

File I

Implementation

1 l3draw implementation

```

1 <*package>
2 <@@=draw>
3 \ProvidesExplPackage{l3draw}{2026-05-15}{}
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         \dim_horizontal: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
\draw_coffin_use:Nnnn   width before the reference point.
\__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_set_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. After creating a local copy of the layer box, we must clear the global one so nested drawings do not inherit the parent's layers.

```

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       \box_gc_clear:c { g__draw_layer_ ##1 _box }
170     }
171   }
172 }
173 \cs_new_protected:Npn \__draw_layers_restore:
174 {
175   \clist_map_inline:Nn \l_draw_layers_clist
176   {
177     \str_if_eq:nnF {##1} { main }
178     {
179       \box_gset_eq:cc { g__draw_layer_ ##1 _box }
180       { l__draw_layer_ ##1 _box }
181     }
182   }
183 }
```

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

```

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

```

188 { The~'main'~layer~is~reserved. }
189 \msg_new:nnnn { draw } { unknown-layer }
190 { Layer~'#1'~has~not~been~created. }
191 { You~have~tried~to~use~layer~'#1',~but~it~was~never~set~up. }
192 % \end{macrocode}
193 %
194 % \begin{macrocode}
195 \end{package}

```

4 l3draw-paths implementation

```

196 \*package
197 \@@=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 specialized 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 198 \tl_new:N \l__draw_path_tmp_tl
\l__draw_path_tmpb_fp 199 \fp_new:N \l__draw_path_tmpa_fp
200 \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 201 \dim_new:N \g__draw_path_lastx_dim
202 \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 203 \dim_new:N \g__draw_path_xmax_dim
\g__draw_path_ymax_dim 204 \dim_new:N \g__draw_path_xmin_dim
\g__draw_path_ymin_dim 205 \dim_new:N \g__draw_path_ymax_dim
206 \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:`

```

207 \cs_new_protected:Npn \__draw_path_update_limits:nn #1#2
208 {
209   \dim_gset:Nn \g__draw_path_xmax_dim
210   { \dim_max:nn \g__draw_path_xmax_dim {#1} }
211   \dim_gset:Nn \g__draw_path_xmin_dim
212   { \dim_min:nn \g__draw_path_xmin_dim {#1} }
213   \dim_gset:Nn \g__draw_path_ymax_dim
214   { \dim_max:nn \g__draw_path_ymax_dim {#2} }
215   \dim_gset:Nn \g__draw_path_ymin_dim
216   { \dim_min:nn \g__draw_path_ymin_dim {#2} }
217   \bool_if:NT \l_draw_bb_update_bool
218   {
219     \dim_gset:Nn \g_draw_bb_xmax_dim
220     { \dim_max:nn \g_draw_bb_xmax_dim {#1} }
221     \dim_gset:Nn \g_draw_bb_xmin_dim
222     { \dim_min:nn \g_draw_bb_xmin_dim {#1} }
223     \dim_gset:Nn \g_draw_bb_ymax_dim
224     { \dim_max:nn \g_draw_bb_ymax_dim {#2} }
225     \dim_gset:Nn \g_draw_bb_ymin_dim
226     { \dim_min:nn \g_draw_bb_ymin_dim {#2} }
227   }
228 }
229 \cs_new_protected:Npn \__draw_path_reset_limits:
230 {
231   \dim_gset:Nn \g__draw_path_xmax_dim { -\c_max_dim }
232   \dim_gset:Nn \g__draw_path_xmin_dim { \c_max_dim }
233   \dim_gset:Nn \g__draw_path_ymax_dim { -\c_max_dim }
234   \dim_gset:Nn \g__draw_path_ymin_dim { \c_max_dim }
235 }

```

(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.

```

236 \cs_new_protected:Npn \__draw_path_update_last:nn #1#2
237 {
238   \dim_gset:Nn \g__draw_path_lastx_dim {#1}
239   \dim_gset:Nn \g__draw_path_lasty_dim {#2}
240 }

```

(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
241 \dim_new:N \l__draw_corner_xarc_dim
242 \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.

