__draw_path_arc_auxvi:nn <u>348</u>	__draw_path_use:n <u>649</u>
\l__draw_path_arc_delta_fp <u>348</u>	__draw_path_use_action_draw: . . <u>649</u>
\l__draw_path_arc_start_fp <u>348</u>	__draw_path_use_action_fillstroke:
__draw_path_curveto:nnnn <u>305</u> <u>649</u>
__draw_path_curveto:nnnnnn	\l__draw_path_use_bb_bool <u>647</u>
<u>243</u> , 301, 319, 449, 516, 526, 536, 546	\l__draw_path_use_clear_bool <u>647</u> , 706
\c__draw_path_curveto_a_fp <u>305</u>	\l__draw_path_use_clip_bool
\c__draw_path_curveto_b_fp <u>305</u> <u>644</u> , 667, 685
__draw_path_ellipse:nnnnnn . . . <u>492</u>	\l__draw_path_use_fill_bool
__draw_path_ellipse_arci:nnnnnn <u>492</u> <u>644</u> , 668, 690, 695, 701, 715
__draw_path_ellipse_arci:nnnnnn <u>492</u>	__draw_path_use_stroke_bb: . . . <u>649</u>
. <u>492</u>	__draw_path_use_stroke_bb_-
__draw_path_ellipse_arci:nnnnnn	aux:NnN <u>649</u>
. <u>492</u>	\l__draw_path_use_stroke_bool . . .
__draw_path_ellipse_arci:nnnnnn	<u>644</u> , 669, 682, 691, 696, 702, 711, 716
. <u>492</u>	\g__draw_path_xmax_dim
\c__draw_path_ellipse_fp <u>492</u> <u>183</u> , 189, 190, 211, 754, 776
__draw_path_grid_auxi:nnnnnn . . <u>594</u>	\l__draw_path_xmax_dim . <u>740</u> , 754, 776
__draw_path_grid_auxii:nnnnnn . <u>594</u>	\g__draw_path_xmin_dim
__draw_path_grid_auxiii:nnnnnn <u>594</u> <u>183</u> , 191, 192, 212, 755, 777
__draw_path_grid_auxiiii:nnnnnn <u>594</u>	\l__draw_path_xmin_dim . <u>740</u> , 755, 777
__draw_path_grid_auxiv:nnnnnnnn <u>594</u>	\g__draw_path_ymax_dim
\g__draw_path_lastx_dim <u>183</u> , 193, 194, 213, 756, 778
. . . . <u>181</u> , 218, 323, 456, 462, 752, 780	\l__draw_path_ymax_dim . <u>740</u> , 756, 778
\l__draw_path_lastx_dim <u>740</u> , 752, 780	\g__draw_path_ymin_dim
\g__draw_path_lasty_dim <u>183</u> , 195, 196, 214, 757, 779
. . . . <u>181</u> , 219, 330, 457, 463, 753, 781	\l__draw_path_ymin_dim . <u>740</u> , 757, 779
\l__draw_path_lasty_dim <u>740</u> , 753, 781	__draw_point_interpolate_-
__draw_path_lineto:nn <u>243</u> , 295	arcaxes_auxi:nnnnnnnnnn <u>988</u>

```

\__draw_point_interpolate_-
  arcaxes_auxii:nnnnnnnnn . . . . 988
\__draw_point_interpolate_-
  arcaxes_auxiii:nnnnnnnn . . . . 988
\__draw_point_interpolate_-
  arcaxes_auxiv:nnnnnnnn . . . . 988
\__draw_point_interpolate_curve_-
  auxi:nnnnnnnn . . . . . 1023, 1026
\__draw_point_interpolate_curve_-
  auxii:nnnnnnnn . . . . . 1028, 1032, 1040
\__draw_point_interpolate_curve_-
  auxiii:nnnnnn . . . . . 1035, 1046, 1054
\__draw_point_interpolate_curve_-
  auxiv:nnnnnn 1048, 1049, 1050, 1055
\__draw_point_interpolate_curve_-
  auxv:nw . . . . . 1057, 1061, 1068
\__draw_point_interpolate_curve_-
  auxvi:n . . . . . 1052, 1073
\__draw_point_interpolate_curve_-
  auxvii:nnnnnnnn . . . . . 1074, 1075
\__draw_point_interpolate_curve_-
  auxviii:nnnnnn . . . . . 1077, 1084, 1089
\__draw_point_interpolate_-
  distance:nnnn . . . . . 973, 976
\__draw_point_interpolate_-
  distance:nnnnn . . . . . 970, 980
\__draw_point_interpolate_-
  distance:nnnnnn . . . . . 970
\__draw_point_interpolate_line_-
  aux:nnnnn . . . . . 955
\__draw_point_interpolate_line_-
  aux:nnnnnn . . . . . 955
\__draw_point_intersect_circles_-
  auxi:nnnnnnnn . . . . . 899
\__draw_point_intersect_circles_-
  auxii:nnnnnnnn . . . . . 899
\__draw_point_intersect_circles_-
  auxiii:nnnnnnnn . . . . . 899
\__draw_point_intersect_circles_-
  auxiv:nnnnnnnn . . . . . 899
\__draw_point_intersect_circles_-
  auxv:nnnnnnnnnn . . . . . 899
\__draw_point_intersect_circles_-
  auxvi:nnnnnnnn . . . . . 899
\__draw_point_intersect_circles_-
  auxvii:nnnnnnnn . . . . . 899
\__draw_point_intersect_lines:nnnnnn
  . . . . . 874
\__draw_point_intersect_lines:nnnnnnnn
  . . . . . 874
\__draw_point_intersect_lines_-
  aux:nnnnnn . . . . . 874
\__draw_point_process:nn . . . . .
