

File I

Implementation

1 l3draw implementation

```

1 <*package>
2 <@@=draw>
3 \ProvidesExplPackage{l3draw}{2024-01-04}{ }
4 {L3 Experimental core drawing support}

```

1.1 Internal auxiliaries

`\s__draw_mark` Internal scan marks.

```

\s__draw_stop      5 \scan_new:N \s__draw_mark
                   6 \scan_new:N \s__draw_stop

```

(End of definition for \s__draw_mark and \s__draw_stop.)

`\q__draw_recursion_tail` Internal recursion quarks.

```

\q__draw_recursion_stop 7 \quark_new:N \q__draw_recursion_tail
                        8 \quark_new:N \q__draw_recursion_stop

```

(End of definition for \q__draw_recursion_tail and \q__draw_recursion_stop.)

`__draw_if_recursion_tail_stop_do:Nn` Functions to query recursion quarks.

```

9 \__kernel_quark_new_test:N \__draw_if_recursion_tail_stop_do:Nn

```

(End of definition for __draw_if_recursion_tail_stop_do:Nn.)

Everything else is in the sub-files!

```

10 </package>

```

2 l3draw-boxes implementation

```

11 <*package>

```

```

12 <@@=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`

```

13 \box_new:N \l__draw_tmp_box

```

(End of definition for \l__draw_tmp_box.)

`\draw_box_use:N`

`\draw_box_use:Nn`

`__draw_box_use:nNnnnnn`

`__draw_box_use:Nnnnn`

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.

```

14 \cs_new_protected:Npn \draw_box_use:N #1

```

```

15 {

```

```

16     \__draw_box_use:Nnnnnnn #1
17     { Opt } { -\box_dp:N #1 } { \box_wd:N #1 } { \box_ht:N #1 }
18 }
19 \cs_new_protected:Npn \draw_box_use:Nn #1#2
20 {
21     \__draw_box_use:nNnnnn {#2} #1
22     { Opt } { -\box_dp:N #1 } { \box_wd:N #1 } { \box_ht:N #1 }
23 }
24 \cs_new_protected:Npn \__draw_box_use:nNnnnn #1#2#3#4#5#6
25 {
26     \draw_scope_begin:
27     \draw_transform_shift:n {#1}
28     \__draw_box_use:Nnnnnnn #2 {#3} {#4} {#5} {#6}
29     \draw_scope_end:
30 }
31 \cs_new_protected:Npn \__draw_box_use:Nnnnnnn #1#2#3#4#5
32 {
33     \bool_if:NT \l_draw_bb_update_bool
34     {
35         \__draw_point_process:nn
36         { \__draw_path_update_limits:nn }
37         { \draw_point_transform:n { #2 , #3 } }
38         \__draw_point_process:nn
39         { \__draw_path_update_limits:nn }
40         { \draw_point_transform:n { #4 , #3 } }
41         \__draw_point_process:nn
42         { \__draw_path_update_limits:nn }
43         { \draw_point_transform:n { #4 , #5 } }
44         \__draw_point_process:nn
45         { \__draw_path_update_limits:nn }
46         { \draw_point_transform:n { #2 , #5 } }
47     }
48     \group_begin:
49     \hbox_set:Nn \l__draw_tmp_box
50     {
51         \use:e
52         {
53             \__draw_backend_box_use:Nnnnn #1
54             { \fp_use:N \l__draw_matrix_a_fp }
55             { \fp_use:N \l__draw_matrix_b_fp }
56             { \fp_use:N \l__draw_matrix_c_fp }
57             { \fp_use:N \l__draw_matrix_d_fp }
58         }
59     }
60     \hbox_set:Nn \l__draw_tmp_box
61     {
62         \__kernel_kern:n { \l__draw_xshift_dim }
63         \box_move_up:nn { \l__draw_yshift_dim }
64         { \box_use_drop:N \l__draw_tmp_box }
65     }
66     \box_set_ht:Nn \l__draw_tmp_box { Opt }
67     \box_set_dp:Nn \l__draw_tmp_box { Opt }
68     \box_set_wd:Nn \l__draw_tmp_box { Opt }
69     \box_use_drop:N \l__draw_tmp_box

```

```

70     \group_end:
71 }

```

(End of definition for `\draw_box_use:N` and others. These functions are 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.

`\draw_coffin_use:Nnnn`

`_draw_coffin_use:nNnn`

```

72 \cs_new_protected:Npn \draw_coffin_use:Nnn #1#2#3
73 {
74     \_draw_coffin_use:nNnn { \_draw_box_use:Nnnnnnn }
75     #1 {#2} {#3}
76 }
77 \cs_new_protected:Npn \draw_coffin_use:Nnnn #1#2#3#4
78 {
79     \_draw_coffin_use:nNnn { \_draw_box_use:nNnnnn {#4} }
80     #1 {#2} {#3}
81 }
82 \cs_new_protected:Npn \_draw_coffin_use:nNnn #1#2#3#4
83 {
84     \group_begin:
85     \hbox_set:Nn \l__draw_tmp_box
86     { \coffin_typeset:Nnnnn #2 {#3} {#4} { Opt } { Opt } }
87     #1 \l__draw_tmp_box
88     { \box_wd:N \l__draw_tmp_box - \coffin_wd:N #2 }
89     { -\box_dp:N \l__draw_tmp_box }
90     { \box_wd:N \l__draw_tmp_box }
91     { \box_ht:N \l__draw_tmp_box }
92     \group_end:
93 }

```

(End of definition for `\draw_coffin_use:Nnn`, `\draw_coffin_use:Nnnn`, and `_draw_coffin_use:nNnn`. These functions are documented on page ??.)

```

94 \</package>

```

3 l3draw-layers implementation

```

95 \*package>
96 \@@=draw>

```

3.1 User interface

`\draw_layer_new:n`

```

97 \cs_new_protected:Npn \draw_layer_new:n #1
98 {
99     \str_if_eq:nnTF {#1} { main }
100     { \msg_error:nnn { draw } { main-reserved } }
101     {
102         \box_new:c { g__draw_layer_ #1 _box }
103         \box_new:c { l__draw_layer_ #1 _box }
104     }
105 }

```

(End of 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`.

```

106 \tl_new:N \l__draw_layer_tl
107 \tl_set:Nn \l__draw_layer_tl { main }

```

(End of definition for `\l__draw_layer_tl`.)

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

```

108 \bool_new:N \l__draw_layer_close_bool

```

(End of 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 109 \clist_new:N \l_draw_layers_clist
110 \clist_set:Nn \l_draw_layers_clist { main }
111 \clist_new:N \g__draw_layers_clist

```

(End of 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.

```

112 \cs_new_protected:Npn \draw_layer_begin:n #1
113 {
114   \group_begin:
115   \box_if_exist:cTF { g__draw_layer_ #1 _box }
116   {
117     \str_if_eq:VnTF \l__draw_layer_tl {#1}
118     { \bool_set_false:N \l__draw_layer_close_bool }
119     {
120       \bool_set_true:N \l__draw_layer_close_bool
121       \tl_set:Nn \l__draw_layer_tl {#1}
122       \box_gset:cn { g__draw_layer_ #1 _box } { Opt }
123       \hbox_gset:cw { g__draw_layer_ #1 _box }
124       \box_use_drop:c { g__draw_layer_ #1 _box }
125       \group_begin:
126     }
127     \draw_linewidth:n { \l_draw_default_linewidth_dim }
128   }
129   {
130     \str_if_eq:nnTF {#1} { main }
131     { \msg_error:nnn { draw } { unknown-layer } {#1} }
132     { \msg_error:nnn { draw } { main-layer } }
133   }
134 }
135 \cs_new_protected:Npn \draw_layer_end:
136 {
137   \bool_if:NT \l__draw_layer_close_bool
138   {
139     \group_end:
140     \hbox_gset_end:
141   }
142   \group_end:
143 }

```

(End of 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.

```

144 \cs_new_protected:Npn \__draw_layers_insert:
145 {
146   \clist_map_inline:Nn \l_draw_layers_clist
147   {
148     \str_if_eq:nnTF {##1} { main }
149     {
150       \box_set_wd:Nn \l__draw_layer_main_box { Opt }
151       \box_use_drop:N \l__draw_layer_main_box
152     }
153     {
154       \__draw_backend_scope_begin:
155       \box_gset_wd:cn { g__draw_layer_ ##1 _box } { Opt }
156       \box_use_drop:c { g__draw_layer_ ##1 _box }
157       \__draw_backend_scope_end:
158     }
159   }
160 }
```

(End of definition for __draw_layers_insert:.)

`__draw_layers_save:` Simple save/restore functions.
`__draw_layers_restore:`

```

161 \cs_new_protected:Npn \__draw_layers_save:
162 {
163   \clist_map_inline:Nn \l_draw_layers_clist
164   {
165     \str_if_eq:nnF {##1} { main }
166     {
167       \box_set_eq:cc { l__draw_layer_ ##1 _box }
168       { g__draw_layer_ ##1 _box }
169     }
170   }
171 }
172 \cs_new_protected:Npn \__draw_layers_restore:
173 {
174   \clist_map_inline:Nn \l_draw_layers_clist
175   {
176     \str_if_eq:nnF {##1} { main }
177     {
178       \box_gset_eq:cc { g__draw_layer_ ##1 _box }
179       { l__draw_layer_ ##1 _box }
180     }
181   }
182 }
```

(End of definition for __draw_layers_save: and __draw_layers_restore:.)

```

183 \msg_new:nnnn { draw } { main-layer }
184 { Material~cannot~be~added~to~'main'~layer. }
185 { The~main~layer~may~only~be~accessed~at~the~top~level. }
186 \msg_new:nnn { draw } { main-reserved }
187 { The~'main'~layer~is~reserved. }
188 \msg_new:nnnn { draw } { unknown-layer }
```

```

189 { Layer~'#1'~has-not-been-created. }
190 { You-have-tryed-to-use-layer~'#1',~but-it-was-never-set-up. }
191 % \end{macrocode}
192 %
193 % \begin{macrocode}
194 </package>

```

4 l3draw-paths implementation

```

195 <*package>
196 <@@=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 197 \tl_new:N \l__draw_path_tmp_tl
\l__draw_path_tmpb_fp 198 \fp_new:N \l__draw_path_tmpa_fp
199 \fp_new:N \l__draw_path_tmpb_fp

```

(End of 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 200 \dim_new:N \g__draw_path_lastx_dim
201 \dim_new:N \g__draw_path_lasty_dim

```

(End of 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 202 \dim_new:N \g__draw_path_xmax_dim
\g__draw_path_ymax_dim 203 \dim_new:N \g__draw_path_xmin_dim
\g__draw_path_ymin_dim 204 \dim_new:N \g__draw_path_ymax_dim
205 \dim_new:N \g__draw_path_ymin_dim

```

(End of 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:`

```

206 \cs_new_protected:Npn \__draw_path_update_limits:nn #1#2
207 {
208   \dim_gset:Nn \g__draw_path_xmax_dim
209   { \dim_max:nn \g__draw_path_xmax_dim {#1} }
210   \dim_gset:Nn \g__draw_path_xmin_dim
211   { \dim_min:nn \g__draw_path_xmin_dim {#1} }
212   \dim_gset:Nn \g__draw_path_ymax_dim
213   { \dim_max:nn \g__draw_path_ymax_dim {#2} }
214   \dim_gset:Nn \g__draw_path_ymin_dim
215   { \dim_min:nn \g__draw_path_ymin_dim {#2} }
216   \bool_if:NT \l_draw_bb_update_bool
217   {
218     \dim_gset:Nn \g__draw_xmax_dim
219     { \dim_max:nn \g__draw_xmax_dim {#1} }
220     \dim_gset:Nn \g__draw_xmin_dim
221     { \dim_min:nn \g__draw_xmin_dim {#1} }
222     \dim_gset:Nn \g__draw_ymax_dim
223     { \dim_max:nn \g__draw_ymax_dim {#2} }
224     \dim_gset:Nn \g__draw_ymin_dim
225     { \dim_min:nn \g__draw_ymin_dim {#2} }
226   }
227 }
228 \cs_new_protected:Npn \__draw_path_reset_limits:
229 {
230   \dim_gset:Nn \g__draw_path_xmax_dim { -\c_max_dim }
231   \dim_gset:Nn \g__draw_path_xmin_dim { \c_max_dim }
232   \dim_gset:Nn \g__draw_path_ymax_dim { -\c_max_dim }
233   \dim_gset:Nn \g__draw_path_ymin_dim { \c_max_dim }
234 }

```

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

`__draw_path_update_last:nn` A simple auxiliary to avoid repetition.

```

235 \cs_new_protected:Npn \__draw_path_update_last:nn #1#2
236 {
237   \dim_gset:Nn \g__draw_path_lastx_dim {#1}
238   \dim_gset:Nn \g__draw_path_lasty_dim {#2}
239 }

```

(End of 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
240 \dim_new:N \l__draw_corner_xarc_dim
241 \dim_new:N \l__draw_corner_yarc_dim

```

(End of 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.

```
242 \bool_new:N \l__draw_corner_arc_bool
```

(End of definition for \l__draw_corner_arc_bool.)

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

```
243 \cs_new_protected:Npn \draw_path_corner_arc:nn #1#2
244 {
245   \dim_set:Nn \l__draw_corner_xarc_dim {#1}
246   \dim_set:Nn \l__draw_corner_yarc_dim {#2}
247   \bool_lazy_and:nnTF
248     { \dim_compare_p:nNn \l__draw_corner_xarc_dim = { Opt } }
249     { \dim_compare_p:nNn \l__draw_corner_yarc_dim = { Opt } }
250     { \bool_set_false:N \l__draw_corner_arc_bool }
251     { \bool_set_true:N \l__draw_corner_arc_bool }
252 }
```

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

__draw_path_mark_corner: Mark up corners for arc post-processing.

```
253 \cs_new_protected:Npn \__draw_path_mark_corner:
254 {
255   \bool_if:NT \l__draw_corner_arc_bool
256   {
257     \__draw_softpath_roundpoint:VV
258       \l__draw_corner_xarc_dim
259       \l__draw_corner_yarc_dim
260   }
261 }
```

(End of 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 262 \cs_new_protected:Npn \draw_path_moveto:n #1
\__draw_path_lineto:nn 263 {
\draw_path_curveto:nnn 264   \__draw_point_process:nn
\__draw_path_curveto:nnnnnn 265     { \__draw_path_moveto:nn }
266     { \draw_point_transform:n {#1} }
267   }
268 \cs_new_protected:Npn \__draw_path_moveto:nn #1#2
269 {
270   \__draw_path_update_limits:nn {#1} {#2}
271   \__draw_softpath_moveto:nn {#1} {#2}
272   \__draw_path_update_last:nn {#1} {#2}
273 }
274 \cs_new_protected:Npn \draw_path_lineto:n #1
275 {
276   \__draw_point_process:nn
277   { \__draw_path_lineto:nn }
```



```

278     { \draw_point_transform:n {#1} }
279   }
280 \cs_new_protected:Npn \__draw_path_lineto:nn #1#2
281 {
282   \__draw_path_mark_corner:
283   \__draw_path_update_limits:nn {#1} {#2}
284   \__draw_softpath_lineto:nn {#1} {#2}
285   \__draw_path_update_last:nn {#1} {#2}
286 }
287 \cs_new_protected:Npn \draw_path_curveto:nnn #1#2#3
288 {
289   \__draw_point_process:nnnn
290   {
291     \__draw_path_mark_corner:
292     \__draw_path_curveto:nnnnnn
293   }
294   { \draw_point_transform:n {#1} }
295   { \draw_point_transform:n {#2} }
296   { \draw_point_transform:n {#3} }
297 }
298 \cs_new_protected:Npn \__draw_path_curveto:nnnnnn #1#2#3#4#5#6
299 {
300   \__draw_path_update_limits:nn {#1} {#2}
301   \__draw_path_update_limits:nn {#3} {#4}
302   \__draw_path_update_limits:nn {#5} {#6}
303   \__draw_softpath_curveto:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
304   \__draw_path_update_last:nn {#5} {#6}
305 }

```

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

\draw_path_close: A simple wrapper.

```

306 \cs_new_protected:Npn \draw_path_close:
307 {
308   \__draw_path_mark_corner:
309   \__draw_softpath_closepath:
310 }

```

(End of 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
311 \cs_new_protected:Npn \draw_path_canvas_moveto:n #1
312 { \__draw_point_process:nn { \__draw_path_moveto:nn } {#1} }
313 \cs_new_protected:Npn \draw_path_canvas_lineto:n #1
314 { \__draw_point_process:nn { \__draw_path_lineto:nn } {#1} }
315 \cs_new_protected:Npn \draw_path_canvas_curveto:nnn #1#2#3
316 {
317   \__draw_point_process:nnnn
318   {
319     \__draw_path_mark_corner:
320     \__draw_path_curveto:nnnnnn
321   }

```

```

322     {#1} {#2} {#3}
323 }

```

(End of 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
  \__draw_path_curveto:nnnn
\c__draw_path_curveto_a_fp
\c__draw_path_curveto_b_fp

```

A quadratic curve with one control point (x_c, y_c) . The two required control points are then

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

and

$$x_2 = \frac{1}{3}x_e + \frac{2}{3}x_c \quad y_2 = \frac{1}{3}y_e + \frac{2}{3}y_c$$

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

```

324 \cs_new_protected:Npn \draw_path_curveto:nn #1#2
325 {
326   \__draw_point_process:nnn
327   { \__draw_path_curveto:nnnn }
328   { \draw_point_transform:n {#1} }
329   { \draw_point_transform:n {#2} }
330 }
331 \cs_new_protected:Npn \__draw_path_curveto:nnnn #1#2#3#4
332 {
333   \fp_set:Nn \l__draw_path_tmpa_fp { \c__draw_path_curveto_b_fp * #1 }
334   \fp_set:Nn \l__draw_path_tmpb_fp { \c__draw_path_curveto_b_fp * #2 }
335   \use:e
336   {
337     \__draw_path_mark_corner:
338     \__draw_path_curveto:nnnnnn
339     {
340       \fp_to_dim:n
341       {
342         \c__draw_path_curveto_a_fp * \g__draw_path_lastx_dim
343         + \l__draw_path_tmpa_fp
344       }
345     }
346     {
347       \fp_to_dim:n
348       {
349         \c__draw_path_curveto_a_fp * \g__draw_path_lasty_dim
350         + \l__draw_path_tmpb_fp
351       }
352     }
353     {
354       \fp_to_dim:n
355       { \c__draw_path_curveto_a_fp * #3 + \l__draw_path_tmpa_fp }
356     }
357     {
358       \fp_to_dim:n

```

```

359         { \c__draw_path_curveto_a_fp * #4 + \l__draw_path_tmpb_fp }
360     }
361     {#3}
362     {#4}
363 }
364 }
365 \fp_const:Nn \c__draw_path_curveto_a_fp { 1 / 3 }
366 \fp_const:Nn \c__draw_path_curveto_b_fp { 2 / 3 }