```
243 \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.

```
244 \cs_new_protected:Npn \draw_path_corner_arc:nn #1#2
245 {
246   \dim_set:Nn \l__draw_corner_xarc_dim { \fp_to_dim:n {#1} }
247   \dim_set:Nn \l__draw_corner_yarc_dim { \fp_to_dim:n {#2} }
248   \bool_lazy_and:nnTF
249     { \dim_compare_p:nNn \l__draw_corner_xarc_dim = { Opt } }
250     { \dim_compare_p:nNn \l__draw_corner_yarc_dim = { Opt } }
251     { \bool_set_false:N \l__draw_corner_arc_bool }
252     { \bool_set_true:N \l__draw_corner_arc_bool }
253 }
```

(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.

```
254 \cs_new_protected:Npn \__draw_path_mark_corner:
255 {
256   \bool_if:NT \l__draw_corner_arc_bool
257   {
258     \__draw_softpath_roundpoint:VV
259     \l__draw_corner_xarc_dim
260     \l__draw_corner_yarc_dim
261   }
262 }
```

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



```

279     { \draw_point_transform:n {#1} }
280   }
281 \cs_new_protected:Npn \__draw_path_lineto:nn #1#2
282 {
283   \__draw_path_mark_corner:
284   \__draw_path_update_limits:nn {#1} {#2}
285   \__draw_softpath_lineto:nn {#1} {#2}
286   \__draw_path_update_last:nn {#1} {#2}
287 }
288 \cs_new_protected:Npn \draw_path_curveto:nnn #1#2#3
289 {
290   \__draw_point_process:nnnn
291   {
292     \__draw_path_mark_corner:
293     \__draw_path_curveto:nnnnnn
294   }
295   { \draw_point_transform:n {#1} }
296   { \draw_point_transform:n {#2} }
297   { \draw_point_transform:n {#3} }
298 }
299 \cs_new_protected:Npn \__draw_path_curveto:nnnnnn #1#2#3#4#5#6
300 {
301   \__draw_path_update_limits:nn {#1} {#2}
302   \__draw_path_update_limits:nn {#3} {#4}
303   \__draw_path_update_limits:nn {#5} {#6}
304   \__draw_softpath_curveto:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
305   \__draw_path_update_last:nn {#5} {#6}
306 }

```

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

\draw_path_close: A simple wrapper.

```

307 \cs_new_protected:Npn \draw_path_close:
308 {
309   \__draw_path_mark_corner:
310   \__draw_softpath_closepath:
311 }

```

(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
312 \cs_new_protected:Npn \draw_path_canvas_moveto:n #1
313 { \__draw_point_process:nn { \__draw_path_moveto:nn } { \draw_point:n {#1} } }
314 \cs_new_protected:Npn \draw_path_canvas_lineto:n #1
315 { \__draw_point_process:nn { \__draw_path_lineto:nn } { \draw_point:n {#1} } }
316 \cs_new_protected:Npn \draw_path_canvas_curveto:nnn #1#2#3
317 {
318   \__draw_point_process:nnnn
319   {
320     \__draw_path_mark_corner:
321     \__draw_path_curveto:nnnnnn
322   }

```

```

323     { \draw_point:n {#1} }
324     { \draw_point:n {#2} }
325     { \draw_point:n {#3} }
326 }

```

(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

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

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

```

327 \cs_new_protected:Npn \draw_path_curveto:nn #1#2
328 {
329   \__draw_point_process:nnn
330   { \__draw_path_curveto:nnnn }
331   { \draw_point_transform:n {#1} }
332   { \draw_point_transform:n {#2} }
333 }
334 \cs_new_protected:Npn \__draw_path_curveto:nnnn #1#2#3#4
335 {
336   \fp_set:Nn \l__draw_path_tmpa_fp { \c__draw_path_curveto_b_fp * #1 }
337   \fp_set:Nn \l__draw_path_tmpb_fp { \c__draw_path_curveto_b_fp * #2 }
338   \use:e
339   {
340     \__draw_path_mark_corner:
341     \__draw_path_curveto:nnnnnn
342     {
343       \fp_to_dim:n
344       {
345         \c__draw_path_curveto_a_fp * \g__draw_path_lastx_dim
346         + \l__draw_path_tmpa_fp
347       }
348     }
349     {
350       \fp_to_dim:n
351       {
352         \c__draw_path_curveto_a_fp * \g__draw_path_lasty_dim
353         + \l__draw_path_tmpb_fp
354       }
355     }
356     {
357       \fp_to_dim:n
358       { \c__draw_path_curveto_a_fp * #3 + \l__draw_path_tmpa_fp }
359     }

```