  . . . . . 26, 29, 32, 35, 245, 257,
  293, 295, 427, 443, 793, 858, 972,
  978, 1104, 1159, 1191, 1747, 1779, 1825
\__draw_point_process:nnn . . . . . 307,
  433, 559, 588, 596, 793, 901, 957, 1823
\__draw_point_process:nnnn . . . . .
  . . . . . 270, 298, 494, 793, 990
\__draw_point_process:nnnnn . . . . .
  . . . . . 793, 876, 1022
\__draw_point_process_auxi:nn . . . . . 793
\__draw_point_process_auxii:nw . . . . . 793
\__draw_point_process_auxiii:nnn . . . . . 793
\__draw_point_process_auxiv:nw . . . . . 793
\__draw_point_process_auxv:nnnn . . . . . 793
\__draw_point_process_auxvi:nw . . . . . 793
\__draw_point_process_auxvii:nnnnn
  . . . . . 793
\__draw_point_process_auxviii:nw . . . . . 793
\__draw_point_to_dim:n . . . . . 796,
  806, 807, 817, 818, 819, 830, 831,
  832, 833, 844, 1016, 1086, 1118,
  1132, 1150, 1166, 1182, 1198, 1211
\__draw_point_to_dim_aux:n . . . . . 844
\__draw_point_to_dim_aux:w . . . . . 844
\__draw_point_transform:nn . . . . . 1157
\__draw_point_transform_noshift:n
  . . . . . 430, 446, 497, 498, 1189
\__draw_point_transform_noshift:nn
  . . . . . 1189
\__draw_point_unit_vector:nn . . . . . 857
\__draw_point_unit_vector:nnn . . . . . 857
\__draw_point_vec:nn . . . . . 1114
\__draw_point_vec:nnn . . . . . 1114
\__draw_point_vec_polar:nnn . . . . . 1144
\__draw_reset_bb: . . . . . 1224, 1237, 1309
\__draw_scope_bb_begin: . . . . . 1302, 1321
\__draw_scope_bb_end: . . . . . 1302, 1331
\__draw_softpath_add:n . . . . .
  . . . . . 773, 1339, 1358,
  1367, 1376, 1381, 1391, 1399, 1654
\c__draw_softpath_arc_fp . . . . .
  . . . . . 1433, 1596, 1600, 1605, 1609
\__draw_softpath_clear: . . . . .
  . . . . . 660, 707, 765, 769, 1242, 1342
\__draw_softpath_close_op:nn . . . . .
  . . . . . 1360, 1404, 1495, 1531, 1633
\__draw_softpath_closepath: . . . . .
  . . . . . 290, 511, 1356
\l__draw_softpath_corneri_dim . . . . .
  . . . . . 1427, 1480, 1483, 1569
\l__draw_softpath_cornerii_dim . . . . .
  . . . . . 1427, 1481, 1484, 1560
\g__draw_softpath_corners_bool . . . . .