```

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

```

\draw_path_arc:nnn Drawing an arc means dividing the total curve required into sections: using Bézier curves
\draw_path_arc:nnnn we can cover at most 90° at once. To allow for later manipulations, we aim to have roughly
\__draw_path_arc:nnnn equal last segments to the line, with the split set at a final part of 115°.
\__draw_path_arc:nnNnn
\__draw_path_arc_auxi:nnnnNnn
\__draw_path_arc_auxi:enenNnn
\__draw_path_arc_auxi:eennNnn
\__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
\l__draw_path_arc_start_fp
\c__draw_path_arc_90_fp
\c__draw_path_arc_60_fp
367 \cs_new_protected:Npn \draw_path_arc:nnn #1#2#3
368 { \draw_path_arc:nnnn {#1} {#2} {#3} {#3} }
369 \cs_new_protected:Npn \draw_path_arc:nnnn #1#2#3#4
370 {
371     \use:e
372     {
373         \__draw_path_arc:nnnn
374         { \fp_eval:n {#1} }
375         { \fp_eval:n {#2} }
376         { \fp_to_dim:n {#3} }
377         { \fp_to_dim:n {#4} }
378     }
379 }
380 \cs_new_protected:Npn \__draw_path_arc:nnnn #1#2#3#4
381 {
382     \fp_compare:nNnTF {#1} > {#2}
383     { \__draw_path_arc:nnNnn {#1} {#2} - {#3} {#4} }
384     { \__draw_path_arc:nnNnn {#1} {#2} + {#3} {#4} }
385 }
386 \cs_new_protected:Npn \__draw_path_arc:nnNnn #1#2#3#4#5
387 {
388     \fp_set:Nn \l__draw_path_arc_start_fp {#1}
389     \fp_set:Nn \l__draw_path_arc_delta_fp { abs( #1 - #2 ) }
390     \fp_while_do:nNnn { \l__draw_path_arc_delta_fp } > { 90 }
391     {
392         \fp_compare:nNnTF \l__draw_path_arc_delta_fp > { 115 }
393         {
394             \__draw_path_arc_auxi:eennNnn
395             { \fp_to_decimal:N \l__draw_path_arc_start_fp }
396             { \fp_eval:n { \l__draw_path_arc_start_fp #3 90 } }
397             { 90 } {#2}
398             #3 {#4} {#5}
399         }
400         {
401             \__draw_path_arc_auxi:eennNnn
402             { \fp_to_decimal:N \l__draw_path_arc_start_fp }
403             { \fp_eval:n { \l__draw_path_arc_start_fp #3 60 } }
404             { 60 } {#2}
405             #3 {#4} {#5}
406         }

```

```

407     }
408     \__draw_path_mark_corner:
409     \__draw_path_arc_auxi:enenNnn
410     { \fp_to_decimal:N \l__draw_path_arc_start_fp }
411     {#2}
412     { \fp_eval:n { abs( \l__draw_path_arc_start_fp - #2 ) } }
413     {#2}
414     #3 {#4} {#5}
415 }

```

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: $\frac{4}{3}(\sqrt{2} - 1) = 0.55228475$. For other cases, we follow the calculation used by `pgf` but with the second common case of 60° pre-calculated for speed.

```

416 \cs_new_protected:Npn \__draw_path_arc_auxi:nnnnNnn #1#2#3#4#5#6#7
417 {
418   \use:e
419   {
420     \__draw_path_arc_auxii:nnnNnnnn
421     {#1} {#2} {#4} #5 {#6} {#7}
422     {
423       \fp_to_dim:n
424       {
425         \cs_if_exist_use:cF
426         { c__draw_path_arc_ #3 _fp }
427         { 4/3 * tand( 0.25 * #3 ) }
428         * #6
429       }
430     }
431     {
432       \fp_to_dim:n
433       {
434         \cs_if_exist_use:cF
435         { c__draw_path_arc_ #3 _fp }
436         { 4/3 * tand( 0.25 * #3 ) }
437         * #7
438       }
439     }
440   }
441 }
442 \cs_generate_variant:Nn \__draw_path_arc_auxi:nnnnNnn { ene , ee }

```

We can now calculate the required points. As everything here is non-expandable, that is best done by using `e`-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

```

443 \cs_new_protected:Npn \__draw_path_arc_auxii:nnnNnnnn #1#2#3#4#5#6#7#8
444 {
445   \tl_clear:N \l__draw_path_tmp_tl
446   \__draw_point_process:nn
447   { \__draw_path_arc_auxiii:nn }
448   {
449     \__draw_point_transform_noshift:n

```

```

450         { \draw_point_polar:nnn {#7} {#8} { #1 #4 90 } }
451     }
452 \__draw_point_process:nnn
453 { \__draw_path_arc_auxiv:nnnn }
454 {
455     \draw_point_transform:n
456     { \draw_point_polar:nnn {#5} {#6} {#1} }
457 }
458 {
459     \draw_point_transform:n
460     { \draw_point_polar:nnn {#5} {#6} {#2} }
461 }
462 \__draw_point_process:nn
463 { \__draw_path_arc_auxv:nn }
464 {
465     \__draw_point_transform_noshift:n
466     { \draw_point_polar:nnn {#7} {#8} { #2 #4 -90 } }
467 }
468 \exp_after:wN \__draw_path_curveto:nnnnnn \l__draw_path_tmp_tl
469 \fp_set:Nn \l__draw_path_arc_delta_fp { abs ( #2 - #3 ) }
470 \fp_set:Nn \l__draw_path_arc_start_fp {#2}
471 }

```

The first control point.

```

472 \cs_new_protected:Npn \__draw_path_arc_auxiii:nn #1#2
473 {
474     \__draw_path_arc_aux_add:nn
475     { \g__draw_path_lastx_dim + #1 }
476     { \g__draw_path_lasty_dim + #2 }
477 }

```

The end point: simple arithmetic.

```

478 \cs_new_protected:Npn \__draw_path_arc_auxiv:nnnn #1#2#3#4
479 {
480     \__draw_path_arc_aux_add:nn
481     { \g__draw_path_lastx_dim - #1 + #3 }
482     { \g__draw_path_lasty_dim - #2 + #4 }
483 }

```

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

```

484 \cs_new_protected:Npn \__draw_path_arc_auxv:nn #1#2
485 {
486     \exp_after:wN \__draw_path_arc_auxvi:nn
487     \l__draw_path_tmp_tl {#1} {#2}
488 }
489 \cs_new_protected:Npn \__draw_path_arc_auxvi:nn #1#2#3#4#5#6
490 {
491     \tl_set:Nn \l__draw_path_tmp_tl { {#1} {#2} }
492     \__draw_path_arc_aux_add:nn
493     { #5 + #3 }
494     { #6 + #4 }
495     \tl_put_right:Nn \l__draw_path_tmp_tl { {#3} {#4} }
496 }
497 \cs_new_protected:Npn \__draw_path_arc_aux_add:nn #1#2
498 {

```

```

499 \tl_put_right:Ne \l__draw_path_tmp_tl
500 { { \fp_to_dim:n {#1} } { \fp_to_dim:n {#2} } }
501 }
502 \fp_new:N \l__draw_path_arc_delta_fp
503 \fp_new:N \l__draw_path_arc_start_fp
504 \fp_const:cn { c__draw_path_arc_90_fp } { 4/3 * (sqrt(2) - 1) }
505 \fp_const:cn { c__draw_path_arc_60_fp } { 4/3 * tand(15) }

```

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

\draw_path_arc_axes:nnnn A simple wrapper.

```

506 \cs_new_protected:Npn \draw_path_arc_axes:nnnn #1#2#3#4
507 {
508   \group_begin:
509   \draw_transform_triangle:nnn { 0cm , 0cm } {#3} {#4}
510   \draw_path_arc:nnn {#1} {#2} { 1pt }
511   \group_end:
512 }

```

(End of 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 same constant. We need to deal with the ellipse in four parts and also deal with moving to the right place, closing it and ending up back at the center. That is handled on a per-arc basis, each in a separate auxiliary for readability.

```

\__draw_path_ellipse:nnnnnnn
  \__draw_path_ellipse_arci:nnnnnn
  \__draw_path_ellipse_arcii:nnnnnn
  \__draw_path_ellipse_arciiii:nnnnnn
  \__draw_path_ellipse_arciv:nnnnnn
\c__draw_path_ellipse_fp
513 \cs_new_protected:Npn \draw_path_ellipse:nnn #1#2#3
514 {
515   \__draw_point_process:nnnn
516   { \__draw_path_ellipse:nnnnnn }
517   { \draw_point_transform:n {#1} }
518   { \__draw_point_transform_noshift:n {#2} }
519   { \__draw_point_transform_noshift:n {#3} }
520 }
521 \cs_new_protected:Npn \__draw_path_ellipse:nnnnnn #1#2#3#4#5#6
522 {
523   \use:e
524   {
525     \__draw_path_moveto:nn
526     { \fp_to_dim:n { #1 + #3 } } { \fp_to_dim:n { #2 + #4 } }
527     \__draw_path_ellipse_arci:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
528     \__draw_path_ellipse_arcii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
529     \__draw_path_ellipse_arciiii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
530     \__draw_path_ellipse_arciv:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
531   }
532   \__draw_softpath_closepath:
533   \__draw_path_moveto:nn {#1} {#2}
534 }
535 \cs_new:Npn \__draw_path_ellipse_arci:nnnnnn #1#2#3#4#5#6
536 {
537   \__draw_path_curveto:nnnnnn
538   { \fp_to_dim:n { #1 + #3 + #5 * \c__draw_path_ellipse_fp } }
539   { \fp_to_dim:n { #2 + #4 + #6 * \c__draw_path_ellipse_fp } }
540   { \fp_to_dim:n { #1 + #3 * \c__draw_path_ellipse_fp + #5 } }
541   { \fp_to_dim:n { #2 + #4 * \c__draw_path_ellipse_fp + #6 } }

```

```

542     { \fp_to_dim:n { #1 + #5 } }
543     { \fp_to_dim:n { #2 + #6 } }
544 }
545 \cs_new:Npn \__draw_path_ellipse_arci:nnnnnn #1#2#3#4#5#6
546 {
547   \__draw_path_curveto:nnnnnn
548   { \fp_to_dim:n { #1 - #3 * \c__draw_path_ellipse_fp + #5 } }
549   { \fp_to_dim:n { #2 - #4 * \c__draw_path_ellipse_fp + #6 } }
550   { \fp_to_dim:n { #1 - #3 + #5 * \c__draw_path_ellipse_fp } }
551   { \fp_to_dim:n { #2 - #4 + #6 * \c__draw_path_ellipse_fp } }
552   { \fp_to_dim:n { #1 - #3 } }
553   { \fp_to_dim:n { #2 - #4 } }
554 }
555 \cs_new:Npn \__draw_path_ellipse_arci:nnnnnn #1#2#3#4#5#6
556 {
557   \__draw_path_curveto:nnnnnn
558   { \fp_to_dim:n { #1 - #3 - #5 * \c__draw_path_ellipse_fp } }
559   { \fp_to_dim:n { #2 - #4 - #6 * \c__draw_path_ellipse_fp } }
560   { \fp_to_dim:n { #1 - #3 * \c__draw_path_ellipse_fp - #5 } }
561   { \fp_to_dim:n { #2 - #4 * \c__draw_path_ellipse_fp - #6 } }
562   { \fp_to_dim:n { #1 - #5 } }
563   { \fp_to_dim:n { #2 - #6 } }
564 }
565 \cs_new:Npn \__draw_path_ellipse_arci:nnnnnn #1#2#3#4#5#6
566 {
567   \__draw_path_curveto:nnnnnn
568   { \fp_to_dim:n { #1 + #3 * \c__draw_path_ellipse_fp - #5 } }
569   { \fp_to_dim:n { #2 + #4 * \c__draw_path_ellipse_fp - #6 } }
570   { \fp_to_dim:n { #1 + #3 - #5 * \c__draw_path_ellipse_fp } }
571   { \fp_to_dim:n { #2 + #4 - #6 * \c__draw_path_ellipse_fp } }
572   { \fp_to_dim:n { #1 + #3 } }
573   { \fp_to_dim:n { #2 + #4 } }
574 }
575 \fp_const:Nn \c__draw_path_ellipse_fp { \fp_use:c { c__draw_path_arc_90_fp } }

```

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

\draw_path_circle:nn A shortcut.

```

576 \cs_new_protected:Npn \draw_path_circle:nn #1#2
577 { \draw_path_ellipse:nnn {#1} { #2 , Opt } { Opt , #2 } }

```

(End of 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
578 \cs_new_protected:Npn \draw_path_rectangle:nn #1#2
579 {
580   \__draw_point_process:nnn
581   {
582     \bool_lazy_or:nnTF
583     { \l__draw_corner_arc_bool }
584     { \l__draw_matrix_active_bool }

```

```

585         { \_draw_path_rectangle_rounded:nnnn }
586         { \_draw_path_rectangle:nnnn }
587     }
588     {#1}
589     {#2}
590 }
591 \cs_new_protected:Npn \_draw_path_rectangle:nnnn #1#2#3#4
592 {
593     \_draw_path_update_limits:nn {#1} {#2}
594     \_draw_path_update_limits:nn { #1 + #3 } { #2 + #4 }
595     \_draw_softpath_rectangle:nnnn {#1} {#2} {#3} {#4}
596     \_draw_path_update_last:nn {#1} {#2}
597 }
598 \cs_new_protected:Npn \_draw_path_rectangle_rounded:nnnn #1#2#3#4
599 {
600     \draw_path_moveto:n { #1 + #3 , #2 + #4 }
601     \draw_path_lineto:n { #1 , #2 + #4 }
602     \draw_path_lineto:n { #1 , #2 }
603     \draw_path_lineto:n { #1 + #3 , #2 }
604     \draw_path_close:
605     \draw_path_moveto:n { #1 , #2 }
606 }

```

(End of 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
607 \cs_new_protected:Npn \draw_path_rectangle_corners:nn #1#2
608 {
609     \_draw_point_process:nnn
610     { \_draw_path_rectangle_corners:nnnnn {#1} }
611     {#1} {#2}
612 }
613 \cs_new_protected:Npn \_draw_path_rectangle_corners:nnnnn #1#2#3#4#5
614 { \draw_path_rectangle:nn {#1} { #4 - #2 , #5 - #3 } }

```

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

```

\_draw_path_grid_auxi:nnnnnn
\_draw_path_grid_auxi:eennnn
\_draw_path_grid_auxii:nnnnnn
\_draw_path_grid_auxiii:nnnnnn
\_draw_path_grid_auxiiii:eennnn
\_draw_path_grid_auxiv:nnnnnnnn
\_draw_path_grid_auxiv:eennnnnn
615 \cs_new_protected:Npn \draw_path_grid:nnnn #1#2#3#4
616 {
617     \_draw_point_process:nnn
618     {
619         \_draw_path_grid_auxi:eennnn
620         { \dim_eval:n { \dim_abs:n {#1} } } }
621         { \dim_eval:n { \dim_abs:n {#2} } } }
622     }
623     {#3} {#4}
624 }
625 \cs_new_protected:Npn \_draw_path_grid_auxi:nnnnnn #1#2#3#4#5#6

```



```

626 {
627   \dim_compare:nNnTF {#3} > {#5}
628     { \__draw_path_grid_auxii:nnnnnn {#1} {#2} {#5} {#4} {#3} {#6} }
629     { \__draw_path_grid_auxii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6} }
630 }
631 \cs_generate_variant:Nn \__draw_path_grid_auxi:nnnnnn { ee }
632 \cs_new_protected:Npn \__draw_path_grid_auxii:nnnnnn #1#2#3#4#5#6
633 {
634   \dim_compare:nNnTF {#4} > {#6}
635     { \__draw_path_grid_auxiii:nnnnnn {#1} {#2} {#3} {#6} {#5} {#4} }
636     { \__draw_path_grid_auxiii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6} }
637 }
638 \cs_new_protected:Npn \__draw_path_grid_auxiii:nnnnnn #1#2#3#4#5#6
639 {
640   \__draw_path_grid_auxiv:eennnnnn
641   { \fp_to_dim:n { #1 * trunc(#3/(#1)) } }
642   { \fp_to_dim:n { #2 * trunc(#4/(#2)) } }
643   {#1} {#2} {#3} {#4} {#5} {#6}
644 }
645 \cs_new_protected:Npn \__draw_path_grid_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
646 {
647   \dim_step_inline:nnnn
648   {#1}
649   {#3}
650   {#7}
651   {
652     \draw_path_moveto:n { ##1 , #6 }
653     \draw_path_lineto:n { ##1 , #8 }
654   }
655   \dim_step_inline:nnnn
656   {#2}
657   {#4}
658   {#8}
659   {
660     \draw_path_moveto:n { #5 , ##1 }
661     \draw_path_lineto:n { #7 , ##1 }
662   }
663 }
664 \cs_generate_variant:Nn \__draw_path_grid_auxiv:nnnnnnnn { ee }

```

(End of 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
665 \bool_new:N \l__draw_path_use_clip_bool
666 \bool_new:N \l__draw_path_use_fill_bool
667 \bool_new:N \l__draw_path_use_stroke_bool

```

(End of 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
668 \bool_new:N \l__draw_path_use_bb_bool
669 \bool_new:N \l__draw_path_use_clear_bool

```

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

\draw_path_use:n 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_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
670 \cs_new_protected:Npn \draw_path_use:n #1
671 {
672   \tl_if_blank:nF {#1}
673     { \__draw_path_use:n {#1} }
674 }
675 \cs_new_protected:Npn \draw_path_use_clear:n #1
676 {
677   \bool_lazy_or:nnTF
678     { \tl_if_blank_p:n {#1} }
679     { \str_if_eq_p:nn {#1} { clear } }
680   {
681     \__draw_softpath_clear:
682     \__draw_path_reset_limits:
683   }
684   { \__draw_path_use:n { #1 , clear } }
685 }

```

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).

```

686 \cs_new_protected:Npn \__draw_path_use:n #1
687 {
688   \bool_set_false:N \l__draw_path_use_clip_bool
689   \bool_set_false:N \l__draw_path_use_fill_bool
690   \bool_set_false:N \l__draw_path_use_stroke_bool
691   \clist_map_inline:nn {#1}
692   {
693     \cs_if_exist:cTF { l__draw_path_use_ ##1 _ bool }
694       { \bool_set_true:c { l__draw_path_use_ ##1 _ bool } }
695       {
696         \cs_if_exist_use:cF { __draw_path_use_action_ ##1 : }
697         { \msg_error:nnn { draw } { invalid-path-action } {##1} }
698       }
699   }
700   \__draw_softpath_round_corners:
701   \bool_lazy_and:nnT
702     { \l_draw_bb_update_bool }
703     { \l__draw_path_use_stroke_bool }
704     { \__draw_path_use_stroke_bb: }
705   \__draw_softpath_use:
706   \bool_if:NT \l__draw_path_use_clip_bool
707   {
708     \__draw_backend_clip:
709     \bool_set_false:N \l_draw_bb_update_bool
710     \bool_lazy_or:nnF
711       { \l__draw_path_use_fill_bool }
712       { \l__draw_path_use_stroke_bool }
713       { \__draw_backend_discardpath: }
714   }

```

```

715 \bool_lazy_or:nnT
716 { \l__draw_path_use_fill_bool }
717 { \l__draw_path_use_stroke_bool }
718 {
719   \use:c
720   {
721     __draw_backend_
722     \bool_if:NT \l__draw_path_use_fill_bool { fill }
723     \bool_if:NT \l__draw_path_use_stroke_bool { stroke }
724     :
725   }
726 }
727 \bool_if:NT \l__draw_path_use_clear_bool
728 { \__draw_softpath_clear: }
729 }
730 \cs_new_protected:Npn \__draw_path_use_action_draw:
731 {
732   \bool_set_true:N \l__draw_path_use_stroke_bool
733 }
734 \cs_new_protected:Npn \__draw_path_use_action_fillstroke:
735 {
736   \bool_set_true:N \l__draw_path_use_fill_bool
737   \bool_set_true:N \l__draw_path_use_stroke_bool
738 }

```

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.