```

360         {
361             \fp_to_dim:n
362             { \c__draw_path_curveto_a_fp * #4 + \l__draw_path_tmpb_fp }
363         }
364         {#3}
365         {#4}
366     }
367 }
368 \fp_const:Nn \c__draw_path_curveto_a_fp { 1 / 3 }
369 \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 we can cover at most 90° at once. To allow for later manipulations, we aim to have roughly equal last segments to the line, with the split set at a final part of 115°.

```

\__draw_path_arc:nnnn
\__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
370 \cs_new_protected:Npn \draw_path_arc:nnn #1#2#3
371 { \draw_path_arc:nnnn {#1} {#1} {#2} {#3} }
372 \cs_new_protected:Npn \draw_path_arc:nnnn #1#2#3#4
373 {
374     \use:e
375     {
376         \__draw_path_arc:nnnn
377         { \fp_eval:n {#3} }
378         { \fp_eval:n {#4} }
379         { \fp_to_dim:n {#1} }
380         { \fp_to_dim:n {#2} }
381     }
382 }
383 \cs_new_protected:Npn \__draw_path_arc:nnnn #1#2#3#4
384 {
385     \fp_compare:nNnTF {#1} > {#2}
386     { \__draw_path_arc:nnNnn {#1} {#2} - {#3} {#4} }
387     { \__draw_path_arc:nnNnn {#1} {#2} + {#3} {#4} }
388 }
389 \cs_new_protected:Npn \__draw_path_arc:nnNnn #1#2#3#4#5
390 {
391     \fp_set:Nn \l__draw_path_arc_start_fp {#1}
392     \fp_set:Nn \l__draw_path_arc_delta_fp { abs( #1 - #2 ) }
393     \fp_while_do:nNnn { \l__draw_path_arc_delta_fp } > { 90 }
394     {
395         \fp_compare:nNnTF \l__draw_path_arc_delta_fp > { 115 }
396         {
397             \__draw_path_arc_auxi:eennNnn
398             { \fp_to_decimal:N \l__draw_path_arc_start_fp }
399             { \fp_eval:n { \l__draw_path_arc_start_fp #3 90 } }
400             { 90 } {#2}
401             #3 {#4} {#5}
402         }
403         {
404             \__draw_path_arc_auxi:eennNnn
405             { \fp_to_decimal:N \l__draw_path_arc_start_fp }
406             { \fp_eval:n { \l__draw_path_arc_start_fp #3 60 } }
407             { 60 } {#2}

```

```

408         #3 {#4} {#5}
409     }
410 }
411 \__draw_path_mark_corner:
412 \__draw_path_arc_auxi:enenNnn
413 { \fp_to_decimal:N \l__draw_path_arc_start_fp }
414 {#2}
415 { \fp_eval:n { abs( \l__draw_path_arc_start_fp - #2 ) } }
416 {#2}
417 #3 {#4} {#5}
418 }

```

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.

```

419 \cs_new_protected:Npn \__draw_path_arc_auxi:nnnnNnn #1#2#3#4#5#6#7
420 {
421   \use:e
422   {
423     \__draw_path_arc_auxii:nnnNnnnn
424     {#1} {#2} {#4} #5 {#6} {#7}
425     {
426       \fp_to_dim:n
427       {
428         \cs_if_exist_use:cF
429         { c__draw_path_arc_ #3 _fp }
430         { 4/3 * tand( 0.25 * #3 ) }
431         * #6
432       }
433     }
434     {
435       \fp_to_dim:n
436       {
437         \cs_if_exist_use:cF
438         { c__draw_path_arc_ #3 _fp }
439         { 4/3 * tand( 0.25 * #3 ) }
440         * #7
441       }
442     }
443   }
444 }
445 \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