  . . . . . 764, 771, 1338, 1350, 1401, 1436, 1452

```


\l__draw_softpath_corners_bool	740, 763, 772	__draw_softpath_rectangle_-	
\l__draw_softpath_curve_end_tl	1426, 1557, 1576, 1625, 1636	opi:nnNnn	1404
__draw_softpath_curveto:nnnnnn	284, 1356	__draw_softpath_rectangle_-	
__draw_softpath_curveto_opi:nn	1369, 1404, 1492, 1530, 1593	opii:nn	1394, 1404
__draw_softpath_curveto_-		__draw_softpath_round_action:nn	
opi:nnNnnNnn	1404	1434
__draw_softpath_curveto_opii:nn		__draw_softpath_round_action:Nnn	
.	1370, 1404, 1602	1434
__draw_softpath_curveto_-		__draw_softpath_round_action_-	
opiii:nn	1371, 1404, 1611	close:	1434
\l__draw_softpath_first_tl	1427, 1443, 1460,	__draw_softpath_round_action_-	
1461, 1471, 1490, 1491, 1517, 1521		curveto:NnnNnn	1434
\l__draw_softpath_internal_tl	1337, 1344, 1345, 1445, 1447	__draw_softpath_round_calc:NnnNnn	
\g__draw_softpath_lastx_dim	758, 774, 1352, 1361, 1385	1434
\l__draw_softpath_lastx_dim	746, 758, 774	__draw_softpath_round_calc:nnnnw	
\l__draw_softpath_lastx_fp	1427, 1441, 1462, 1510, 1572, 1579	1434
\g__draw_softpath_lasty_dim	759, 775, 1352, 1362, 1386	__draw_softpath_round_close:nn	1434
\l__draw_softpath_lasty_dim	747, 759, 775	__draw_softpath_round_close:w	1434
\l__draw_softpath_lasty_fp	1427, 1442, 1463, 1511, 1573, 1580	__draw_softpath_round_corners: .	
__draw_softpath_lineto:nn	265, 1356	679, 1434
__draw_softpath_lineto_op:nn	1377, 1404, 1498, 1529, 1642	__draw_softpath_round_end: . .	1434
\g__draw_softpath_main_tl	761, 1336, 1340, 1344, 1349, 1445, 1653	__draw_softpath_round_lookahead:NnnNnn	
\l__draw_softpath_main_tl	19, 761, 773, 1424,	1434
1439, 1466, 1468, 1649, 1651, 1654		__draw_softpath_round_loop:Nnn	1434
\g__draw_softpath_move_bool	1354, 1383	__draw_softpath_round_roundpoint:NnnNnnNnn	
\l__draw_softpath_move_tl	1427, 1444,	1434
1467, 1470, 1518, 1620, 1643, 1650		__draw_softpath_roundpoint:nn . .	
__draw_softpath_moveto:nn	252, 1356	238, 1356
__draw_softpath_moveto_op:nn	1382, 1404, 1464, 1622	__draw_softpath_roundpoint_-	
\l__draw_softpath_part_tl	1425, 1440,	op:nn	1400, 1404, 1457, 1539
1469, 1472, 1474, 1508, 1563, 1652		__draw_softpath_use:	684, 1342
__draw_softpath_rectangle:nnnn	574, 1356	\l__draw_stroke_color_tl	1281
__draw_softpath_rectangle_-		\l__draw_tmp_box	16, 40, 51,
opi:nn	1393, 1404	55, 57, 58, 59, 60, 66, 68, 69, 70, 71, 72	
		\l__draw_tmp_seq	1667
		__draw_tranform_triangle:nnnnnn	
		1826, 1831
		__draw_transform:nnnn	1755
		__draw_transform_invert:n	1787
		__draw_transform_rotate:n	1857
		__draw_transform_rotate:nn	1857
		__draw_transform_shift:nn	1755
		__draw_transform_shift_absolute:nn	
		1729
		__draw_vec:nn	1096
		__draw_vec:nnn	1096
		\g__draw_xmax_dim	199,
		200, 1216, 1226, 1261, 1276, 1305, 1313	
		\l__draw_xmax_dim	1298, 1305, 1313
		\g__draw_xmin_dim	
		201, 202, 1216, 1227,
		1259, 1262, 1268, 1276, 1306, 1314	

<code>\l__draw_xmin_dim</code> . . .	1298 , 1306 , 1314	1579 , 1580 , 1717 , 1720 , 1731 , 1732 , 1733 , 1734 , 1806 , 1808 , 1810 , 1812	
<code>\l__draw_xshift_dim</code>	53 , 1171 , 1185 , 1709 , 1724 , 1752 , 1784 , 1818	<code>\fp_to_decimal:N</code>	376 , 383 , 391