```

739 \cs_new_protected:Npn \__draw_path_use_stroke_bb:
740 {
741   \__draw_path_use_stroke_bb_aux:NnN x { max } +
742   \__draw_path_use_stroke_bb_aux:NnN y { max } +
743   \__draw_path_use_stroke_bb_aux:NnN x { min } -
744   \__draw_path_use_stroke_bb_aux:NnN y { min } -
745 }
746 \cs_new_protected:Npn \__draw_path_use_stroke_bb_aux:NnN #1#2#3
747 {
748   \dim_compare:nNnF { \dim_use:c { g__draw_ #1#2 _dim } } = { #3 -\c_max_dim }
749   {
750     \dim_gset:cn { g__draw_ #1#2 _dim }
751     {
752       \use:c { dim_ #2 :nn }
753       { \dim_use:c { g__draw_ #1#2 _dim } }
754       {
755         \dim_use:c { g__draw_path_ #1#2 _dim }
756         #3 0.5 \g__draw_linewidth_dim
757       }
758     }
759   }
760 }

```

(End of 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_lastx_dim
\l__draw_path_lasty_dim
\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
761 \dim_new:N \l__draw_path_lastx_dim
762 \dim_new:N \l__draw_path_lasty_dim
763 \dim_new:N \l__draw_path_xmax_dim
764 \dim_new:N \l__draw_path_xmin_dim
765 \dim_new:N \l__draw_path_ymax_dim
766 \dim_new:N \l__draw_path_ymin_dim
767 \dim_new:N \l__draw_softpath_lastx_dim
768 \dim_new:N \l__draw_softpath_lasty_dim
769 \bool_new:N \l__draw_softpath_corners_bool

```

(End of 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.

```

\draw_path_scope_end:
770 \cs_new_protected:Npn \draw_path_scope_begin:
771 {
772   \group_begin:
773   \dim_set_eq:NN \l__draw_path_lastx_dim \g__draw_path_lastx_dim
774   \dim_set_eq:NN \l__draw_path_lasty_dim \g__draw_path_lasty_dim
775   \dim_set_eq:NN \l__draw_path_xmax_dim \g__draw_path_xmax_dim
776   \dim_set_eq:NN \l__draw_path_xmin_dim \g__draw_path_xmin_dim
777   \dim_set_eq:NN \l__draw_path_ymax_dim \g__draw_path_ymax_dim
778   \dim_set_eq:NN \l__draw_path_ymin_dim \g__draw_path_ymin_dim
779   \dim_set_eq:NN \l__draw_softpath_lastx_dim \g__draw_softpath_lastx_dim
780   \dim_set_eq:NN \l__draw_softpath_lasty_dim \g__draw_softpath_lasty_dim
781   \__draw_path_reset_limits:
782   \__draw_softpath_save:
783 }
784 \cs_new_protected:Npn \draw_path_scope_end:
785 {
786   \__draw_softpath_restore:
787   \dim_gset_eq:NN \g__draw_softpath_lastx_dim \l__draw_softpath_lastx_dim
788   \dim_gset_eq:NN \g__draw_softpath_lasty_dim \l__draw_softpath_lasty_dim
789   \dim_gset_eq:NN \g__draw_path_xmax_dim \l__draw_path_xmax_dim
790   \dim_gset_eq:NN \g__draw_path_xmin_dim \l__draw_path_xmin_dim
791   \dim_gset_eq:NN \g__draw_path_ymax_dim \l__draw_path_ymax_dim
792   \dim_gset_eq:NN \g__draw_path_ymin_dim \l__draw_path_ymin_dim
793   \dim_gset_eq:NN \g__draw_path_lastx_dim \l__draw_path_lastx_dim
794   \dim_gset_eq:NN \g__draw_path_lasty_dim \l__draw_path_lasty_dim
795   \group_end:
796 }

```

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

```

797 \msg_new:nnnn { draw } { invalid-path-action }
798 { Invalid-action-~'#1'-for-path. }
799 { Paths-can-be-used-with-actions-'draw',~'clip',~'fill'~or~'stroke'. }
800 % \end{macrocode}
801 %

```

```

802 % \begin{macrocode}
803 </package>

```

5 l3draw-points implementation

```

804 <*package>
805 <@@=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 `e`-type expansion of a point expression.
- `\pgfgetlastxy`: Unused in the entire pgf core, may be emulated by `e`-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

```

\__draw_point_process:nn Execute whatever code is passed to extract the  $x$  and  $y$  co-ordinates. The first argument
    \__draw_point_process_auxi:nn here should itself absorb two arguments. There is also a version to deal with two co-
    \__draw_point_process_auxi:en ordinates: common enough to justify a separate function.
    \__draw_point_process_auxii:nw
\__draw_point_process:nnn
    \__draw_point_process_auxiii:nn
    \__draw_point_process_auxiii:een
    \__draw_point_process_auxiv:nw
\__draw_point_process:nnnn
    \__draw_point_process_auxv:nnnn
    \__draw_point_process_auxv:eeen
    \__draw_point_process_auxvi:nw
\__draw_point_process:nnnnn
    \__draw_point_process_auxvii:nnnnn
    \__draw_point_process_auxvii:eeeen
    \__draw_point_process_auxviii:nw

```

```

806 \cs_new:Npn \__draw_point_process:nn #1#2
807 {
808     \__draw_point_process_auxi:en
809     { \draw_point:n {#2} }
810     {#1}
811 }
812 \cs_new:Npn \__draw_point_process_auxi:nn #1#2
813 { \__draw_point_process_auxii:nw {#2} #1 \s__draw_stop }
814 \cs_generate_variant:Nn \__draw_point_process_auxi:nn { e }
815 \cs_new:Npn \__draw_point_process_auxii:nw #1 #2 , #3 \s__draw_stop
816 { #1 {#2} {#3} }
817 \cs_new:Npn \__draw_point_process:nnn #1#2#3

```

```

818 {
819   \_draw_point_process_auxiii:een
820   { \draw_point:n {#2} }
821   { \draw_point:n {#3} }
822   {#1}
823 }
824 \cs_new:Npn \_draw_point_process_auxiii:nnn #1#2#3
825 { \_draw_point_process_auxiv:nw {#3} #1 \s__draw_mark #2 \s__draw_stop }
826 \cs_generate_variant:Nn \_draw_point_process_auxiii:nnn { ee }
827 \cs_new:Npn \_draw_point_process_auxiv:nw #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_stop
828 { #1 {#2} {#3} {#4} {#5} }
829 \cs_new:Npn \_draw_point_process:nnnn #1#2#3#4
830 {
831   \_draw_point_process_auxv:eeen
832   { \draw_point:n {#2} }
833   { \draw_point:n {#3} }
834   { \draw_point:n {#4} }
835   {#1}
836 }
837 \cs_new:Npn \_draw_point_process_auxv:nnnn #1#2#3#4
838 { \_draw_point_process_auxvi:nw {#4} #1 \s__draw_mark #2 \s__draw_mark #3 \s__draw_stop }
839 \cs_generate_variant:Nn \_draw_point_process_auxv:nnnn { eee }
840 \cs_new:Npn \_draw_point_process_auxvi:nw
841   #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_mark #6 , #7 \s__draw_stop
842   { #1 {#2} {#3} {#4} {#5} {#6} {#7} }
843 \cs_new:Npn \_draw_point_process:nnnnn #1#2#3#4#5
844 {
845   \_draw_point_process_auxvii:eeeen
846   { \draw_point:n {#2} }
847   { \draw_point:n {#3} }
848   { \draw_point:n {#4} }
849   { \draw_point:n {#5} }
850   {#1}
851 }
852 \cs_new:Npn \_draw_point_process_auxvii:nnnnn #1#2#3#4#5
853 {
854   \_draw_point_process_auxviii:nw
855   {#5} #1 \s__draw_mark #2 \s__draw_mark #3 \s__draw_mark #4 \s__draw_stop
856 }
857 \cs_generate_variant:Nn \_draw_point_process_auxvii:nnnnn { eeee }
858 \cs_new:Npn \_draw_point_process_auxviii:nw
859   #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_mark #6 , #7 \s__draw_mark #8 , #9 \s__draw_stop
860   { #1 {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9} }

```

(End of definition for _draw_point_process:nn and others.)

5.2 Basic points

\draw_point:n Co-ordinates are always returned as two dimensions.

```

\_draw_point_to_dim:n 861 \cs_new:Npn \draw_point:n #1
\_draw_point_to_dim:e 862 { \_draw_point_to_dim:e { \fp_eval:n {#1} } }
\_draw_point_to_dim:w 863 \cs_new:Npn \_draw_point_to_dim:n #1
864 { \_draw_point_to_dim:w #1 }
865 \cs_generate_variant:Nn \_draw_point_to_dim:n { e }

```

```
866 \cs_new:Npn \__draw_point_to_dim:w ( #1 , ~ #2 ) { #1pt , #2pt }
```

5.3 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:enn
867 \cs_new:Npn \draw_point_polar:nn #1#2
868 { \draw_point_polar:nnn {#1} {#1} {#2} }
869 \cs_new:Npn \draw_point_polar:nnn #1#2#3
870 { \__draw_draw_polar:enn { \fp_eval:n {#3} } {#1} {#2} }
871 \cs_new:Npn \__draw_draw_polar:nnn #1#2#3
872 { \draw_point:n { cosd(#1) * (#2) , sind(#1) * (#3) } }
873 \cs_generate_variant:Nn \__draw_draw_polar:nnn { e }
```

5.4 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.*

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

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

```
874 \cs_new:Npn \draw_point_unit_vector:n #1
875 { \__draw_point_process:nn { \__draw_point_unit_vector:nn } {#1} }
876 \cs_new:Npn \__draw_point_unit_vector:nn #1#2
877 {
878   \__draw_point_unit_vector:nnn
879   { \fp_eval:n { (sqrt(#1 * #1 + #2 * #2)) } }
880   {#1} {#2}
881 }
882 \cs_new:Npn \__draw_point_unit_vector:nnn #1#2#3
883 {
884   \fp_compare:nNnTF {#1} = \c_zero_fp
885   { Opt, 1pt }
886   {
887     \draw_point:n
888     { ( #2 , #3 ) / #1 }
889   }
890 }
891 \cs_generate_variant:Nn \__draw_point_unit_vector:nnn { e }
```

5.5 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

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

```
\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:eeeeee
```

and

$$P_y = \frac{(x_1y_2 - y_1x_2)(y_3 - y_5) - (x_3y_4 - y_3x_4)(y_1 - y_2)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}$$

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.)

```

892 \cs_new:Npn \draw_point_intersect_lines:nnnn #1#2#3#4
893 {
894   \__draw_point_process:nnnnn
895   { \__draw_point_intersect_lines:nnnnnnnn }
896   {#1} {#2} {#3} {#4}
897 }

```

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.

```

898 \cs_new:Npn \__draw_point_intersect_lines:nnnnnnnn #1#2#3#4#5#6#7#8
899 {
900   \__draw_point_intersect_lines_aux:eeeeee
901   { \fp_eval:n { #1 * #4 - #2 * #3 } }
902   { \fp_eval:n { #5 * #8 - #6 * #7 } }
903   { \fp_eval:n { #1 - #3 } }
904   { \fp_eval:n { #5 - #7 } }
905   { \fp_eval:n { #2 - #4 } }
906   { \fp_eval:n { #6 - #8 } }
907 }
908 \cs_new:Npn \__draw_point_intersect_lines_aux:nnnnnn #1#2#3#4#5#6
909 {
910   \draw_point:n
911   {
912     ( #2 * #3 - #1 * #4 , #2 * #5 - #1 * #6 )
913     / ( #4 * #5 - #6 * #3 )
914   }
915 }
916 \cs_generate_variant:Nn \__draw_point_intersect_lines_aux:nnnnnn { eeeeeee }

```



```

\draw_point_intersect_circles:nnnnn
__draw_point_intersect_circles_auxi:nnnnnnn
draw_point_intersect_circles_auxii:nnnnnnn
draw_point_intersect_circles_auxiii:eennnnn
draw_point_intersect_circles_auxiiii:nnnnnnn
draw_point_intersect_circles_auxiii:eennnnn
draw_point_intersect_circles_auxiv:nnnnnnn
draw_point_intersect_circles_auxiv:ennnnnnn
draw_point_intersect_circles_auxv:nnnnnnnnn
draw_point_intersect_circles_auxv:eennnnnnn
draw_point_intersect_circles_auxvi:nnnnnnnnn
draw_point_intersect_circles_auxvi:ennnnnnnn
draw_point_intersect_circles_auxvii:nnnnnnnn
draw_point_intersect_circles_auxvii:eeennnn

```

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.

```

917 \cs_new:Npn \draw_point_intersect_circles:nnnnn #1#2#3#4#5
918 {
919   \__draw_point_process:nnn
920   { \__draw_point_intersect_circles_auxi:nnnnnnn {#2} {#4} {#5} }
921   {#1} {#3}
922 }
923 \cs_new:Npn \__draw_point_intersect_circles_auxi:nnnnnnn #1#2#3#4#5#6#7
924 {
925   \__draw_point_intersect_circles_auxii:eennnnnn
926   { \fp_eval:n {#1} } { \fp_eval:n {#2} } {#4} {#5} {#6} {#7} {#3}
927 }

```

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.

```

928 \cs_new:Npn \__draw_point_intersect_circles_auxii:nnnnnnn #1#2#3#4#5#6#7
929 {
930   \__draw_point_intersect_circles_auxiii:ennnnnnn
931   { \fp_eval:n { #5 - #3 } }
932   { \fp_eval:n { #6 - #4 } }
933   {#1} {#2} {#3} {#4} {#7}
934 }
935 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxii:nnnnnnn { ee }
936 \cs_new:Npn \__draw_point_intersect_circles_auxiii:nnnnnnn #1#2#3#4#5#6#7
937 {
938   \__draw_point_intersect_circles_auxiv:ennnnnnn
939   { \fp_eval:n { sqrt( #1 * #1 + #2 * #2 ) } }
940   {#1} {#2} {#3} {#4} {#5} {#6} {#7}
941 }
942 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxiii:nnnnnnn { ee }

```

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.

```

943 \cs_new:Npn \__draw_point_intersect_circles_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
944 {
945   \__draw_point_intersect_circles_auxv:ennnnnnnnn
946   { \fp_eval:n { 1 / #1 } }
947   { \fp_eval:n { #4 * #4 } }
948   {#1} {#2} {#3} {#5} {#6} {#7} {#8}
949 }
950 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxiv:nnnnnnnn { e }
951 \cs_new:Npn \__draw_point_intersect_circles_auxv:nnnnnnnnn #1#2#3#4#5#6#7#8#9
952 {
953   \__draw_point_intersect_circles_auxvi:ennnnnnnnn
954   { \fp_eval:n { 0.5 * #1 * ( #2 + #3 * #3 - #6 * #6 ) } }
955   {#1} {#2} {#4} {#5} {#7} {#8} {#9}
956 }
957 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxv:nnnnnnnnn { ee }

```

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.

```

958 \cs_new:Npn \__draw_point_intersect_circles_auxvi:nnnnnnnn #1#2#3#4#5#6#7#8
959 {
960   \__draw_point_intersect_circles_auxvii:eeennnnn
961   { \fp_eval:n { #1 * #2 } }
962   { \int_if_odd:nTF {#8} { 1 } { -1 } }
963   { \fp_eval:n { sqrt ( #3 - #1 * #1 ) * #2 } }
964   {#4} {#5} {#6} {#7}
965 }
966 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxvi:nnnnnnnn { e }
967 \cs_new:Npn \__draw_point_intersect_circles_auxvii:nnnnnnnn #1#2#3#4#5#6#7
968 {
969   \draw_point:n
970   { #6 + #4 * #1 + #2 * #3 * #5 , #7 + #5 * #1 + -1 * #2 * #3 * #4 }
971 }
972 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxvii:nnnnnnnn { eee }

```

The intersection points P_1 and P_2 between a line joining points (x_1, y_1) and (x_2, y_2) and a circle with center (x_3, y_3) and radius r . We use the intermediate values

```

\draw_point_intersect_line_circle:nnnnn
w_point_intersect_line_circle_auxi:nnnnnnnn
_point_intersect_line_circle_auxii:nnnnnnnn
_point_intersect_line_circle_auxiii:ennnnnnnn
_point_intersect_line_circle_auxiii:nnnnnnnn
_point_intersect_line_circle_auxiii:eeennnnnn
_point_intersect_line_circle_auxiv:nnnnnnnn
_point_intersect_line_circle_auxiv:eeennnnnn
draw_point_intersect_line_circle_auxv:nnnnn
draw_point_intersect_line_circle_auxv:ennnn

```

$$\begin{aligned}
a &= (x_2 - x_1)^2 + (y_2 - y_1)^2 \\
b &= 2 \times ((x_2 - x_1) \times (x_1 - x_3) + (y_2 - y_1) \times (y_1 - y_3)) \\
c &= x_3^2 + y_3^2 + x_1^2 + y_1^2 - 2 \times (x_3 \times x_1 + y_3 \times y_1) - r^2 \\
d &= b^2 - 4 \times a \times c \\
\mu_1 &= \frac{-b + \sqrt{d}}{2 \times a} \\
\mu_2 &= \frac{-b - \sqrt{d}}{2 \times a}
\end{aligned}$$

in either

$$\begin{aligned}
P_{1x} &= x_1 + \mu_1 \times (x_2 - x_1) \\
P_{1y} &= y_1 + \mu_1 \times (y_2 - y_1)
\end{aligned}$$

or

$$\begin{aligned}
P_{2x} &= x_1 + \mu_2 \times (x_2 - x_1) \\
P_{2y} &= y_1 + \mu_2 \times (y_2 - y_1)
\end{aligned}$$

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

```

973 \cs_new:Npn \draw_point_intersect_line_circle:nnnnn #1#2#3#4#5
974 {
975   \__draw_point_process:nnnn
976   { \__draw_point_intersect_line_circle_auxi:nnnnnnnn {#4} {#5} }
977   {#1} {#2} {#3}
978 }
979 \cs_new:Npn \__draw_point_intersect_line_circle_auxi:nnnnnnnn #1#2#3#4#5#6#7#8
980 {
981   \__draw_point_intersect_line_circle_auxii:ennnnnnnn
982   { \fp_eval:n {#1} } {#3} {#4} {#5} {#6} {#7} {#8} {#2}
983 }

```

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

```
#1 r
#2 x1
#3 y1
#4 x2
#5 y2
#6 x3
#7 y3
#8 n
```

Once we evaluate a , b and c , the co-ordinate (x_3, y_3) and r are no longer required: handy as we will need various intermediate values in the following.

```
984 \cs_new:Npn \__draw_point_intersect_line_circle_auxii:nnnnnnnn #1#2#3#4#5#6#7#8
985 {
986   \__draw_point_intersect_line_circle_auxiii:eeennnnnn
987   { \fp_eval:n { (#4-#2)*(#4-#2)+(#5-#3)*(#5-#3) } }
988   { \fp_eval:n { 2*((#4-#2)*(#2-#6)+(#5-#3)*(#3-#7)) } }
989   { \fp_eval:n { (#6*#6+#7*#7)+(#2*#2+#3*#3)-(2*(#6*#2+#7*#3))-(#1*#1) } }
990   {#2} {#3} {#4} {#5} {#8}
991 }
992 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxii:nnnnnnnn { e }
```

then we can get $d = b^2 - 4 \times a \times c$ and the usage of n .

```
993 \cs_new:Npn \__draw_point_intersect_line_circle_auxiii:nnnnnnnn #1#2#3#4#5#6#7#8
994 {
995   \__draw_point_intersect_line_circle_auxiv:eeennnnnn
996   { \fp_eval:n { #2 * #2 - 4 * #1 * #3 } }
997   { \int_if_odd:nTF {#8} { 1 } { -1 } }
998   {#1} {#2} {#4} {#5} {#6} {#7}
999 }
1000 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxiii:nnnnnnnn { eee }
```

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 a
#2 b
#3 c
#4 d
#5 ±(the usage of n)
#6 x1
#7 y1
#8 x2
```

#9 y_2

There are some final pre-calculations, $\mu = \frac{-b \pm \sqrt{d}}{2 \times a}$ then, we can yield a result.