```

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

```

```

451     {
452       \__draw_point_transform_noshift:n
453       { \draw_point_polar:nnn {#7} {#8} { #1 #4 90 } }
454     }
455   \__draw_point_process:nnn
456   { \__draw_path_arc_auxiv:nnnn }
457   {
458     \draw_point_transform:n
459     { \draw_point_polar:nnn {#5} {#6} {#1} }
460   }
461   {
462     \draw_point_transform:n
463     { \draw_point_polar:nnn {#5} {#6} {#2} }
464   }
465   \__draw_point_process:nn
466   { \__draw_path_arc_auxv:nn }
467   {
468     \__draw_point_transform_noshift:n
469     { \draw_point_polar:nnn {#7} {#8} { #2 #4 -90 } }
470   }
471   \exp_after:wN \__draw_path_curveto:nnnnnn \l__draw_path_tmp_tl
472   \fp_set:Nn \l__draw_path_arc_delta_fp { abs ( #2 - #3 ) }
473   \fp_set:Nn \l__draw_path_arc_start_fp {#2}
474 }

```

The first control point.

```

475 \cs_new_protected:Npn \__draw_path_arc_auxiii:nn #1#2
476 {
477   \__draw_path_arc_aux_add:nn
478   { \g__draw_path_lastx_dim + #1 }
479   { \g__draw_path_lasty_dim + #2 }
480 }

```

The end point: simple arithmetic.

```

481 \cs_new_protected:Npn \__draw_path_arc_auxiv:nnnn #1#2#3#4
482 {
483   \__draw_path_arc_aux_add:nn
484   { \g__draw_path_lastx_dim - #1 + #3 }
485   { \g__draw_path_lasty_dim - #2 + #4 }
486 }

```

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

```

487 \cs_new_protected:Npn \__draw_path_arc_auxv:nn #1#2
488 {
489   \exp_after:wN \__draw_path_arc_auxvi:nn
490   \l__draw_path_tmp_tl {#1} {#2}
491 }
492 \cs_new_protected:Npn \__draw_path_arc_auxvi:nn #1#2#3#4#5#6
493 {
494   \tl_set:Nn \l__draw_path_tmp_tl { {#1} {#2} }
495   \__draw_path_arc_aux_add:nn
496   { #5 + #3 }
497   { #6 + #4 }
498   \tl_put_right:Nn \l__draw_path_tmp_tl { {#3} {#4} }
499 }

```

```

500 \cs_new_protected:Npn \__draw_path_arc_aux_add:nn #1#2
501 {
502   \tl_put_right:Ne \l__draw_path_tmp_tl
503   { { \fp_to_dim:n {#1} } { \fp_to_dim:n {#2} } }
504 }
505 \fp_new:N \l__draw_path_arc_delta_fp
506 \fp_new:N \l__draw_path_arc_start_fp
507 \fp_const:cn { c__draw_path_arc_90_fp } { 4/3 * (sqrt(2) - 1) }
508 \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.

```

509 \cs_new_protected:Npn \draw_path_arc_axes:nnnn #1#2#3#4
510 {
511   \group_begin:
512     \draw_transform_triangle:nnn { 0cm , 0cm } {#1} {#2}
513     \draw_path_arc:nnn { 1pt } {#3} {#4}
514   \group_end:
515 }