<code>\l__draw_xvec_x_dim</code>	1090 , 1120 , 1134 , 1152	<code>\fp_to_dim:n</code>	321 , 328 , 335 , 339 , 357 , 358 , 404 , 413 , 481 , 505 , 517 , 518 , 519 , 520 , 521 , 522 , 527 , 528 , 529 , 530 , 531 , 532 , 537 , 538 , 539 , 540 , 541 , 542 , 547 , 548 , 549 , 550 , 551 , 552 , 620 , 621 , 1595 , 1599 , 1604 , 1608 , 1664 , 1672 , 1677
<code>\l__draw_xvec_y_dim</code> .	1090 , 1121 , 1138	<code>\fp_use:N</code>	45 , 46 , 47 , 48 , 554
<code>\g__draw_ymax_dim</code>	203 , 204 , 1216 , 1228 , 1263 , 1273 , 1307 , 1315	<code>\fp_while_do:nNnn</code>	371
<code>\l__draw_ymax_dim</code> . . .	1298 , 1307 , 1315	<code>\fp_zero:N</code>	1441 , 1442 , 1718 , 1719
<code>\g__draw_ymin_dim</code> .	205 , 206 , 1216 , 1229 , 1264 , 1269 , 1273 , 1308 , 1316	<code>\c_one_fp</code>	1737 , 1740
<code>\l__draw_ymin_dim</code> . . .	1298 , 1308 , 1316	<code>\c_zero_fp</code>	867 , 1738 , 1739
<code>\l__draw_yshift_dim</code>	54 , 1177 , 1185 , 1709 , 1725 , 1753 , 1785 , 1819	G	
<code>\l__draw_yvec_x_dim</code> .	1090 , 1120 , 1135	group commands:	
<code>\l__draw_yvec_y_dim</code>	1090 , 1121 , 1139 , 1153	<code>\group_begin:</code>	39 , 65 , 95 , 106 , 751 , 1233 , 1287 , 1304 , 1438 , 1669
<code>\l__draw_zvec_x_dim</code>	1090 , 1136	<code>\group_end:</code>	61 , 73 , 120 , 123 , 782 , 1279 , 1295 , 1317 , 1450 , 1679
<code>\l__draw_zvec_y_dim</code>	1090 , 1140	H	
E		hbox commands:	
<code>\end</code>	172 , 787	<code>\hbox_gset:Nw</code>	104
exp commands:		<code>\hbox_gset_end:</code>	121
<code>\exp_after:wN</code>	449 , 467 , 1446 , 1520 , 1623 , 1634 , 1643 , 1701	<code>\hbox_set:Nn</code>	40 , 51 , 66 , 1266
<code>\exp_args:Nf</code>	795 , 861	<code>\hbox_set:Nw</code>	1235 , 1250
<code>\exp_args:Nff</code>	805	<code>\hbox_set_end:</code>	1254 , 1258
<code>\exp_args:Nfff</code>	816	I	
<code>\exp_args:Nffff</code>	829	int commands:	
<code>\exp_args:NNNV</code>	1254	<code>\int_gincr:N</code>	1234
<code>\exp_args:Nx</code>	1683	<code>\int_if_odd:nTF</code>	944
<code>\exp_not:N</code>	1565 , 1593 , 1602 , 1611 , 1620 , 1623 , 1624 , 1625 , 1630 , 1634 , 1635 , 1636	<code>\int_new:N</code>	1223
F		K	
fp commands:		kernel internal commands:	
<code>\fp_compare:nNnTF</code>	363 , 373 , 867	<code>__kernel_quark_new_test:N</code>	12
<code>\fp_compare_p:nNn</code>	1737 , 1738 , 1739 , 1740	M	
<code>\fp_const:Nn</code>	346 , 347 , 485 , 486 , 554 , 1433	mode commands:	
<code>\fp_eval:n</code> .	355 , 356 , 377 , 384 , 393 , 845 , 853 , 862 , 883 , 884 , 885 , 886 , 887 , 888 , 908 , 913 , 914 , 921 , 928 , 929 , 936 , 943 , 945 , 958 , 963 , 981 , 997 , 1002 , 1009 , 1010 , 1029 , 1036 , 1058 , 1059 , 1078 , 1079 , 1080 , 1081 , 1115 , 1128 , 1147 , 1683 , 1760 , 1761 , 1762 , 1763 , 1793 , 1858 , 1862 , 1863	<code>\mode_leave_vertical:</code>	1277
<code>\fp_new:N</code>	179 , 180 , 483 , 484 , 1427 , 1428 , 1709 , 1710 , 1711 , 1712	msg commands:	
<code>\fp_set:Nn</code>	314 , 315 , 369 , 370 , 450 , 451 , 1462 , 1463 , 1510 , 1511 , 1579 , 1580 , 1717 , 1720 , 1731 , 1732 , 1733 , 1734 , 1806 , 1808 , 1810 , 1812	<code>\msg_error:nnn</code>	81 , 112 , 113 , 676
		<code>\msg_new:nnn</code>	167
		<code>\msg_new:nnnn</code>	164 , 169 , 784
P		P	