```

1001 \cs_new:Npn \__draw_point_intersect_line_circle_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
1002 {
1003   \__draw_point_intersect_line_circle_auxv:ennnn
1004   { \fp_eval:n { (-1 * #4 + #2 * sqrt(#1)) / (2 * #3) } }
1005   {#5} {#6} {#7} {#8}
1006 }
1007 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxiv:nnnnnnnn { ee }
1008 \cs_new:Npn \__draw_point_intersect_line_circle_auxv:nnnnn #1#2#3#4#5
1009 {
1010   \draw_point:n
1011   { #2 + #1 * (#4 - #2), #3 + #1 * (#5 - #3) }
1012 }
1013 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxv:nnnnn { e }

```

5.6 Interpolation on a line (vector) or arc

Simple maths after expansion.

```

\draw_point_interpolate_line:nnn 1014 \cs_new:Npn \draw_point_interpolate_line:nnn #1#2#3
\_draw_point_interpolate_line_aux:nnnnn 1015 {
\_draw_point_interpolate_line_aux:ennnn 1016   \__draw_point_process:nnn
\_draw_point_interpolate_line_aux:nnnnnn 1017   { \__draw_point_interpolate_line_aux:ennnn { \fp_eval:n {#1} } }
\_draw_point_interpolate_line_aux:ennnnnn 1018   {#2} {#3}
1019 }
1020 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnn #1#2#3#4#5
1021 {
1022   \__draw_point_interpolate_line_aux:ennnnn { \fp_eval:n { 1 - #1 } }
1023   {#1} {#2} {#3} {#4} {#5}
1024 }
1025 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnn { e }
1026 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnnn #1#2#3#4#5#6
1027 { \draw_point:n { #2 * #3 + #1 * #5 , #2 * #4 + #1 * #6 } }
1028 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnnn { e }

```

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:nnn 1029 \cs_new:Npn \draw_point_interpolate_distance:nnn #1#2#3
\_draw_point_interpolate_distance:nnnnn 1030 {
\_draw_point_interpolate_distance:nnnnnn 1031   \__draw_point_process:nn
\_draw_point_interpolate_distance:ennnnnn 1032   { \__draw_point_interpolate_distance:nnnn {#1} {#3} }
1033   {#2}
1034 }
1035 \cs_new:Npn \__draw_point_interpolate_distance:nnnn #1#2#3#4
1036 {
1037   \__draw_point_process:nn
1038   {
1039     \__draw_point_interpolate_distance:ennnn
1040     { \fp_eval:n {#1} } {#3} {#4}
1041   }
1042   { \draw_point_unit_vector:n { ( #2 ) - ( #3 , #4 ) } }
1043 }

```

```

1044 \cs_new:Npn \__draw_point_interpolate_distance:nnnnn #1#2#3#4#5
1045 { \draw_point:n { #2 + #1 * #4 , #3 + #1 * #5 } }
1046 \cs_generate_variant:Nn \__draw_point_interpolate_distance:nnnnn { e }

```

(End of definition for \draw_point: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_auxiii:nnnnnnnnnn
\draw_point_interpolate_arcaxes_auxiv:nnnnnnnnnn
\draw_point_interpolate_arcaxes_auxv:nnnnnnnnnn

```

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.

```

1047 \cs_new:Npn \draw_point_interpolate_arcaxes:nnnnnn #1#2#3#4#5#6
1048 {
1049   \__draw_point_process:nnnn
1050   { \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn {#1} {#5} {#6} }
1051   {#2} {#3} {#4}
1052 }
1053 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
1054 {
1055   \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnnn
1056   { \fp_eval:n {#1} } {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1057 }

```

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.

```

1058 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
1059 {
1060   \__draw_point_interpolate_arcaxes_auxiii:nnnnnnnnnn
1061   { \fp_eval:n { #1 * (#3) + ( 1 - #1 ) * (#2) } }
1062   {#4} {#5} {#6} {#7} {#8} {#9}
1063 }
1064 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnnn { e }
1065 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnnn #1#2#3#4#5#6#7
1066 {
1067   \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnnnn
1068   { \fp_eval:n { cosd (#1) } }
1069   { \fp_eval:n { sind (#1) } }
1070   {#2} {#3} {#4} {#5} {#6} {#7}
1071 }

```

```

1072 \cs_generate_variant:Nn \_draw_point_interpolate_arcaxes_auxiii:nnnnnnn { e }
1073 \cs_new:Npn \_draw_point_interpolate_arcaxes_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
1074 {
1075   \draw_point:n
1076     { #3 + #1 * #5 + #2 * #7 , #4 + #1 * #6 + #2 * #8 }
1077 }
1078 \cs_generate_variant:Nn \_draw_point_interpolate_arcaxes_auxiv:nnnnnnnn { ee }

```

(End of 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:nnnnnnnnn
\draw_point_interpolate_curve_auxiiii:nnnnnnn
\draw_point_interpolate_curve_auxiii:nnnnnnn
\draw_point_interpolate_curve_auxiv:nnnnnnn
\draw_point_interpolate_curve_auxv:nnw
\draw_point_interpolate_curve_auxv:eev
\draw_point_interpolate_curve_auxvi:n
draw_point_interpolate_curve_auxvii:nnnnnnnnn
draw_point_interpolate_curve_auxviii:nnnnnnn
draw_point_interpolate_curve_auxviii:eeennnn

```

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.

```

1079 \cs_new:Npn \draw_point_interpolate_curve:nnnnnn #1#2#3#4#5
1080 {
1081   \_draw_point_process:nnnnnn
1082     { \_draw_point_interpolate_curve_auxi:nnnnnnnnn {#1} }
1083     {#2} {#3} {#4} {#5}
1084 }
1085 \cs_new:Npn \_draw_point_interpolate_curve_auxi:nnnnnnnnn #1#2#3#4#5#6#7#8#9
1086 {
1087   \_draw_point_interpolate_curve_auxii:nnnnnnnnn
1088     { \fp_eval:n {#1} }
1089     {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1090 }

```

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

$$P_x = (1 - p)x_1'' + px_2''$$

$$P_y = (1 - p)y_1'' + py_2''$$

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

```

1091 \cs_new:Npn \__draw_point_interpolate_curve_auxiii:nnnnnnnnn
1092   #1#2#3#4#5#6#7#8#9
1093   {
1094     \__draw_point_interpolate_curve_auxiii:ennnnnn
1095     { \fp_eval:n { 1 - #1 } }
1096     {#1}
1097     { {#2} {#3} } { {#4} {#5} } { {#6} {#7} } { {#8} {#9} }
1098   }
1099 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxiii:nnnnnnnnn { e }
1100 % \begin{macrocode}
1101 % We need to do the first cycle, but haven't got enough arguments to keep
1102 % everything in play at once. So here we use a bit of argument re-ordering
1103 % and a single auxiliary to get the job done.
1104 % \begin{macrocode}
1105 \cs_new:Npn \__draw_point_interpolate_curve_auxiii:nnnnnn #1#2#3#4#5#6
1106   {
1107     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #3 #4
1108     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #4 #5
1109     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #5 #6
1110     \prg_do_nothing:
1111     \__draw_point_interpolate_curve_auxvi:n { {#1} {#2} }
1112   }
1113 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxiii:nnnnnn { e }
1114 \cs_new:Npn \__draw_point_interpolate_curve_auxiv:nnnnnn #1#2#3#4#5#6
1115   {
1116     \__draw_point_interpolate_curve_auxv:eev
1117     { \fp_eval:n { #1 * #3 + #2 * #5 } }
1118     { \fp_eval:n { #1 * #4 + #2 * #6 } }
1119   }
1120 \cs_new:Npn \__draw_point_interpolate_curve_auxv:nnw
1121   #1#2#3 \prg_do_nothing: #4#5
1122   {
1123     #3
1124     \prg_do_nothing:
1125     #4 { #5 {#1} {#2} }
1126   }
1127 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxv:nnw { ee }
1128 % \begin{macrocode}
1129 % Get the arguments back into the right places and to the second and
1130 % third cycles directly.
1131 % \begin{macrocode}
1132 \cs_new:Npn \__draw_point_interpolate_curve_auxvi:n #1
1133   { \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1 }
1134 \cs_new:Npn \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1#2#3#4#5#6#7#8
1135   {
1136     \__draw_point_interpolate_curve_auxviii:eeeenn
1137     { \fp_eval:n { #1 * #5 + #2 * #3 } }
1138     { \fp_eval:n { #1 * #6 + #2 * #4 } }

```



```

1139     { \fp_eval:n { #1 * #7 + #2 * #5 } }
1140     { \fp_eval:n { #1 * #8 + #2 * #6 } }
1141     {#1} {#2}
1142   }
1143   \cs_new:Npn \__draw_point_interpolate_curve_auxviii:nnnnnn #1#2#3#4#5#6
1144   {
1145     \draw_point:n
1146     { #5 * #3 + #6 * #1 , #5 * #4 + #6 * #2 }
1147   }
1148   \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxviii:nnnnnn { eeee }

```

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

5.7 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
\l__draw_yvec_y_dim
\l__draw_zvec_x_dim
\l__draw_zvec_y_dim

```

(End of definition for \l__draw_xvec_x_dim and others.)

```

\draw_xvec:n Calculate the underlying position and store it.
\draw_yvec:n
\draw_zvec:n
\__draw_vec:nn
\__draw_vec:nnn

```

```

1155 \cs_new_protected:Npn \draw_xvec:n #1
1156 { \__draw_vec:nn { x } {#1} }
1157 \cs_new_protected:Npn \draw_yvec:n #1
1158 { \__draw_vec:nn { y } {#1} }
1159 \cs_new_protected:Npn \draw_zvec:n #1
1160 { \__draw_vec:nn { z } {#1} }
1161 \cs_new_protected:Npn \__draw_vec:nn #1#2
1162 {
1163   \__draw_point_process:nn { \__draw_vec:nnn {#1} } {#2}
1164 }
1165 \cs_new_protected:Npn \__draw_vec:nnn #1#2#3
1166 {
1167   \dim_set:cn { l__draw_ #1 vec_x_dim } {#2}
1168   \dim_set:cn { l__draw_ #1 vec_y_dim } {#3}
1169 }

```

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

Initialise the vectors.

```

1170 \draw_xvec:n { 1cm , 0cm }
1171 \draw_yvec:n { 0cm , 1cm }
1172 \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
\__draw_point_vec:ee
\draw_point_vec:nnn
\__draw_point_vec:nnn
\__draw_point_vec:eee

```

```

1173 \cs_new:Npn \draw_point_vec:nn #1#2
1174 { \__draw_point_vec:ee { \fp_eval:n {#1} } { \fp_eval:n {#2} } }

```

```

1175 \cs_new:Npn \__draw_point_vec:nn #1#2
1176 {
1177   \draw_point:n
1178   {
1179     #1 * \l__draw_xvec_x_dim + #2 * \l__draw_yvec_x_dim ,
1180     #1 * \l__draw_xvec_y_dim + #2 * \l__draw_yvec_y_dim
1181   }
1182 }
1183 \cs_generate_variant:Nn \__draw_point_vec:nn { ee }
1184 \cs_new:Npn \draw_point_vec:nnn #1#2#3
1185 {
1186   \__draw_point_vec:eee
1187   { \fp_eval:n {#1} } { \fp_eval:n {#2} } { \fp_eval:n {#3} }
1188 }
1189 \cs_new:Npn \__draw_point_vec:nnn #1#2#3
1190 {
1191   \draw_point:n
1192   {
1193     #1 * \l__draw_xvec_x_dim
1194     + #2 * \l__draw_yvec_x_dim
1195     + #3 * \l__draw_zvec_x_dim
1196     ,
1197     #1 * \l__draw_xvec_y_dim
1198     + #2 * \l__draw_yvec_y_dim
1199     + #3 * \l__draw_zvec_y_dim
1200   }
1201 }
1202 \cs_generate_variant:Nn \__draw_point_vec:nnn { eee }

```

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

```

\draw_point_vec_polar:nn
\draw_point_vec_polar:nnn
\__draw_point_vec_polar:nnn
\__draw_point_vec_polar:enn
\cs_new:Npn \draw_point_vec_polar:nn #1#2
1203 { \draw_point_vec_polar:nnn {#1} {#1} {#2} }
1204 \cs_new:Npn \draw_point_vec_polar:nnn #1#2#3
1205 { \__draw_draw_vec_polar:enn { \fp_eval:n {#3} } {#1} {#2} }
1206 \cs_new:Npn \__draw_draw_vec_polar:nnn #1#2#3
1207 {
1208   \draw_point:n
1209   {
1210     cosd(#1) * (#2) * \l__draw_xvec_x_dim ,
1211     sind(#1) * (#3) * \l__draw_yvec_y_dim
1212   }
1213 }
1214 \cs_generate_variant:Nn \__draw_draw_vec_polar:nnn { e }
1215

```

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

5.8 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.

```

1216 \cs_new:Npn \draw_point_transform:n #1

```

```

1217 {
1218   \__draw_point_process:nn
1219   { \__draw_point_transform:nn } {#1}
1220 }
1221 \cs_new:Npn \__draw_point_transform:nn #1#2
1222 {
1223   \bool_if:NTF \l__draw_matrix_active_bool
1224   {
1225     \draw_point:n
1226     {
1227       (
1228         \l__draw_matrix_a_fp * #1
1229         + \l__draw_matrix_c_fp * #2
1230         + \l__draw_xshift_dim
1231       )
1232       ,
1233       (
1234         \l__draw_matrix_b_fp * #1
1235         + \l__draw_matrix_d_fp * #2
1236         + \l__draw_yshift_dim
1237       )
1238     }
1239   }
1240   {
1241     \draw_point:n
1242     {
1243       (#1, #2)
1244       + ( \l__draw_xshift_dim , \l__draw_yshift_dim )
1245     }
1246   }
1247 }

```

(End of 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.

```

1248 \cs_new:Npn \__draw_point_transform_noshift:n #1
1249 {
1250   \__draw_point_process:nn
1251   { \__draw_point_transform_noshift:nn } {#1}
1252 }
1253 \cs_new:Npn \__draw_point_transform_noshift:nn #1#2
1254 {
1255   \bool_if:NTF \l__draw_matrix_active_bool
1256   {
1257     \draw_point:n
1258     {
1259       (
1260         \l__draw_matrix_a_fp * #1
1261         + \l__draw_matrix_c_fp * #2
1262       )
1263       ,
1264       (
1265         \l__draw_matrix_b_fp * #1

```

```

1266         + \l__draw_matrix_d_fp * #2
1267     )
1268 }
1269 }
1270 { \draw_point:n { (#1, #2) } }
1271 }

```

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

```

1272 \</package>

```

6 l3draw-scopes implementation

```

1273 \*package>
1274 \@@=draw>

```

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

- `\pgftext`: This is covered at this level by the coffin-based interface `\draw-coffin_use:Nnn`

6.1 Drawing environment

<pre> \g__draw_xmax_dim \g__draw_xmin_dim \g__draw_ymax_dim \g__draw_ymin_dim </pre>	<p>Used to track the overall (official) size of the image created: may not actually be the natural size of the content.</p> <pre> 1275 \dim_new:N \g__draw_xmax_dim 1276 \dim_new:N \g__draw_xmin_dim 1277 \dim_new:N \g__draw_ymax_dim 1278 \dim_new:N \g__draw_ymin_dim </pre> <p>(End of definition for <code>\g__draw_xmax_dim</code> and others.)</p>
<pre> \l_draw_bb_update_bool </pre>	<p>Flag to indicate that a path (or similar) should update the bounding box of the drawing.</p> <pre> 1279 \bool_new:N \l_draw_bb_update_bool </pre> <p>(End of definition for <code>\l_draw_bb_update_bool</code>. This variable is documented on page ??.)</p>
<pre> \l__draw_layer_main_box </pre>	<p>Box for setting the drawing itself and the top-level layer.</p> <pre> 1280 \box_new:N \l__draw_main_box 1281 \box_new:N \l__draw_layer_main_box </pre> <p>(End of definition for <code>\l__draw_layer_main_box</code>.)</p>
<pre> \g__draw_id_int </pre>	<p>The drawing number.</p> <pre> 1282 \int_new:N \g__draw_id_int </pre> <p>(End of definition for <code>\g__draw_id_int</code>.)</p>
<pre> __draw_reset_bb: </pre>	<p>A simple auxiliary.</p> <pre> 1283 \cs_new_protected:Npn __draw_reset_bb: 1284 { 1285 \dim_gset:Nn \g__draw_xmax_dim { -\c_max_dim } 1286 \dim_gset:Nn \g__draw_xmin_dim { \c_max_dim } 1287 \dim_gset:Nn \g__draw_ymax_dim { -\c_max_dim } 1288 \dim_gset:Nn \g__draw_ymin_dim { \c_max_dim } 1289 } </pre>

(End of definition for `_draw_reset_bb:`)

`\draw_begin:` Drawings are created by setting them into a box, then adjusting the box before inserting 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

```

1290 \cs_new_protected:Npn \draw_begin:
1291 {
1292   \group_begin:
1293   \int_gincr:N \g__draw_id_int
1294   \hbox_set:Nw \l__draw_main_box
1295     \__draw_backend_begin:
1296     \__draw_reset_bb:
1297     \__draw_path_reset_limits:
1298     \bool_set_true:N \l_draw_bb_update_bool
1299     \draw_transform_matrix_reset:
1300     \draw_transform_shift_reset:
1301     \__draw_softpath_clear:
1302     \draw_linewidth:n { \l_draw_default_linewidth_dim }
1303     \color_select:n { . }
1304     \draw_nonzero_rule:
1305     \draw_cap_but:
1306     \draw_join_miter:
1307     \draw_miterlimit:n { 10 }
1308     \draw_dash_pattern:nn { } { 0cm }
1309     \hbox_set:Nw \l__draw_layer_main_box
1310   }
1311 \cs_new_protected:Npn \draw_end:
1312 {
1313   \__draw_baseline_finalise:w
1314   \exp_args:NNNV \hbox_set_end:
1315   \clist_set:Nn \l_draw_layers_clist \l_draw_layers_clist
1316   \__draw_layers_insert:
1317   \__draw_backend_end:
1318   \hbox_set_end:
1319   \dim_compare:nNnT \g__draw_xmin_dim = \c_max_dim
1320   {
1321     \dim_gzero:N \g__draw_xmax_dim
1322     \dim_gzero:N \g__draw_xmin_dim
1323     \dim_gzero:N \g__draw_ymax_dim
1324     \dim_gzero:N \g__draw_ymin_dim
1325   }
1326   \__draw_finalise:
1327   \box_set_wd:Nn \l__draw_main_box
1328     { \g__draw_xmax_dim - \g__draw_xmin_dim }
1329   \mode_leave_vertical:
1330   \box_use_drop:N \l__draw_main_box
1331 \group_end:
1332 }
```

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

`__draw_finalise:` Finalising the (vertical) size of the output depends on whether we have an explicit baseline
`__draw_finalise_baseline:n` or not. To allow for that, we have two functions, and the one that's used depends on whether the user has set a baseline. Notice that in contrast to `pgf` we *do* allow for a non-zero depth if the explicit baseline is above the lowest edge of the initial bounding box.