```

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

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

```

```

543     { \fp_to_dim:n { #1 + #3 * \c__draw_path_ellipse_fp + #5 } }
544     { \fp_to_dim:n { #2 + #4 * \c__draw_path_ellipse_fp + #6 } }
545     { \fp_to_dim:n { #1 + #5 } }
546     { \fp_to_dim:n { #2 + #6 } }
547   }
548 \cs_new:Npn \__draw_path_ellipse_arci:nnnnnn #1#2#3#4#5#6
549 {
550   \__draw_path_curveto:nnnnnn
551   { \fp_to_dim:n { #1 - #3 * \c__draw_path_ellipse_fp + #5 } }
552   { \fp_to_dim:n { #2 - #4 * \c__draw_path_ellipse_fp + #6 } }
553   { \fp_to_dim:n { #1 - #3 + #5 * \c__draw_path_ellipse_fp } }
554   { \fp_to_dim:n { #2 - #4 + #6 * \c__draw_path_ellipse_fp } }
555   { \fp_to_dim:n { #1 - #3 } }
556   { \fp_to_dim:n { #2 - #4 } }
557 }
558 \cs_new:Npn \__draw_path_ellipse_arci:nnnnnn #1#2#3#4#5#6
559 {
560   \__draw_path_curveto:nnnnnn
561   { \fp_to_dim:n { #1 - #3 - #5 * \c__draw_path_ellipse_fp } }
562   { \fp_to_dim:n { #2 - #4 - #6 * \c__draw_path_ellipse_fp } }
563   { \fp_to_dim:n { #1 - #3 * \c__draw_path_ellipse_fp - #5 } }
564   { \fp_to_dim:n { #2 - #4 * \c__draw_path_ellipse_fp - #6 } }
565   { \fp_to_dim:n { #1 - #5 } }
566   { \fp_to_dim:n { #2 - #6 } }
567 }
568 \cs_new:Npn \__draw_path_ellipse_arciv:nnnnnn #1#2#3#4#5#6
569 {
570   \__draw_path_curveto:nnnnnn
571   { \fp_to_dim:n { #1 + #3 * \c__draw_path_ellipse_fp - #5 } }
572   { \fp_to_dim:n { #2 + #4 * \c__draw_path_ellipse_fp - #6 } }
573   { \fp_to_dim:n { #1 + #3 - #5 * \c__draw_path_ellipse_fp } }
574   { \fp_to_dim:n { #2 + #4 - #6 * \c__draw_path_ellipse_fp } }
575   { \fp_to_dim:n { #1 + #3 } }
576   { \fp_to_dim:n { #2 + #4 } }
577 }
578 \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.

```

579 \cs_new_protected:Npn \draw_path_circle:nn #1#2
580 { \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
581 \cs_new_protected:Npn \draw_path_rectangle:nn #1#2
582 {
583   \bool_lazy_or:nnTF
584   { \l__draw_corner_arc_bool }
585   { \l__draw_matrix_active_bool }

```

```

586     {
587         \__draw_point_process:nnn \__draw_path_rectangle_rounded:nnnn
588         { \draw_point:n {#1} }
589         { \draw_point:n {#2} }
590     }
591     {
592         \__draw_point_process:nnn \__draw_path_rectangle:nnnn
593         { \draw_point:n { (#1) + ( \l__draw_xshift_dim , \l__draw_yshift_dim ) } }
594         { \draw_point:n {#2} }
595     }
596 }
597 \cs_new_protected:Npn \__draw_path_rectangle:nnnn #1#2#3#4
598 {
599     \__draw_path_update_limits:nn {#1} {#2}
600     \__draw_path_update_limits:nn { #1 + #3 } { #2 + #4 }
601     \__draw_softpath_rectangle:nnnn {#1} {#2} {#3} {#4}
602     \__draw_path_update_last:nn {#1} {#2}
603 }
604 \cs_new_protected:Npn \__draw_path_rectangle_rounded:nnnn #1#2#3#4
605 {
606     \draw_path_moveto:n { #1 + #3 , #2 + #4 }
607     \draw_path_lineto:n { #1 , #2 + #4 }
608     \draw_path_lineto:n { #1 , #2 }
609     \draw_path_lineto:n { #1 + #3 , #2 }
610     \draw_path_close:
611     \draw_path_moveto:n { #1 , #2 }
612 }

```

(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`

Another shortcut wrapper.

```

613 \cs_new_protected:Npn \draw_path_rectangle_corners:nn #1#2
614 {
615     \__draw_point_process:nnn
616     { \__draw_path_rectangle_corners:nnnnn {#1} }
617     { \draw_point:n {#1} }
618     { \draw_point:n {#2} }
619 }
620 \cs_new_protected:Npn \__draw_path_rectangle_corners:nnnnn #1#2#3#4#5
621 { \draw_path_rectangle:nn {#1} { #4 - #2 , #5 - #3 } }

```

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

4.7 Grids

`\draw_path_grid:nnnn`
`__draw_path_grid_auxi:nnnnnn`
`__draw_path_grid_auxii:eennnn`
`__draw_path_grid_auxiii:nnnnnn`
`__draw_path_grid_auxiiii:eennnn`
`__draw_path_grid_auxiv:nnnnnnnn`
`__draw_path_grid_auxiv:eennnnnn`