<code>\pgfextractx</code>	21	<code>\pgfextractx</code>	21
<code>\pgfextracty</code>	21	<code>\pgfgetlastxy</code>	21
<code>\pgfgetlastxy</code>	21	<code>\pgfgettransform</code>	47
<code>\pgfgettransform</code>	47	<code>\pgfgettransformentries</code>	47

<code>\pgfinnerlinewidth</code>	45		S
<code>\pgflowlevel</code>	47	scan commands:	
<code>\pgflowlevelsynccm</code>	47	<code>\scan_new:N</code>	8, 9
<code>\pgfpatharcto</code>	6	seq commands:	
<code>\pgfpatharctoprecomputed</code>	6	<code>\seq_new:N</code>	1681
<code>\pgfpathcosine</code>	6	<code>\seq_set_from_clist:Nn</code>	1670
<code>\pgfpathcurvebetweentime</code>	6	<code>\seq_set_map:NNn</code>	1671
<code>\pgfpathcurvebetweentimecontinue</code>	6	<code>\seq_use:Nn</code>	1676
<code>\pgfpathparabola</code>	6	skip commands:	
<code>\pgfpathsine</code>	6	<code>\skip_horizontal:n</code>	1268
<code>\pgfpointadd</code>	21	str commands:	
<code>\pgfpointborderellipse</code>	21	<code>\str_if_eq:nnTF</code>	
<code>\pgfpointborderrectangle</code>	21	80, 98, 111, 129, 146, 157
<code>\pgfpointcylindrical</code>	21	<code>\str_if_eq_p:nn</code>	658
<code>\pgfpointdiff</code>	21		T
<code>\pgfpointorigin</code>	21	tex commands:	
<code>\pgfpointscale</code>	21	<code>\tex_kern:D</code>	53
<code>\pgfpointspherical</code>	21	tl commands:	
<code>\pgfqpoint</code>	21	<code>\tl_build_gclear:N</code>	1349, 1653
<code>\pgfqpointpolar</code>	21	<code>\tl_build_get:NN</code>	761, 1344, 1445
<code>\pgfqpointscale</code>	21	<code>\tl_build_gput_right:Nn</code>	1340
<code>\pgfqpointxy</code>	21	<code>\tl_clear:N</code>	
<code>\pgfqpointxyz</code>	21	426, 1439, 1440, 1443, 1444, 1471, 1472
<code>\pgfsetinnerlinewidth</code>	45	<code>\tl_if_blank:nTF</code>	651
<code>\pgfsetinnerstrokecolor</code>	45	<code>\tl_if_blank_p:n</code>	657
<code>\pgftransformarcaxesattime</code>	47	<code>\tl_if_empty:NnTF</code>	1460, 1490
<code>\pgftransformarrow</code>	47	<code>\tl_if_empty_p:N</code>	1517, 1518
<code>\pgftransformcurveattime</code>	47	<code>\tl_new:N</code>	
<code>\pgftransformlineattime</code>	47	87, 178, 1282, 1283, 1336, 1337,
			1424, 1425, 1426, 1431, 1432, 1692
prg commands:		<code>\tl_put_right:Nn</code>	476, 480, 1466,
<code>\prg_do_nothing:</code>	1051, 1062, 1065	1468, 1474, 1508, 1563, 1649, 1651
<code>\ProvidesExplPackage</code>	4	<code>\tl_set:Nn</code>	88,
		102, 472, 1461, 1470, 1491, 1557, 1620
		token commands:	
Q		<code>\token_if_eq_meaning:NNTF</code>	
quark commands:		1457, 1464, 1492, 1495, 1498, 1539
<code>\quark_new:N</code>	10, 11	<code>\token_if_eq_meaning_p:NN</code>	
quark internal commands:		1529, 1530, 1531
<code>\s__draw_mark</code>	8,		U
.	811, 812, 823, 825, 839, 842, 1586, 1590	use commands:	
<code>\q__draw_recursion_stop</code>	10, 1449	<code>\use:N</code>	698, 731, 1704
<code>\q__draw_recursion_tail</code>	10, 1448	<code>\use:n</code>	42, 316, 352, 399,
<code>\s__draw_stop</code>	502, 1616, 1628, 1673, 1757, 1768, 1833
.	8, 800, 801, 811, 812, 823, 825,	<code>\use_i:nn</code>	21
.	839, 842, 1586, 1590, 1626, 1637, 1646	<code>\use_i:nnnn</code>	1630
		<code>\use_ii:nn</code>	21
		<code>\use_none:n</code>	1643
R			
<code>\RequirePackage</code>	7		