```

1333 \cs_new_protected:Npn \__draw_finalise:
1334 {
1335   \hbox_set:Nn \l__draw_main_box
1336   {
1337     \skip_horizontal:n { -\g__draw_xmin_dim }
1338     \box_move_down:nn
1339     { \g__draw_ymin_dim }
1340     { \box_use_drop:N \l__draw_main_box }
1341   }
1342   \box_set_dp:Nn \l__draw_main_box { Opt }
1343   \box_set_ht:Nn \l__draw_main_box
1344   { \g__draw_ymax_dim - \g__draw_ymin_dim }
1345 }
1346 \cs_new_protected:Npn \__draw_finalise_baseline:n #1
1347 {
1348   \hbox_set:Nn \l__draw_main_box
1349   {
1350     \skip_horizontal:n { -\g__draw_xmin_dim }
1351     \box_move_down:nn
1352     { #1 }
1353     { \box_use_drop:N \l__draw_main_box }
1354   }
1355   \box_set_dp:Nn \l__draw_main_box
1356   {
1357     \dim_max:nn
1358     { #1 - \g__draw_ymin_dim }
1359     { Opt }
1360   }
1361   \box_set_ht:Nn \l__draw_main_box
1362   { \g__draw_ymax_dim - #1 }
1363 }

```

(End of definition for `__draw_finalise:` and `__draw_finalise_baseline:n`.)

6.2 Baseline position

`\l__draw_baseline_bool` For tracking the explicit baseline and whether it is active.

`\l__draw_baseline_dim`

```

1364 \bool_new:N \l__draw_baseline_bool
1365 \dim_new:N \l__draw_baseline_dim

```

(End of definition for `\l__draw_baseline_bool` and `\l__draw_baseline_dim`.)

`\draw_baseline:n` A simple setting of the baseline along with the flag we need to know that it is active.

```

1366 \cs_new_protected:Npn \draw_baseline:n #1
1367 {
1368   \bool_set_true:N \l__draw_baseline_bool
1369   \dim_set:Nn \l__draw_baseline_dim { \fp_to_dim:n {#1} }
1370 }

```

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

`__draw_baseline_finalise:w` Rather than use a global data structure, we can arrange to put the baseline value at the right group level with a small amount of shuffling. That happens here.

```

1371 \cs_new_protected:Npn \__draw_baseline_finalise:w #1 \__draw_finalise:
1372 {
1373   \bool_if:NTF \l__draw_baseline_bool
1374   {
1375     \use:e
1376     {
1377       \exp_not:n {#1}
1378       \__draw_finalise_baseline:n { \dim_use:N \l__draw_baseline_dim }
1379     }
1380   }
1381   { #1 \__draw_finalise: }
1382 }

```

(End of definition for `__draw_baseline_finalise:w`.)

6.3 Scopes

`\l__draw_linewidth_dim` Storage for local variables.
`\l__draw_fill_color_tl` 1383 `\dim_new:N \l__draw_linewidth_dim`
`\l__draw_stroke_color_tl` 1384 `\tl_new:N \l__draw_fill_color_tl`
1385 `\tl_new:N \l__draw_stroke_color_tl`

(End of 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 \TeX) scope, also deal with global data structures.

```

\draw_scope_begin: 1386 \cs_new_protected:Npn \draw_scope_begin:
1387 {
1388   \__draw_backend_scope_begin:
1389   \group_begin:
1390     \dim_set_eq:NN \l__draw_linewidth_dim \g__draw_linewidth_dim
1391     \draw_path_scope_begin:
1392   }
1393 \cs_new_protected:Npn \draw_scope_end:
1394 {
1395   \draw_path_scope_end:
1396   \dim_gset_eq:NN \g__draw_linewidth_dim \l__draw_linewidth_dim
1397   \group_end:
1398   \__draw_backend_scope_end:
1399 }

```

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

`\l__draw_xmax_dim` Storage for the bounding box.
`\l__draw_xmin_dim` 1400 `\dim_new:N \l__draw_xmax_dim`
`\l__draw_ymax_dim` 1401 `\dim_new:N \l__draw_xmin_dim`
`\l__draw_ymin_dim` 1402 `\dim_new:N \l__draw_ymax_dim`
1403 `\dim_new:N \l__draw_ymin_dim`

(End of 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:
1404 \cs_new_protected:Npn \__draw_scope_bb_begin:
1405 {
1406   \group_begin:
1407     \dim_set_eq:NN \l__draw_xmax_dim \g__draw_xmax_dim
1408     \dim_set_eq:NN \l__draw_xmin_dim \g__draw_xmin_dim
1409     \dim_set_eq:NN \l__draw_ymax_dim \g__draw_ymax_dim
1410     \dim_set_eq:NN \l__draw_ymin_dim \g__draw_ymin_dim
1411     \__draw_reset_bb:
1412   }
1413 \cs_new_protected:Npn \__draw_scope_bb_end:
1414 {
1415   \dim_gset_eq:NN \g__draw_xmax_dim \l__draw_xmax_dim
1416   \dim_gset_eq:NN \g__draw_xmin_dim \l__draw_xmin_dim
1417   \dim_gset_eq:NN \g__draw_ymax_dim \l__draw_ymax_dim
1418   \dim_gset_eq:NN \g__draw_ymin_dim \l__draw_ymin_dim
1419   \group_end:
1420 }
```

(End of definition for __draw_scope_bb_begin: and __draw_scope_bb_end:.)

`\draw_suspend_begin:` Suspend all parts of a drawing.

```

\draw_suspend_end:
1421 \cs_new_protected:Npn \draw_suspend_begin:
1422 {
1423   \__draw_scope_bb_begin:
1424   \draw_path_scope_begin:
1425   \draw_transform_matrix_reset:
1426   \draw_transform_shift_reset:
1427   \__draw_layers_save:
1428 }
1429 \cs_new_protected:Npn \draw_suspend_end:
1430 {
1431   \__draw_layers_restore:
1432   \draw_path_scope_end:
1433   \__draw_scope_bb_end:
1434 }
```

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

```

1435 </package>
```

7 l3draw-softpath implementation

```

1436 <*package>
```

```

1437 <@@=draw>
```

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.
1438 \tl_new:N \g__draw_softpath_main_tl
(End of definition for \g__draw_softpath_main_tl.)

\l__draw_softpath_tmp_tl Scratch space.
1439 \tl_new:N \l__draw_softpath_tmp_tl
(End of definition for \l__draw_softpath_tmp_tl.)

\g__draw_softpath_corners_bool Allow for optimised path use.
1440 \bool_new:N \g__draw_softpath_corners_bool
(End of definition for \g__draw_softpath_corners_bool.)

\__draw_softpath_add:n
\__draw_softpath_add:o 1441 \cs_new_protected:Npn \__draw_softpath_add:n
\__draw_softpath_add:e 1442 { \tl_build_gput_right:Nn \g__draw_softpath_main_tl }
1443 \cs_generate_variant:Nn \__draw_softpath_add:n { o, e }
(End of definition for \__draw_softpath_add:n.)

\__draw_softpath_use: Using and clearing is trivial.
\__draw_softpath_clear: 1444 \cs_new_protected:Npn \__draw_softpath_use:
1445 {
1446 \tl_build_get_intermediate:NN
1447 \g__draw_softpath_main_tl
1448 \l__draw_softpath_tmp_tl
1449 \l__draw_softpath_tmp_tl
1450 }
1451 \cs_new_protected:Npn \__draw_softpath_clear:
1452 {
1453 \tl_build_gbegin:N \g__draw_softpath_main_tl
1454 \bool_gset_false:N \g__draw_softpath_corners_bool
1455 }
(End of definition for \__draw_softpath_use: and \__draw_softpath_clear:.)

\__draw_softpath_save: Abstracted ideas to keep variables inside this submodule.
\__draw_softpath_restore: 1456 \cs_new_protected:Npn \__draw_softpath_save:
1457 {
1458 \tl_build_gend:N \g__draw_softpath_main_tl
1459 \tl_set_eq:NN
1460 \l__draw_softpath_main_tl
1461 \g__draw_softpath_main_tl
1462 \bool_set_eq:NN
1463 \l__draw_softpath_corners_bool
1464 \g__draw_softpath_corners_bool

```

```

1465     \__draw_softpath_clear:
1466   }
1467   \cs_new_protected:Npn \__draw_softpath_restore:
1468   {
1469     \__draw_softpath_clear:
1470     \__draw_softpath_add:o \l__draw_softpath_main_tl
1471     \bool_gset_eq:NN
1472       \g__draw_softpath_corners_bool
1473       \l__draw_softpath_corners_bool
1474   }

```

(End of definition for __draw_softpath_save: and __draw_softpath_restore:.)

\g__draw_softpath_lastx_dim For tracking the end of the path (to close it).

```

\g__draw_softpath_lasty_dim
1475 \dim_new:N \g__draw_softpath_lastx_dim
1476 \dim_new:N \g__draw_softpath_lasty_dim

```

(End of definition for \g__draw_softpath_lastx_dim and \g__draw_softpath_lasty_dim.)

\g__draw_softpath_move_bool Track if moving a point should update the close position.

```

1477 \bool_new:N \g__draw_softpath_move_bool
1478 \bool_gset_true:N \g__draw_softpath_move_bool

```

(End of definition for \g__draw_softpath_move_bool.)

__draw_softpath_curveto:nnnnnn The various parts of a path expressed as the appropriate soft path functions.

```

\__draw_softpath_lineto:nn
\__draw_softpath_moveto:nn
\__draw_softpath_rectangle:nnnn
\__draw_softpath_roundpoint:nn
\__draw_softpath_roundpoint:VV
1479 \cs_new_protected:Npn \__draw_softpath_closepath:
1480 {
1481   \__draw_softpath_add:e
1482   {
1483     \__draw_softpath_close_op:nn
1484     { \dim_use:N \g__draw_softpath_lastx_dim }
1485     { \dim_use:N \g__draw_softpath_lasty_dim }
1486   }
1487 }
1488 \cs_new_protected:Npn \__draw_softpath_curveto:nnnnnn #1#2#3#4#5#6
1489 {
1490   \__draw_softpath_add:n
1491   {
1492     \__draw_softpath_curveto_opi:nn {#1} {#2}
1493     \__draw_softpath_curveto_opii:nn {#3} {#4}
1494     \__draw_softpath_curveto_opiii:nn {#5} {#6}
1495   }
1496 }
1497 \cs_new_protected:Npn \__draw_softpath_lineto:nn #1#2
1498 {
1499   \__draw_softpath_add:n
1500   { \__draw_softpath_lineto_op:nn {#1} {#2} }
1501 }
1502 \cs_new_protected:Npn \__draw_softpath_moveto:nn #1#2
1503 {
1504   \__draw_softpath_add:n
1505   { \__draw_softpath_moveto_op:nn {#1} {#2} }
1506   \bool_if:NT \g__draw_softpath_move_bool
1507   {

```

```

1508         \dim_gset:Nn \g__draw_softpath_lastx_dim {#1}
1509         \dim_gset:Nn \g__draw_softpath_lasty_dim {#2}
1510     }
1511 }
1512 \cs_new_protected:Npn \__draw_softpath_rectangle:nnnn #1#2#3#4
1513 {
1514     \__draw_softpath_add:n
1515     {
1516         \__draw_softpath_rectangle_opi:nn {#1} {#2}
1517         \__draw_softpath_rectangle_opii:nn {#3} {#4}
1518     }
1519 }
1520 \cs_new_protected:Npn \__draw_softpath_roundpoint:nn #1#2
1521 {
1522     \__draw_softpath_add:n
1523     { \__draw_softpath_roundpoint_op:nn {#1} {#2} }
1524     \bool_gset_true:N \g__draw_softpath_corners_bool
1525 }
1526 \cs_generate_variant:Nn \__draw_softpath_roundpoint:nn { VV }

```

(End of 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.

```

1527 \cs_new_protected:Npn \__draw_softpath_close_op:nn #1#2
1528 { \__draw_backend_closepath: }
1529 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nn #1#2
1530 { \__draw_softpath_curveto_opi:nnNnnNnn {#1} {#2} }
1531 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nnNnnNnn #1#2#3#4#5#6#7#8
1532 { \__draw_backend_curveto:nnnnnn {#1} {#2} {#4} {#5} {#7} {#8} }
1533 \cs_new_protected:Npn \__draw_softpath_curveto_opii:nn #1#2
1534 { \__draw_softpath_curveto_opii:nn }
1535 \cs_new_protected:Npn \__draw_softpath_curveto_opiii:nn #1#2
1536 { \__draw_softpath_curveto_opiii:nn }
1537 \cs_new_protected:Npn \__draw_softpath_lineto_op:nn #1#2
1538 { \__draw_backend_lineto:nn {#1} {#2} }
1539 \cs_new_protected:Npn \__draw_softpath_moveto_op:nn #1#2
1540 { \__draw_backend_moveto:nn {#1} {#2} }
1541 \cs_new_protected:Npn \__draw_softpath_roundpoint_op:nn #1#2 { }
1542 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nn #1#2
1543 { \__draw_softpath_rectangle_opi:nnNnn {#1} {#2} }
1544 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nnNnn #1#2#3#4#5
1545 { \__draw_backend_rectangle:nnnn {#1} {#2} {#4} {#5} }
1546 \cs_new_protected:Npn \__draw_softpath_rectangle_opii:nn #1#2 { }

```

(End of 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. 1547 \tl_new:N \l__draw_softpath_main_tl <i>(End of definition for \l__draw_softpath_main_tl.)</i>
\l__draw_softpath_part_tl	Data structures. 1548 \tl_new:N \l__draw_softpath_part_tl 1549 \tl_new:N \l__draw_softpath_curve_end_tl <i>(End of definition for \l__draw_softpath_part_tl.)</i>
\l__draw_softpath_lastx_fp \l__draw_softpath_lasty_fp \l__draw_softpath_corneri_dim \l__draw_softpath_cornerii_dim \l__draw_softpath_first_tl \l__draw_softpath_move_tl	Position tracking: the token list data may be entirely empty or set to a co-ordinate. 1550 \fp_new:N \l__draw_softpath_lastx_fp 1551 \fp_new:N \l__draw_softpath_lasty_fp 1552 \dim_new:N \l__draw_softpath_corneri_dim 1553 \dim_new:N \l__draw_softpath_cornerii_dim 1554 \tl_new:N \l__draw_softpath_first_tl 1555 \tl_new:N \l__draw_softpath_move_tl <i>(End of definition for \l__draw_softpath_lastx_fp and others.)</i>
\c__draw_softpath_arc_fp	The magic constant. 1556 \fp_const:Nn \c__draw_softpath_arc_fp { 4/3 * (sqrt(2) - 1) } <i>(End of definition for \c__draw_softpath_arc_fp.)</i>
__draw_softpath_round_corners: __draw_softpath_round_loop:Nnn __draw_softpath_round_action:nn __draw_softpath_round_action:Nnn __draw_softpath_round_action_curveto:NnnNnn __draw_softpath_round_action_close: __draw_softpath_round_lookahead:NnnNnn __draw_softpath_round_roundpoint:NnnNnnNnn __draw_softpath_round_calc:NnnNnn __draw_softpath_round_calc:nnnnnn __draw_softpath_round_calc:eVnnnn __draw_softpath_round_calc:nnnnw __draw_softpath_round_close:nn __draw_softpath_round_close:w __draw_softpath_round_end:	Rounding corners on a path means going through the entire path and adjusting it. As such, we avoid this entirely if we know there are no corners to deal with. Assuming there is work to do, we recover the existing path and start a loop. 1557 \cs_new_protected:Npn __draw_softpath_round_corners: 1558 { 1559 \bool_if:NT \g__draw_softpath_corners_bool 1560 { 1561 \group_begin: 1562 \tl_clear:N \l__draw_softpath_main_tl 1563 \tl_clear:N \l__draw_softpath_part_tl 1564 \fp_zero:N \l__draw_softpath_lastx_fp 1565 \fp_zero:N \l__draw_softpath_lasty_fp 1566 \tl_clear:N \l__draw_softpath_first_tl 1567 \tl_clear:N \l__draw_softpath_move_tl 1568 \tl_build_gend:N \g__draw_softpath_main_tl 1569 \exp_after:wN __draw_softpath_round_loop:Nnn 1570 \g__draw_softpath_main_tl 1571 \q__draw_recursion_tail ? ? 1572 \q__draw_recursion_stop 1573 \group_end: 1574 } 1575 \bool_gset_false:N \g__draw_softpath_corners_bool 1576 }

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.

```

1577 \cs_new_protected:Npn \__draw_softpath_round_loop:Nnn #1#2#3
1578 {
1579   \__draw_if_recursion_tail_stop_do:Nn #1 { \__draw_softpath_round_end: }
1580   \token_if_eq_meaning:NNTF #1 \__draw_softpath_roundpoint_op:nn
1581   { \__draw_softpath_round_action:nn {#2} {#3} }
1582   {
1583     \tl_if_empty:NT \l__draw_softpath_first_tl
1584     { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1585     \fp_set:Nn \l__draw_softpath_lastx_fp {#2}
1586     \fp_set:Nn \l__draw_softpath_lasty_fp {#3}
1587     \token_if_eq_meaning:NNTF #1 \__draw_softpath_moveto_op:nn
1588     {
1589       \tl_put_right:No \l__draw_softpath_main_tl
1590       \l__draw_softpath_move_tl
1591       \tl_put_right:No \l__draw_softpath_main_tl
1592       \l__draw_softpath_part_tl
1593       \tl_set:Nn \l__draw_softpath_move_tl { #1 {#2} {#3} }
1594       \tl_clear:N \l__draw_softpath_first_tl
1595       \tl_clear:N \l__draw_softpath_part_tl
1596     }
1597     { \tl_put_right:Nn \l__draw_softpath_part_tl { #1 {#2} {#3} } }
1598     \__draw_softpath_round_loop:Nnn
1599   }
1600 }
1601 \cs_new_protected:Npn \__draw_softpath_round_action:nn #1#2
1602 {
1603   \dim_set:Nn \l__draw_softpath_corneri_dim {#1}
1604   \dim_set:Nn \l__draw_softpath_cornerii_dim {#2}
1605   \bool_lazy_and:nnTF
1606   { \dim_compare_p:nNn \l__draw_softpath_corneri_dim = { Opt } }
1607   { \dim_compare_p:nNn \l__draw_softpath_cornerii_dim = { Opt } }
1608   { \__draw_softpath_round_loop:Nnn }
1609   { \__draw_softpath_round_action:Nnn }
1610 }

```

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.

```

1611 \cs_new_protected:Npn \__draw_softpath_round_action:Nnn #1#2#3
1612 {
1613   \tl_if_empty:NT \l__draw_softpath_first_tl
1614   { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1615   \token_if_eq_meaning:NNTF #1 \__draw_softpath_curveto_opi:nn
1616   { \__draw_softpath_round_action_curveto:NnnNnn }
1617   {
1618     \token_if_eq_meaning:NNTF #1 \__draw_softpath_close_op:nn
1619     { \__draw_softpath_round_action_close: }
1620     {
1621       \token_if_eq_meaning:NNTF #1 \__draw_softpath_lineto_op:nn

```

```

1622         { \__draw_softpath_round_lookahead:NnnNnn }
1623         { \__draw_softpath_round_loop:Nnn }
1624     }
1625 }
1626 #1 {#2} {#3}
1627 }

```

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.

```

1628 \cs_new_protected:Npn \__draw_softpath_round_action_curveto:NnnNnn
1629 #1#2#3#4#5#6
1630 {
1631   \tl_put_right:Nn \l__draw_softpath_part_tl
1632   { #1 {#2} {#3} #4 {#5} {#6} }
1633   \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1634   \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1635   \__draw_softpath_round_lookahead:NnnNnn
1636 }
1637 \cs_new_protected:Npn \__draw_softpath_round_action_close:
1638 {
1639   \bool_lazy_and:nnTF
1640   { ! \tl_if_empty_p:N \l__draw_softpath_first_tl }
1641   { ! \tl_if_empty_p:N \l__draw_softpath_move_tl }
1642   {
1643     \exp_after:wN \__draw_softpath_round_close:nn
1644     \l__draw_softpath_first_tl
1645   }
1646   { \__draw_softpath_round_loop:Nnn }
1647 }

```

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.

```

1648 \cs_new_protected:Npn \__draw_softpath_round_lookahead:NnnNnn #1#2#3#4#5#6
1649 {
1650   \bool_lazy_any:nTF
1651   {
1652     { \token_if_eq_meaning_p:NN #4 \__draw_softpath_lineto_op:nn }
1653     { \token_if_eq_meaning_p:NN #4 \__draw_softpath_curveto_opi:nn }
1654     { \token_if_eq_meaning_p:NN #4 \__draw_softpath_close_op:nn }
1655   }
1656   {
1657     \__draw_softpath_round_calc:NnnNnn
1658     \__draw_softpath_round_loop:Nnn
1659     {#5} {#6}
1660   }
1661   {
1662     \token_if_eq_meaning:NNTF #4 \__draw_softpath_roundpoint_op:nn
1663     { \__draw_softpath_round_roundpoint:NnnNnnNnn }
1664     { \__draw_softpath_round_loop:Nnn }
1665   }
1666   #1 {#2} {#3}
1667   #4 {#5} {#6}
1668 }
1669 \cs_new_protected:Npn \__draw_softpath_round_roundpoint:NnnNnnNnn

```

```

1670 #1#2#3#4#5#6#7#8#9
1671 {
1672   \__draw_softpath_round_calc:NnnNnn
1673   \__draw_softpath_round_loop:Nnn
1674   {#8} {#9}
1675   #1 {#2} {#3}
1676   #4 {#5} {#6} #7 {#8} {#9}
1677 }

```

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:Ne`. The end point of the line is worked out up-front and saved: we need that if dealing with a close-path operation.

```

1678 \cs_new_protected:Npn \__draw_softpath_round_calc:NnnNnn #1#2#3#4#5#6
1679 {
1680   \tl_set:Nx \l__draw_softpath_curve_end_tl
1681   {
1682     \draw_point_interpolate_distance:nnn
1683     \l__draw_softpath_cornerii_dim
1684     { #5 , #6 } { #2 , #3 }
1685   }
1686   \tl_put_right:Nx \l__draw_softpath_part_tl
1687   {
1688     \exp_not:N #4
1689     \__draw_softpath_round_calc:eVnnnn
1690     {
1691       \draw_point_interpolate_distance:nnn
1692       \l__draw_softpath_corneri_dim
1693       { #5 , #6 }
1694       {
1695         \l__draw_softpath_lastx_fp ,
1696         \l__draw_softpath_lasty_fp
1697       }
1698     }
1699     \l__draw_softpath_curve_end_tl
1700     {#5} {#6} {#2} {#3}
1701   }
1702   \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1703   \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1704   #1
1705 }

```

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).

```

1706 \cs_new:Npn \__draw_softpath_round_calc:nnnnnn #1#2#3#4#5#6
1707 {
1708   \__draw_softpath_round_calc:nnnnw {#3} {#4} {#5} {#6}
1709   #1 \s__draw_mark #2 \s__draw_stop
1710 }
1711 \cs_generate_variant:Nn \__draw_softpath_round_calc:nnnnnn { eV }

```

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

```

1712 \cs_new:Npn \__draw_softpath_round_calc:nnnnw
1713 #1#2#3#4 #5 , #6 \s__draw_mark #7 , #8 \s__draw_stop
1714 {
1715   {#5} {#6}
1716   \exp_not:N \__draw_softpath_curveto_opi:nn
1717   {
1718     \fp_to_dim:n
1719     { #5 + \c__draw_softpath_arc_fp * ( #1 - #5 ) }
1720   }
1721   {
1722     \fp_to_dim:n
1723     { #6 + \c__draw_softpath_arc_fp * ( #2 - #6 ) }
1724   }
1725   \exp_not:N \__draw_softpath_curveto_opii:nn
1726   {
1727     \fp_to_dim:n
1728     { #7 + \c__draw_softpath_arc_fp * ( #1 - #7 ) }
1729   }
1730   {
1731     \fp_to_dim:n
1732     { #8 + \c__draw_softpath_arc_fp * ( #2 - #8 ) }
1733   }
1734   \exp_not:N \__draw_softpath_curveto_opiii:nn
1735   {#7} {#8}
1736 }

```

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.

```

1737 \cs_new_protected:Npn \__draw_softpath_round_close:nn #1#2
1738 {
1739   \use:e
1740   {
1741     \__draw_softpath_round_calc:NnnNnn
1742     {
1743       \tl_set:Nc \exp_not:N \l__draw_softpath_move_tl
1744       {
1745         \__draw_softpath_moveto_op:nn
1746         \exp_not:N \exp_after:wN
1747         \exp_not:N \__draw_softpath_round_close:w
1748         \exp_not:N \l__draw_softpath_curve_end_tl
1749         \s__draw_stop
1750       }
1751     }
1752     \use:e
1753     {
1754       \exp_not:N \exp_not:N \exp_not:N \use_i:nnnn
1755       {
1756         \__draw_softpath_round_loop:Nnn
1757         \__draw_softpath_close_op:nn
1758         \exp_not:N \exp_after:wN
1759         \exp_not:N \__draw_softpath_round_close:w

```



```

1759             \exp_not:N \l__draw_softpath_curve_end_tl
1760             \s__draw_stop
1761         }
1762     }
1763 }
1764 {#1} {#2}
1765 \__draw_softpath_lineto_op:nn
1766 \exp_after:wN \use_none:n \l__draw_softpath_move_tl
1767 }
1768 }
1769 \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.

```

1770 \cs_new_protected:Npn \__draw_softpath_round_end:
1771 {
1772     \tl_put_right:No \l__draw_softpath_main_tl
1773     \l__draw_softpath_move_tl
1774     \tl_put_right:No \l__draw_softpath_main_tl
1775     \l__draw_softpath_part_tl
1776     \tl_build_gbegin:N \g__draw_softpath_main_tl
1777     \__draw_softpath_add:o \l__draw_softpath_main_tl
1778 }

```

(End of definition for __draw_softpath_round_corners: and others.)

```

1779 </package>

```

8 l3draw-state implementation

```

1780 <*package>
1781 <@@=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.

```

1782 \dim_new:N \g__draw_linewidth_dim

```

(End of definition for \g__draw_linewidth_dim.)

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

```

1783 \dim_new:N \l_draw_default_linewidth_dim
1784 \dim_set:Nn \l_draw_default_linewidth_dim { 0.4pt }

```

(End of 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.

```

1785 \cs_new_protected:Npn \draw_linewidth:n #1
1786 {
1787     \dim_gset:Nn \g__draw_linewidth_dim { \fp_to_dim:n {#1} }
1788     \__draw_backend_linewidth:n \g__draw_linewidth_dim
1789 }

```

(End of 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 1790 \cs_new_protected:Npn \draw_dash_pattern:nn #1#2
                    1791 {
                    1792   \group_begin:
                    1793     \seq_set_from_clist:Nn \l__draw_tmp_seq {#1}
                    1794     \seq_set_map:Nn \l__draw_tmp_seq \l__draw_tmp_seq
                    1795       { \fp_to_dim:n {##1} }
                    1796     \use:e
                    1797     {
                    1798       \__draw_backend_dash_pattern:nn
                    1799       { \seq_use:Nn \l__draw_tmp_seq { , } }
                    1800       { \fp_to_dim:n {#2} }
                    1801     }
                    1802   \group_end:
                    1803 }
                    1804 \seq_new:N \l__draw_tmp_seq

```

(End of 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.
                    1805 \cs_new_protected:Npn \draw_miterlimit:n #1
                    1806   { \exp_args:Ne \__draw_backend_miterlimit:n { \fp_eval:n {#1} } }

```

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

```

\draw_cap_but: All straight wrappers.
\draw_cap_rectangle: 1807 \cs_new_protected:Npn \draw_cap_but: { \__draw_backend_cap_but: }
\draw_cap_round: 1808 \cs_new_protected:Npn \draw_cap_rectangle: { \__draw_backend_cap_rectangle: }
\draw_evenodd_rule: 1809 \cs_new_protected:Npn \draw_cap_round: { \__draw_backend_cap_round: }
\draw_nonzero_rule: 1810 \cs_new_protected:Npn \draw_evenodd_rule: { \__draw_backend_evenodd_rule: }
\draw_join_bevel: 1811 \cs_new_protected:Npn \draw_nonzero_rule: { \__draw_backend_nonzero_rule: }
\draw_join_miter: 1812 \cs_new_protected:Npn \draw_join_bevel: { \__draw_backend_join_bevel: }
\draw_join_round: 1813 \cs_new_protected:Npn \draw_join_miter: { \__draw_backend_join_miter: }
                  1814 \cs_new_protected:Npn \draw_join_round: { \__draw_backend_join_round: }

```

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

```
1815 \</package>
```

9 l3draw-transforms implementation

```
1816 \*package>
```

```
1817 \<@@=draw>
```

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.
- `\pgftransformlineattime`, `\pgftransformarcaxesattime`, `\pgftransformcurveattime`: Need to look at the use cases for these to fully understand them.
- `\pgftransformarrow`: Likely to be done when other arrow functions are added.

- `\pgftransformationadjustments`: Used mainly by CircuiTikZ although also for shapes, likely needs more use cases before addressing.
- `\pgflevelsynchronism`, `\pgflevel`: Likely to be added when use cases are encountered in other parts of the code.
- `\pgfviewboxscope`: Seems very speicalied, need to understand the requirements here.

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

```
1818 \bool_new:N \l__draw_matrix_active_bool
```

(End of definition for \l__draw_matrix_active_bool.)

`\l__draw_matrix_a_fp` The active matrix and shifts.

```
\l__draw_matrix_b_fp 1819 \fp_new:N \l__draw_matrix_a_fp
\l__draw_matrix_c_fp 1820 \fp_new:N \l__draw_matrix_b_fp
\l__draw_xshift_dim 1821 \fp_new:N \l__draw_matrix_c_fp
\l__draw_yshift_dim 1822 \fp_new:N \l__draw_matrix_d_fp
1823 \dim_new:N \l__draw_xshift_dim
1824 \dim_new:N \l__draw_yshift_dim
```

(End of definition for \l__draw_matrix_a_fp and others.)

`\draw_transform_matrix_reset:` Fast resetting.

```
\draw_transform_shift_reset: 1825 \cs_new_protected:Npn \draw_transform_matrix_reset:
1826 {
1827   \fp_set:Nn \l__draw_matrix_a_fp { 1 }
1828   \fp_zero:N \l__draw_matrix_b_fp
1829   \fp_zero:N \l__draw_matrix_c_fp
1830   \fp_set:Nn \l__draw_matrix_d_fp { 1 }
1831 }
1832 \cs_new_protected:Npn \draw_transform_shift_reset:
1833 {
1834   \dim_zero:N \l__draw_xshift_dim
1835   \dim_zero:N \l__draw_yshift_dim
1836 }
1837 \draw_transform_matrix_reset:
1838 \draw_transform_shift_reset:
```

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

`\draw_transform_matrix_absolute:nmmn` 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.

```
\_draw_transform_shift_absolute:nn 1839 \cs_new_protected:Npn \draw_transform_matrix_absolute:nmmn #1#2#3#4
1840 {
1841   \fp_set:Nn \l__draw_matrix_a_fp {#1}
1842   \fp_set:Nn \l__draw_matrix_b_fp {#2}
1843   \fp_set:Nn \l__draw_matrix_c_fp {#3}
1844   \fp_set:Nn \l__draw_matrix_d_fp {#4}
1845   \bool_lazy_all:nTF
1846   {
1847     { \fp_compare_p:nNn \l__draw_matrix_a_fp = \c_one_fp }
1848     { \fp_compare_p:nNn \l__draw_matrix_b_fp = \c_zero_fp }

```

```

1849         { \fp_compare_p:nNn \l__draw_matrix_c_fp = \c_zero_fp }
1850         { \fp_compare_p:nNn \l__draw_matrix_d_fp = \c_one_fp }
1851     }
1852     { \bool_set_false:N \l__draw_matrix_active_bool }
1853     { \bool_set_true:N \l__draw_matrix_active_bool }
1854 }
1855 \cs_new_protected:Npn \draw_transform_shift_absolute:n #1
1856 {
1857     \__draw_point_process:nn
1858     { \__draw_transform_shift_absolute:nn } {#1}
1859 }
1860 \cs_new_protected:Npn \__draw_transform_shift_absolute:nn #1#2
1861 { \__draw_transform_shift:nnnn { Opt } { Opt } {#1} {#2} }

```

(End of 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 that the calculation uses “frozen” values.

```

\__draw_transform:nnnn
\draw_transform_shift:n
\__draw_transform_shift:nn
1862 \cs_new_protected:Npn \draw_transform_matrix:nnnn #1#2#3#4
1863 {
1864     \use:e
1865     {
1866         \__draw_transform:nnnn
1867         { \fp_eval:n {#1} }
1868         { \fp_eval:n {#2} }
1869         { \fp_eval:n {#3} }
1870         { \fp_eval:n {#4} }
1871     }
1872 }
1873 \cs_new_protected:Npn \__draw_transform:nnnn #1#2#3#4
1874 {
1875     \use:e
1876     {
1877         \draw_transform_matrix_absolute:nnnn
1878         { #1 * \l__draw_matrix_a_fp + #2 * \l__draw_matrix_c_fp }
1879         { #1 * \l__draw_matrix_b_fp + #2 * \l__draw_matrix_d_fp }
1880         { #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_c_fp }
1881         { #3 * \l__draw_matrix_b_fp + #4 * \l__draw_matrix_d_fp }
1882     }
1883 }
1884 \cs_new_protected:Npn \draw_transform_shift:n #1
1885 {
1886     \__draw_point_process:nn
1887     { \__draw_transform_shift:nn } {#1}
1888 }
1889 \cs_new_protected:Npn \__draw_transform_shift:nn #1#2
1890 {
1891     \__draw_transform_shift:nnnn
1892     \l__draw_xshift_dim
1893     \l__draw_yshift_dim
1894     {#1} {#2}
1895 }

```

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

`__draw_transform_shift:nnnn` Apply the current transformation matrix to the shift, then store the resulting values: we may or may not have a none-zero starting point here.

```

1896 \cs_new_protected:Npn \__draw_transform_shift:nnnn #1#2#3#4
1897 {
1898   \dim_set:Nn \l__draw_xshift_dim
1899   {
1900     \fp_to_dim:n
1901     {
1902       #1 +
1903       ( #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_b_fp )
1904     }
1905   }
1906   \dim_set:Nn \l__draw_yshift_dim
1907   {
1908     \fp_to_dim:n
1909     {
1910       #2 +
1911       ( #3 * \l__draw_matrix_c_fp + #4 * \l__draw_matrix_d_fp )
1912     }
1913   }
1914 }

```

(End of definition for `__draw_transform_shift:nnnn`.)

`\draw_transform_matrix_invert:` Standard mathematics: calculate the inverse matrix and use that, then undo the shifts.

```

\__draw_transform_invert:n 1915 \cs_new_protected:Npn \draw_transform_matrix_invert:
\__draw_transform_invert:e 1916 {
\draw_transform_shift_invert: 1917   \bool_if:NT \l__draw_matrix_active_bool
1918   {
1919     \__draw_transform_invert:e
1920     {
1921       \fp_eval:n
1922       {
1923         1 /
1924         (
1925           \l__draw_matrix_a_fp * \l__draw_matrix_d_fp
1926           - \l__draw_matrix_b_fp * \l__draw_matrix_c_fp
1927         )
1928       }
1929     }
1930   }
1931 }
1932 \cs_new_protected:Npn \__draw_transform_invert:n #1
1933 {
1934   \fp_set:Nn \l__draw_matrix_a_fp
1935   { \l__draw_matrix_d_fp * #1 }
1936   \fp_set:Nn \l__draw_matrix_b_fp
1937   { -\l__draw_matrix_b_fp * #1 }
1938   \fp_set:Nn \l__draw_matrix_c_fp
1939   { -\l__draw_matrix_c_fp * #1 }
1940   \fp_set:Nn \l__draw_matrix_d_fp

```

```

1941     { \l__draw_matrix_a_fp * #1 }
1942   }
1943   \cs_generate_variant:Nn \__draw_transform_invert:n { e }
1944   \cs_new_protected:Npn \draw_transform_shift_invert:
1945   {
1946     \dim_set:Nn \l__draw_xshift_dim { -\l__draw_xshift_dim }
1947     \dim_set:Nn \l__draw_yshift_dim { -\l__draw_yshift_dim }
1948   }

```

(End of 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.

```

1949   \cs_new_protected:Npn \draw_transform_triangle:nnn #1#2#3
1950   {
1951     \__draw_point_process:nnn
1952     {
1953       \__draw_point_process:nn
1954       { \__draw_transform_triangle:nnnnnn }
1955       {#1}
1956     }
1957     {#2} {#3}
1958   }
1959   \cs_new_protected:Npn \__draw_transform_triangle:nnnnnn #1#2#3#4#5#6
1960   {
1961     \use:e
1962     {
1963       \draw_transform_matrix_absolute:nnnn
1964       { #3 - #1 }
1965       { #4 - #2 }
1966       { #5 - #1 }
1967       { #6 - #2 }
1968       \draw_transform_shift_absolute:n { #1 , #2 }
1969     }
1970   }

```

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

\draw_transform_scale:n Lots of shortcuts.

```

\draw_transform_xscale:n 1971 \cs_new_protected:Npn \draw_transform_scale:n #1
\draw_transform_yscale:n 1972 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { #1 } }
\draw_transform_xshift:n 1973 \cs_new_protected:Npn \draw_transform_xscale:n #1
\draw_transform_yshift:n 1974 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { #1 } }
\draw_transform_xslant:n 1975 \cs_new_protected:Npn \draw_transform_yscale:n #1
\draw_transform_yslant:n 1976 { \draw_transform_matrix:nnnn { 1 } { 0 } { 0 } { #1 } }
1977 \cs_new_protected:Npn \draw_transform_xshift:n #1
1978 { \draw_transform_shift:n { #1 , Opt } }
1979 \cs_new_protected:Npn \draw_transform_yshift:n #1
1980 { \draw_transform_shift:n { Opt , #1 } }
1981 \cs_new_protected:Npn \draw_transform_xslant:n #1
1982 { \draw_transform_matrix:nnnn { 1 } { 0 } { #1 } { #1 } }
1983 \cs_new_protected:Npn \draw_transform_yslant:n #1
1984 { \draw_transform_matrix:nnnn { 1 } { #1 } { 0 } { #1 } }

```

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

<code>\draw_transform_rotate:n</code>	Slightly more involved: evaluate the angle only once, and the sine and cosine only once.
<code>__draw_transform_rotate:n</code>	1985 <code>\cs_new_protected:Npn \draw_transform_rotate:n #1</code>
<code>__draw_transform_rotate:e</code>	1986 <code>{ __draw_transform_rotate:e { \fp_eval:n {#1} } }</code>
<code>__draw_transform_rotate:nn</code>	1987 <code>\cs_new_protected:Npn __draw_transform_rotate:n #1</code>
<code>__draw_transform_rotate:ee</code>	1988 <code>{</code>
	1989 <code>__draw_transform_rotate:ee</code>
	1990 <code>{ \fp_eval:n { cosd(#1) } }</code>
	1991 <code>{ \fp_eval:n { sind(#1) } }</code>
	1992 <code>}</code>
	1993 <code>\cs_generate_variant:Nn __draw_transform_rotate:n { e }</code>
	1994 <code>\cs_new_protected:Npn __draw_transform_rotate:nn #1#2</code>
	1995 <code>{ \draw_transform_matrix:nnnn {#1} {#2} { -#2 } { #1 } }</code>
	1996 <code>\cs_generate_variant:Nn __draw_transform_rotate:nn { ee }</code>
 (End of definition for <code>\draw_transform_rotate:n</code> , <code>__draw_transform_rotate:n</code> , and <code>__draw_transform_rotate:nn</code> . This function is documented on page ??.)	
	1997 <code>\</package></code>

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` ... 193, 802, 1100, 1104, 1128, 1131

bool commands:

`\bool_gset_eq:Nn` 1471

`\bool_gset_false:N` 1454, 1575

`\bool_gset_true:N` 1478, 1524

`\bool_if:NTF`
 . 33, 137, 216, 255, 706, 722, 723,
 727, 1223, 1255, 1373, 1506, 1559, 1917

`\bool_lazy_all:nTF` 1845

`\bool_lazy_and:nnTF`
 247, 701, 1605, 1639

`\bool_lazy_any:nTF` 1650

`\bool_lazy_or:nnTF` . 582, 677, 710, 715

`\bool_new:N`
 108, 242, 665, 666, 667, 668,
 669, 769, 1279, 1364, 1440, 1477, 1818

`\bool_set_eq:Nn` 1462

`\bool_set_false:N`
 ... 118, 250, 688, 689, 690, 709, 1852

`\bool_set_true:N` 120, 251,
 694, 732, 736, 737, 1298, 1368, 1853

box commands:

`\box_dp:N` 17, 22, 89

`\box_gset_eq:Nn` 178

`\box_gset_wd:Nn` 122, 155

`\box_ht:N` 17, 22, 91

`\box_if_exist:NTF` 115

`\box_move_down:nn` 1338, 1351

`\box_move_up:nn` 63

`\box_new:N` ... 13, 102, 103, 1280, 1281

`\box_set_dp:Nn` 67, 1342, 1355

`\box_set_eq:Nn` 167

`\box_set_ht:Nn` 66, 1343, 1361

`\box_set_wd:Nn` 68, 150, 1327

`\box_use_drop:N` 64,
 69, 124, 151, 156, 1330, 1340, 1353

`\box_wd:N` 17, 22, 88, 90

C

clist commands:

`\clist_map_inline:Nn` .. 146, 163, 174

`\clist_map_inline:nn` 691

`\clist_new:N` 109, 111

`\clist_set:Nn` 110, 1315

coffin commands:

`\coffin_typeset:Nnnnn` 86

`\coffin_wd:N` 88

color commands:

`\color_select:n` 1303

cs commands:

`\cs_generate_variant:Nn` . 442, 631,
 664, 814, 826, 839, 857, 865, 873,
 891, 916, 935, 942, 950, 957, 966,
 972, 992, 1000, 1007, 1013, 1025,
 1028, 1046, 1064, 1072, 1078, 1099,
 1113, 1127, 1148, 1183, 1202, 1215,
 1443, 1526, 1711, 1943, 1993, 1996

`\cs_if_exist:NTF` 693

`\cs_if_exist_use:NTF` .. 425, 434, 696

`\cs_new:Npn` 535, 545, 555, 565, 806,
 812, 815, 817, 824, 827, 829, 837,
 840, 843, 852, 858, 861, 863, 866,
 867, 869, 871, 874, 876, 882, 892,
 898, 908, 917, 923, 928, 936, 943,
 951, 958, 967, 973, 979, 984, 993,
 1001, 1008, 1014, 1020, 1026, 1029,
 1035, 1044, 1047, 1053, 1058, 1065,
 1073, 1079, 1085, 1091, 1105, 1114,
 1120, 1132, 1134, 1143, 1173, 1175,
 1184, 1189, 1203, 1205, 1207, 1216,
 1221, 1248, 1253, 1706, 1712, 1769

`\cs_new_protected:Npn` 14, 19, 24, 31,
 72, 77, 82, 97, 112, 135, 144, 161,
 172, 206, 228, 235, 243, 253, 262,
 268, 274, 280, 287, 298, 306, 311,
 313, 315, 324, 331, 367, 369, 380,
 386, 416, 443, 472, 478, 484, 489,
 497, 506, 513, 521, 576, 578, 591,
 598, 607, 613, 615, 625, 632, 638,
 645, 670, 675, 686, 730, 734, 739,
 746, 770, 784, 1155, 1157, 1159,
 1161, 1165, 1283, 1290, 1311, 1333,
 1346, 1366, 1371, 1386, 1393, 1404,
 1413, 1421, 1429, 1441, 1444, 1451,
 1456, 1467, 1479, 1488, 1497, 1502,
 1512, 1520, 1527, 1529, 1531, 1533,
 1535, 1537, 1539, 1541, 1542, 1544,
 1546, 1557, 1577, 1601, 1611, 1628,
 1637, 1648, 1669, 1678, 1737, 1770,
 1785, 1790, 1805, 1807, 1808, 1809,
 1810, 1811, 1812, 1813, 1814, 1825,
 1832, 1839, 1855, 1860, 1862, 1873,
 1884, 1889, 1896, 1915, 1932, 1944,
 1949, 1959, 1971, 1973, 1975, 1977,
 1979, 1981, 1983, 1985, 1987, 1994

D

dim commands:

`\dim_abs:n` 620, 621
`\dim_compare:nNnTF` 627, 634, 748, 1319
`\dim_compare_p:nNn` 248, 249, 1606, 1607
`\dim_eval:n` 620, 621
`\dim_gset:Nn` 208, 210,
212, 214, 218, 220, 222, 224, 230,
231, 232, 233, 237, 238, 750, 1285,
1286, 1287, 1288, 1508, 1509, 1787
`\dim_gset_eq:NN`
.... 787, 788, 789, 790, 791, 792,
793, 794, 1396, 1415, 1416, 1417, 1418
`\dim_gzero:N` .. 1321, 1322, 1323, 1324
`\dim_max:nn` .. 209, 213, 219, 223, 1357
`\dim_min:nn` 211, 215, 221, 225
`\dim_new:N` 200, 201,
202, 203, 204, 205, 240, 241, 761,
762, 763, 764, 765, 766, 767, 768,
1149, 1150, 1151, 1152, 1153, 1154,
1275, 1276, 1277, 1278, 1365, 1383,
1400, 1401, 1402, 1403, 1475, 1476,
1552, 1553, 1782, 1783, 1823, 1824
`\dim_set:Nn`
. 245, 246, 1167, 1168, 1369, 1603,
1604, 1784, 1898, 1906, 1946, 1947
`\dim_set_eq:NN`
.... 773, 774, 775, 776, 777, 778,
779, 780, 1390, 1407, 1408, 1409, 1410
`\dim_step_inline:nnnn` 647, 655
`\dim_use:N`
.... 748, 753, 755, 1378, 1484, 1485
`\dim_zero:N` 1834, 1835
`\c_max_dim` 230, 231, 232,
233, 748, 1285, 1286, 1287, 1288, 1319

draw commands:

`\draw_baseline:n` 1366, 1366
`\l_draw_bb_update_bool`
.... 33, 216, 702, 709, 1279, 1298
`\draw_begin:` 1290, 1290
`\draw_box_use:N` 14, 14
`\draw_box_use:Nn` 14, 19
`\draw_cap_but:` 1305, 1807, 1807
`\draw_cap_rectangle:` 1807, 1808
`\draw_cap_round:` 1807, 1809
`\draw_coffin_use:Nnn` 36, 72, 72
`\draw_coffin_use:Nnnn` 72, 77
`\draw_dash_pattern:nn` 1308, 1790, 1790
`\l_draw_default_linewidth_dim` ...
.... 127, 1302, 1783
`\draw_end:` 1290, 1311
`\draw_evenodd_rule:` 1807, 1810
`\draw_join_bevel:` 1807, 1812
`\draw_join_miter:` ... 1306, 1807, 1813

`\draw_join_round:` 1807, 1814
`\draw_layer_begin:n` 112, 112
`\draw_layer_end:` 112, 135
`\draw_layer_new:n` 97, 97
`\l_draw_layers_clist`
.... 109, 146, 163, 174, 1315
`\draw_linewidth:n` 127, 1302, 1785, 1785
`\draw_miterlimit:n` .. 1307, 1805, 1805
`\draw_nonzero_rule:` . 1304, 1807, 1811
`\draw_path_arc:nnn` 367, 367, 510
`\draw_path_arc:nnnn` ... 367, 368, 369
`\draw_path_arc_axes:nnnn` ... 506, 506
`\draw_path_canvas_curveto:nnn` ...
.... 311, 315
`\draw_path_canvas_lineto:n` . 311, 313
`\draw_path_canvas_moveto:n` . 311, 311
`\draw_path_circle:nn` 576, 576
`\draw_path_close:` 306, 306, 604
`\draw_path_corner_arc:nn` ... 243, 243
`\draw_path_curveto:nn` 324, 324
`\draw_path_curveto:nnn` 262, 287
`\draw_path_ellipse:nnn` . 513, 513, 577
`\draw_path_grid:nnnn` 615, 615
`\draw_path_lineto:n`
.... 262, 274, 601, 602, 603, 653, 661
`\draw_path_moveto:n`
.... 262, 262, 600, 605, 652, 660
`\draw_path_rectangle:nn` 578, 578, 614
`\draw_path_rectangle_corners:nn` .
.... 607, 607
`\draw_path_scope_begin:`
.... 770, 770, 1391, 1424
`\draw_path_scope_end:`
.... 770, 784, 1395, 1432
`\draw_path_use:n` 670, 670
`\draw_path_use_clear:n` 670, 675
`\draw_point:n` 809, 820,
821, 832, 833, 834, 846, 847, 848,
849, 861, 861, 872, 887, 910, 969,
1010, 1027, 1045, 1075, 1145, 1177,
1191, 1209, 1225, 1241, 1257, 1270
`\draw_point_interpolate_arcaxes:nnnnnn`
.... 1047, 1047
`\draw_point_interpolate_curve:nnnnn`
.... 1079
`\draw_point_interpolate_curve:nnnnnn`
.... 1079
`\draw_point_interpolate_curve_-`
auxi:nnnnnnnn 1079
`\draw_point_interpolate_curve_-`
auxii:nnnnnnnn 1079
`\draw_point_interpolate_curve_-`
auxiii:nnnnnn 1079

<code>\draw_point_interpolate_curve-</code> <code>auxiv:nnnnnn</code>	1079	<code>\draw_transform_shift_reset:</code>	1300 , 1426 , 1825 , 1832 , 1838
<code>\draw_point_interpolate_curve-</code> <code>auxv:nnw</code>	1079	<code>\draw_transform_triangle:nnn</code>	509 , 1949 , 1949
<code>\draw_point_interpolate_curve-</code> <code>auxvi:n</code>	1079	<code>\draw_transform_xscale:n</code>	1971 , 1973
<code>\draw_point_interpolate_curve-</code> <code>auxvii:nnnnnnnn</code>	1079	<code>\draw_transform_xshift:n</code>	1971 , 1977
<code>\draw_point_interpolate_curve-</code> <code>auxviii:nnnnnnnn</code>	1079	<code>\draw_transform_xslant:n</code>	1971 , 1981
<code>\draw_point_interpolate_distance:nnn</code>	1029 , 1029 , 1682 , 1691	<code>\draw_transform_yscale:n</code>	1971 , 1975
<code>\draw_point_interpolate_line:nnn</code>	1014 , 1014	<code>\draw_transform_yshift:n</code>	1971 , 1979
<code>\draw_point_intersect_circles:nnnnn</code>	917 , 917	<code>\draw_transform_yslant:n</code>	1971 , 1983
<code>\draw_point_intersect_line-</code> <code>circle:nnnnn</code>	973 , 973	<code>\draw_xvec:n</code>	1155 , 1155 , 1170
<code>\draw_point_intersect_lines:nnnn</code>	892 , 892	<code>\draw_yvec:n</code>	1155 , 1157 , 1171
<code>\draw_point_polar:nn</code>	867 , 867	<code>\draw_zvec:n</code>	1155 , 1159 , 1172
<code>\draw_point_polar:nnn</code>	450 , 456 , 460 , 466 , 867 , 868 , 869	draw internal commands:	
<code>\draw_point_transform:n</code>	37 , 40 , 43 , 46 , 266 , 278 , 294 , 295 , 296 , 328 , 329 , 455 , 459 , 517 , 1216 , 1216	<code>__draw_backend_begin:</code>	1295
<code>\draw_point_unit_vector:n</code>	874 , 874 , 1042	<code>__draw_backend_box_use:Nnnnn</code>	53
<code>\draw_point_vec:nn</code>	1173 , 1173	<code>__draw_backend_cap_but:</code>	1807
<code>\draw_point_vec:nnn</code>	1173 , 1184	<code>__draw_backend_cap_rectangle:</code>	1808
<code>\draw_point_vec_polar:nn</code>	1203 , 1203	<code>__draw_backend_cap_round:</code>	1809
<code>\draw_point_vec_polar:nnn</code>	1203 , 1204 , 1205	<code>__draw_backend_clip:</code>	708
<code>\draw_scope_begin:</code>	26 , 1386 , 1386	<code>__draw_backend_closepath:</code>	1528
<code>\draw_scope_end:</code>	29 , 1393	<code>__draw_backend_curveto:nnnnnn</code>	1532
<code>\draw_suspend_begin:</code>	1421 , 1421	<code>__draw_backend_dash_pattern:nn</code>	1798
<code>\draw_suspend_end:</code>	1421 , 1429	<code>__draw_backend_discardpath:</code>	713
<code>\draw_transform_matrix:nnnn</code>	1862 , 1862 , 1972 , 1974 , 1976 , 1982 , 1984 , 1995	<code>__draw_backend_end:</code>	1317
<code>\draw_transform_matrix_absolute:nnnn</code>	1839 , 1839 , 1877 , 1963	<code>__draw_backend_evenodd_rule:</code>	1810
<code>\draw_transform_matrix_invert:</code>	1915 , 1915	<code>__draw_backend_join_bevel:</code>	1812
<code>\draw_transform_matrix_reset:</code>	1299 , 1425 , 1825 , 1825 , 1837	<code>__draw_backend_join_miter:</code>	1813
<code>\draw_transform_rotate:n</code>	1985 , 1985	<code>__draw_backend_join_round:</code>	1814
<code>\draw_transform_scale:n</code>	1971 , 1971	<code>__draw_backend_lineto:nn</code>	1538
<code>\draw_transform_shift:n</code>	27 , 1862 , 1884 , 1978 , 1980	<code>__draw_backend_linewidth:n</code>	1788
<code>\draw_transform_shift_absolute:n</code>	1839 , 1855 , 1968	<code>__draw_backend_miterlimit:n</code>	1806
<code>\draw_transform_shift_invert:</code>	1915 , 1944	<code>__draw_backend_moveto:nn</code>	1540
		<code>__draw_backend_nonzero_rule:</code>	1811
		<code>__draw_backend_rectangle:nnnn</code>	1545
		<code>__draw_backend_scope_begin:</code>	154 , 1388
		<code>__draw_backend_scope_end:</code>	157 , 1398
		<code>l__draw_baseline_bool</code>	1364 , 1368 , 1373
		<code>l__draw_baseline_dim</code>	1364 , 1369 , 1378
		<code>__draw_baseline_finalise:w</code>	1313 , 1371 , 1371
		<code>__draw_box_use:Nnnnn</code>	14
		<code>__draw_box_use:nNnnnn</code>	21 , 24 , 79
		<code>__draw_box_use:Nnnnnnn</code>	16 , 28 , 31 , 74
		<code>__draw_box_use:nNnnnnnn</code>	14
		<code>__draw_coffin_use:nNnn</code>	72 , 74 , 79 , 82
		<code>l__draw_corner_arc_bool</code>	242 , 250 , 251 , 255 , 583
		<code>l__draw_corner_xarc_dim</code>	240 , 245 , 248 , 258
		<code>l__draw_corner_yarc_dim</code>	240 , 246 , 249 , 259

__draw_draw_polar:nnn	__draw_path_arc_auxiii:nn
..... 867 , 870 , 871 , 873 367 , 447 , 472
__draw_draw_vec_polar:nnn	__draw_path_arc_auxiv:nnnn
..... 1206 , 1207 , 1215 367 , 453 , 478
\l__draw_fill_color_tl	__draw_path_arc_auxv:nn 367 , 463 , 484
1383	__draw_path_arc_auxvi:nn
__draw_finalise: 367 , 486 , 489
..... 1326 , 1333 , 1333 , 1371 , 1381	\l__draw_path_arc_delta_fp
__draw_finalise_baseline:n	367
..... 1333 , 1346 , 1378	\l__draw_path_arc_start_fp
\g__draw_id_int	367
1282 , 1293	__draw_path_curveto:nnnn
__draw_if_recursion_tail_stop- 324 , 327 , 331
do:Nn	__draw_path_curveto:nnnnnn
9 , 9 , 1579 262 , 292 ,
\l__draw_layer_close_bool	298 , 320 , 338 , 468 , 537 , 547 , 557 , 567
..... 108 , 118 , 120 , 137	\c__draw_path_curveto_a_fp
\l__draw_layer_main_box	324
..... 150 , 151 , 1280 , 1309	\c__draw_path_curveto_b_fp
\l__draw_layer_tl	324
106 , 117 , 121	__draw_path_ellipse:nnnnnn
\g__draw_layers_clist 513 , 516 , 521
109	__draw_path_ellipse_arci:nnnnnn
__draw_layers_insert: 144 , 144 , 1316 513 , 527 , 535
__draw_layers_restore: 161 , 172 , 1431	__draw_path_ellipse_arci:nnnnnn
__draw_layers_save: . 161 , 161 , 1427 513 , 528 , 545
\g__draw_linewidth_dim	__draw_path_ellipse_arci:nnnnnn
... 756 , 1390 , 1396 , 1782 , 1787 , 1788 513 , 529 , 555
\l__draw_linewidth_dim	__draw_path_ellipse_arci:nnnnnn
..... 1383 , 1390 , 1396 513 , 530 , 565
\l__draw_main_box	\c__draw_path_ellipse_fp
1280 , 1294 , 1327 , 1330 , 1335 , 1340 ,	513
1342 , 1343 , 1348 , 1353 , 1355 , 1361	__draw_path_grid_auxi:nnnnnn
\l__draw_matrix_a_fp 615 , 619 , 625 , 631
54 ,	__draw_path_grid_auxii:nnnnnn
1228 , 1260 , 1819 , 1827 , 1841 , 1847 , 615 , 628 , 629 , 632
1878 , 1880 , 1903 , 1925 , 1934 , 1941	__draw_path_grid_auxiii:nnnnnn
\l__draw_matrix_active_bool 615 , 635 , 636 , 638
584 , 1223 , 1255 , 1818 , 1852 , 1853 , 1917	__draw_path_grid_auxiiii:nnnnnn
\l__draw_matrix_b_fp	615
55 ,	__draw_path_grid_auxiv:nnnnnnnn
1234 , 1265 , 1819 , 1828 , 1842 , 1848 , 615 , 640 , 645 , 664
1879 , 1881 , 1903 , 1926 , 1936 , 1937	\g__draw_path_lastx_dim
\l__draw_matrix_c_fp 200 , 237 , 342 , 475 , 481 , 773 , 793
56 ,	\l__draw_path_lastx_dim 761 , 773 , 793
1229 , 1261 , 1819 , 1829 , 1843 , 1849 ,	\g__draw_path_lasty_dim
1878 , 1880 , 1911 , 1926 , 1938 , 1939 200 , 238 , 349 , 476 , 482 , 774 , 794
\l__draw_matrix_d_fp	\l__draw_path_lasty_dim 761 , 774 , 794
57 ,	__draw_path_lineto:nn
1235 , 1266 , 1822 , 1830 , 1844 , 1850 , 262 , 277 , 280 , 314
1879 , 1881 , 1911 , 1925 , 1935 , 1940	__draw_path_mark_corner:
__draw_path_arc:nnnn . 367 , 373 , 380	253 , 253 , 282 , 291 , 308 , 319 , 337 , 408
__draw_path_arc:nnNnn	__draw_path_moveto:nn
..... 367 , 383 , 384 , 386 262 , 265 , 268 , 312 , 525 , 533
\c__draw_path_arc_60_fp	__draw_path_rectangle:nnnn
367 578 , 586 , 591
\c__draw_path_arc_90_fp	__draw_path_rectangle_corners:nnnn
367 607
__draw_path_arc_add:nnnn	__draw_path_rectangle_corners:nnnnnn
367 610 , 613
__draw_path_arc_aux_add:nn	
..... 474 , 480 , 492 , 497	
__draw_path_arc_auxi:nnnnNnn	
..... 367 , 394 , 401 , 409 , 416 , 442	
__draw_path_arc_auxii:nnnnNnnnn	
..... 367 , 420 , 443	

```

\__draw_path_rectangle_rounded:nnnn
..... 578, 585, 598
\__draw_path_reset_limits: .....
..... 206, 228, 682, 781, 1297
\l__draw_path_tmp_tl .....
..... 197, 445, 468, 487, 491, 495, 499
\l__draw_path_tmpa_fp .....
..... 197, 333, 343, 355
\l__draw_path_tmpb_fp .....
..... 197, 334, 350, 359
\__draw_path_update_last:nn .....
..... 235, 235, 272, 285, 304, 596
\__draw_path_update_limits:nn ...
..... 36, 39, 42, 45, 206,
206, 270, 283, 300, 301, 302, 593, 594
\__draw_path_use:n . 670, 673, 684, 686
\__draw_path_use_action_draw: ...
..... 670, 730
\__draw_path_use_action_fillstroke:
..... 670, 734
\l__draw_path_use_bb_bool ..... 668
\l__draw_path_use_clear_bool 668, 727
\l__draw_path_use_clip_bool .....
..... 665, 688, 706
\l__draw_path_use_fill_bool .....
..... 665, 689, 711, 716, 722, 736
\__draw_path_use_stroke_bb: .....
..... 670, 704, 739
\__draw_path_use_stroke_bb_-
aux:NnN 670, 741, 742, 743, 744, 746
\l__draw_path_use_stroke_bool ...
665, 690, 703, 712, 717, 723, 732, 737
\g__draw_path_xmax_dim .....
..... 202, 208, 209, 230, 775, 789
\l__draw_path_xmax_dim . 761, 775, 789
\g__draw_path_xmin_dim .....
..... 202, 210, 211, 231, 776, 790
\l__draw_path_xmin_dim . 761, 776, 790
\g__draw_path_ymax_dim .....
..... 202, 212, 213, 232, 777, 791
\l__draw_path_ymax_dim . 761, 777, 791
\g__draw_path_ymin_dim .....
..... 202, 214, 215, 233, 778, 792
\l__draw_path_ymin_dim . 761, 778, 792
\__draw_point_interpolate_-
arcaxes_auxi:nnnnnnnnn .....
..... 1047, 1050, 1053
\__draw_point_interpolate_-
arcaxes_auxii:nnnnnnnnn .....
..... 1047, 1055, 1058, 1064
\__draw_point_interpolate_-
arcaxes_auxiii:nnnnnnnn .....
..... 1047, 1060, 1065, 1072
\__draw_point_interpolate_-
arcaxes_auxiv:nnnnnnnn .....
..... 1047, 1067, 1073, 1078
\__draw_point_interpolate_curve_-
auxi:nnnnnnnnn ..... 1082, 1085
\__draw_point_interpolate_curve_-
auxii:nnnnnnnnn . 1087, 1091, 1099
\__draw_point_interpolate_curve_-
auxiii:nnnnnnn ... 1094, 1105, 1113
\__draw_point_interpolate_curve_-
auxiv:nnnnnn 1107, 1108, 1109, 1114
\__draw_point_interpolate_curve_-
auxv:nnw ..... 1116, 1120, 1127
\__draw_point_interpolate_curve_-
auxvi:n ..... 1111, 1132
\__draw_point_interpolate_curve_-
auxvii:nnnnnnnn ..... 1133, 1134
\__draw_point_interpolate_curve_-
auxviii:nnnnnn . 1136, 1143, 1148
\__draw_point_interpolate_-
distance:nnnn ..... 1032, 1035
\__draw_point_interpolate_-
distance:nnnnn .....
..... 1029, 1039, 1044, 1046
\__draw_point_interpolate_-
distance:nnnnnn ..... 1029
\__draw_point_interpolate_line_-
aux:nnnnn .. 1014, 1017, 1020, 1025
\__draw_point_interpolate_line_-
aux:nnnnnn . 1014, 1022, 1026, 1028
\__draw_point_intersect_circles_-
auxi:nnnnnnnn ..... 917, 920, 923
\__draw_point_intersect_circles_-
auxii:nnnnnnnn .. 917, 925, 928, 935
\__draw_point_intersect_circles_-
auxiii:nnnnnnnn . 917, 930, 936, 942
\__draw_point_intersect_circles_-
auxiv:nnnnnnnnn . 917, 938, 943, 950
\__draw_point_intersect_circles_-
auxv:nnnnnnnnnn . 917, 945, 951, 957
\__draw_point_intersect_circles_-
auxvi:nnnnnnnnn . 917, 953, 958, 966
\__draw_point_intersect_circles_-
auxvii:nnnnnnnn . 917, 960, 967, 972
\__draw_point_intersect_line_-
circle_auxi:nnnnnnnn 973, 976, 979
\__draw_point_intersect_line_-
circle_auxii:nnnnnnnn .....
..... 973, 981, 984, 992
\__draw_point_intersect_line_-
circle_auxiii:nnnnnnnn .....
..... 973, 986, 993, 1000
\__draw_point_intersect_line_-
circle_auxiv:nnnnnnnn .....

```

```

..... 973, 995, 1001, 1007
\__draw_point_intersect_line_-
  circle_auxv:nnnnn .....
..... 973, 1003, 1008, 1013
\__draw_point_intersect_lines:nnnnnn
..... 892
\__draw_point_intersect_lines:nnnnnnnn
..... 892, 895, 898
\__draw_point_intersect_lines_-
  aux:nnnnnn ..... 892, 900, 908, 916
\__draw_point_process:nn .....
  35, 38, 41, 44, 264, 276, 312, 314,
  446, 462, 806, 806, 875, 1031, 1037,
  1163, 1218, 1250, 1857, 1886, 1953
\__draw_point_process:nnn 326, 452,
  580, 609, 617, 806, 817, 919, 1016, 1951
\__draw_point_process:nnnn .....
  ... 289, 317, 515, 806, 829, 975, 1049
\__draw_point_process:nnnnn .....
  ..... 806, 843, 894, 1081
\__draw_point_process_auxi:nn .....
  ..... 806, 808, 812, 814
\__draw_point_process_auxii:nw ..
  ..... 806, 813, 815
\__draw_point_process_auxiii:nnn
  ..... 806, 819, 824, 826
\__draw_point_process_auxiv:nw ..
  ..... 806, 825, 827
\__draw_point_process_auxv:nnnn ..
  ..... 806, 831, 837, 839
\__draw_point_process_auxvi:nw ..
  ..... 806, 838, 840
\__draw_point_process_auxvii:nnnnn
  ..... 806, 845, 852, 857
\__draw_point_process_auxviii:nw
  ..... 806, 854, 858
\__draw_point_to_dim:n .....
  ..... 861, 862, 863, 865
\__draw_point_to_dim:w . 861, 864, 866
\__draw_point_transform:nn .....
  ..... 1216, 1219, 1221
\__draw_point_transform_noshift:n
  ..... 449, 465, 518, 519, 1248, 1248
\__draw_point_transform_noshift:nn
  ..... 1248, 1251, 1253
\__draw_point_unit_vector:nn .....
  ..... 874, 875, 876
\__draw_point_unit_vector:nnn .....
  ..... 874, 878, 882, 891
\__draw_point_vec:nn .....
  ..... 1173, 1174, 1175, 1183
\__draw_point_vec:nnn .....
  ..... 1173, 1186, 1189, 1202
\__draw_point_vec_polar:nnn .. 1203

\__draw_reset_bb: .....
  ..... 1283, 1283, 1296, 1411
\__draw_scope_bb_begin: .....
  ..... 1404, 1404, 1423
\__draw_scope_bb_end: 1404, 1413, 1433
\__draw_softpath_add:n .....
  ... 1441, 1441, 1443, 1470, 1481,
  1490, 1499, 1504, 1514, 1522, 1777
\c__draw_softpath_arc_fp .....
  ..... 1556, 1719, 1723, 1728, 1732
\__draw_softpath_clear: .....
  681, 728, 1301, 1444, 1451, 1465, 1469
\__draw_softpath_close_op:nn ...
  .. 1483, 1527, 1527, 1618, 1654, 1756
\__draw_softpath_closepath: .....
  ..... 309, 532, 1479
\l__draw_softpath_corneri_dim ...
  ..... 1550, 1603, 1606, 1692
\l__draw_softpath_cornerii_dim ..
  ..... 1550, 1604, 1607, 1683
\g__draw_softpath_corners_bool ..
  ..... 1440,
  1454, 1464, 1472, 1524, 1559, 1575
\l__draw_softpath_corners_bool ..
  ..... 761, 1463, 1473
\l__draw_softpath_curve_end_tl ...
  ..... 1549, 1680, 1699, 1748, 1759
\__draw_softpath_curveto:nnnnnn ..
  ..... 303, 1479, 1488
\__draw_softpath_curveto_opi:nn ..
  .. 1492, 1527, 1529, 1615, 1653, 1716
\__draw_softpath_curveto_-
  opi:nnNnnNnn ..... 1527, 1530, 1531
\__draw_softpath_curveto_opii:nn
  ..... 1493, 1527, 1533, 1534, 1725
\__draw_softpath_curveto_-
  opiii:nn 1494, 1527, 1535, 1536, 1734
\l__draw_softpath_first_tl .....
  ..... 1550, 1566, 1583,
  1584, 1594, 1613, 1614, 1640, 1644
\g__draw_softpath_lastx_dim .....
  ..... 779, 787, 1475, 1484, 1508
\l__draw_softpath_lastx_dim .....
  ..... 767, 779, 787
\l__draw_softpath_lastx_fp .....
  .. 1550, 1564, 1585, 1633, 1695, 1702
\g__draw_softpath_lasty_dim .....
  ..... 780, 788, 1475, 1485, 1509
\l__draw_softpath_lasty_dim .....
  ..... 768, 780, 788
\l__draw_softpath_lasty_fp .....
  .. 1550, 1565, 1586, 1634, 1696, 1703
\__draw_softpath_lineto:nn .....
  ..... 284, 1479, 1497

```

```

\__draw_softpath_lineto_op:nn ...
... 1500, 1527, 1537, 1621, 1652, 1765
\g__draw_softpath_main_tl .....
..... 1438, 1442, 1447,
1453, 1458, 1461, 1568, 1570, 1776
\l__draw_softpath_main_tl .....
..... 20, 1460, 1470, 1547,
1562, 1589, 1591, 1772, 1774, 1777
\g__draw_softpath_move_bool ....
..... 1477, 1506
\l__draw_softpath_move_tl .....
..... 1550, 1567,
1590, 1593, 1641, 1743, 1766, 1773
\__draw_softpath_moveto:nn .....
..... 271, 1479, 1502
\__draw_softpath_moveto_op:nn ...
..... 1505, 1527, 1539, 1587, 1745
\l__draw_softpath_part_tl .....
..... 1548, 1563,
1592, 1595, 1597, 1631, 1686, 1775
\__draw_softpath_rectangle:nnnn .
..... 595, 1479, 1512
\__draw_softpath_rectangle_-
opi:nn ..... 1516, 1527, 1542
\__draw_softpath_rectangle_-
opi:nnNnn ..... 1527, 1543, 1544
\__draw_softpath_rectangle_-
opii:nn ..... 1517, 1527, 1546
\__draw_softpath_restore: .....
..... 786, 1456, 1467
\__draw_softpath_round_action:nn
..... 1557, 1581, 1601
\__draw_softpath_round_action:Nnn
..... 1557, 1609, 1611
\__draw_softpath_round_action_-
close: ..... 1557, 1619, 1637
\__draw_softpath_round_action_-
curveto:NnnNnn .. 1557, 1616, 1628
\__draw_softpath_round_calc:NnnNnn
..... 1557, 1657, 1672, 1678, 1741
\__draw_softpath_round_calc:nnnnnn
..... 1557, 1689, 1706, 1711
\__draw_softpath_round_calc:nnnnw
..... 1557, 1708, 1712
\__draw_softpath_round_close:nn .
..... 1557, 1643, 1737
\__draw_softpath_round_close:w ..
..... 1557, 1747, 1758, 1769
\__draw_softpath_round_corners: .
..... 700, 1557, 1557
\__draw_softpath_round_end: ....
..... 1557, 1579, 1770
\__draw_softpath_round_lookahead:NnnNnn
..... 1557, 1622, 1635, 1648

\__draw_softpath_round_loop:Nnn .
... 1557, 1569, 1577, 1598, 1608,
1623, 1646, 1658, 1664, 1673, 1755
\__draw_softpath_round_roundpoint:NnnNnnNnn
..... 1557, 1663, 1669
\__draw_softpath_roundpoint:nn ..
..... 257, 1479, 1520, 1526
\__draw_softpath_roundpoint_-
op:nn .. 1523, 1527, 1541, 1580, 1662
\__draw_softpath_save: 782, 1456, 1456
\l__draw_softpath_tmp_tl .....
..... 1439, 1448, 1449
\__draw_softpath_use: 705, 1444, 1444
\l__draw_stroke_color_tl ..... 1383
\l__draw_tmp_box ..... 13, 49, 60,
64, 66, 67, 68, 69, 85, 87, 88, 89, 90, 91
\l__draw_tmp_seq ..... 1790
\__draw_transform:nnnn .....
..... 1862, 1866, 1873
\__draw_transform_invert:n .....
..... 1915, 1919, 1932, 1943
\__draw_transform_rotate:n .....
..... 1985, 1986, 1987, 1993
\__draw_transform_rotate:nn .....
..... 1985, 1989, 1994, 1996
\__draw_transform_shift:nn .....
..... 1862, 1887, 1889
\__draw_transform_shift:nnnn .....
..... 1861, 1891, 1896, 1896
\__draw_transform_shift_absolute:nn
..... 1839, 1858, 1860
\__draw_transform_triangle:nnnnnn
..... 1954, 1959
\__draw_vec:nn .....
..... 1155, 1156, 1158, 1160, 1161
\__draw_vec:nnn ..... 1155, 1163, 1165
\g__draw_xmax_dim ..... 218,
219, 1275, 1285, 1321, 1328, 1407, 1415
\l__draw_xmax_dim ... 1400, 1407, 1415
\g__draw_xmin_dim .....
..... 220, 221, 1275, 1286, 1319,
1322, 1328, 1337, 1350, 1408, 1416
\l__draw_xmin_dim ... 1400, 1408, 1416
\l__draw_xshift_dim .... 62, 1230,
1244, 1819, 1834, 1892, 1898, 1946
\l__draw_xvec_x_dim .....
..... 1149, 1179, 1193, 1211
\l__draw_xvec_y_dim . 1149, 1180, 1197
\g__draw_ymax_dim . 222, 223, 1275,
1287, 1323, 1344, 1362, 1409, 1417
\l__draw_ymax_dim ... 1400, 1409, 1417
\g__draw_ymin_dim .....
..... 224, 225, 1275, 1288,
1324, 1339, 1344, 1358, 1410, 1418

```