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

```

622 \cs_new_protected:Npn \draw_path_grid:nnnn #1#2#3#4
623 {
624     \__draw_point_process:nnn
625     {
626         \__draw_path_grid_auxi:eennnn

```



```

627         { \dim_abs:n {#1} }
628         { \dim_abs:n {#2} }
629     }
630     { \draw_point:n {#3} }
631     { \draw_point:n {#4} }
632 }
633 \cs_new_protected:Npn \__draw_path_grid_auxi:nnnnnn #1#2#3#4#5#6
634 {
635     \dim_compare:nNnTF {#3} > {#5}
636     { \__draw_path_grid_auxii:nnnnnn {#1} {#2} {#5} {#4} {#3} {#6} }
637     { \__draw_path_grid_auxii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6} }
638 }
639 \cs_generate_variant:Nn \__draw_path_grid_auxi:nnnnnn { ee }
640 \cs_new_protected:Npn \__draw_path_grid_auxii:nnnnnn #1#2#3#4#5#6
641 {
642     \dim_compare:nNnTF {#4} > {#6}
643     { \__draw_path_grid_auxiii:nnnnnn {#1} {#2} {#3} {#6} {#5} {#4} }
644     { \__draw_path_grid_auxiii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6} }
645 }
646 \cs_new_protected:Npn \__draw_path_grid_auxiii:nnnnnn #1#2#3#4#5#6
647 {
648     \__draw_path_grid_auxiv:eennnnnn
649     { \fp_to_dim:n { #1 * ceil(#3/(#1)) } }
650     { \fp_to_dim:n { #2 * ceil(#4/(#2)) } }
651     {#1} {#2} {#3} {#4} {#5} {#6}
652 }
653 \cs_new_protected:Npn \__draw_path_grid_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
654 {
655     \dim_step_inline:nnnn
656     {#1}
657     {#3}
658     {#7}
659     {
660         \draw_path_moveto:n { ##1 , #6 }
661         \draw_path_lineto:n { ##1 , #8 }
662     }
663     \dim_step_inline:nnnn
664     {#2}
665     {#4}
666     {#8}
667     {
668         \draw_path_moveto:n { #5 , ##1 }
669         \draw_path_lineto:n { #7 , ##1 }
670     }
671 }
672 \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	673 \bool_new:N \l__draw_path_use_clip_bool
\l__draw_path_use_fill_bool	674 \bool_new:N \l__draw_path_use_fill_bool
\l__draw_path_use_stroke_bool	

```
675 \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.)

\l__draw_path_use_clear_bool Actions handled at the macro layer.

```
676 \bool_new:N \l__draw_path_use_clear_bool
```

(End of definition for \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_replace_bb:
\__draw_path_replace_bb:NnN
  \__draw_path_use:n
    \__draw_path_use_action_draw:
    \__draw_path_use_action_fillstroke:
\__draw_path_use_stroke_bb:
  \__draw_path_use_bb:NnN
677 \cs_new_protected:Npn \draw_path_use:n #1
678 {
679   \tl_if_blank:nF {#1}
680     { \__draw_path_use:n {#1} }
681 }
682 \cs_new_protected:Npn \draw_path_use_clear:n #1
683 {
684   \bool_lazy_or:nnTF
685     { \tl_if_blank_p:n {#1} }
686     { \str_if_eq_p:nn {#1} { clear } }
687     {
688       \__draw_softpath_clear:
689       \__draw_path_reset_limits:
690     }
691   { \__draw_path_use:n { #1 , clear } }
692 }
693 \cs_new_protected:Npn \draw_path_replace_bb:
694 {
695   \__draw_path_replace_bb:NnN x { max } +
696   \__draw_path_replace_bb:NnN y { max } +
697   \__draw_path_replace_bb:NnN x { min } -
698   \__draw_path_replace_bb:NnN y { min } -
699   \__draw_softpath_clear:
700   \__draw_path_reset_limits:
701 }
702 \cs_new_protected:Npn \__draw_path_replace_bb:NnN #1#2#3
703 {
704   \dim_gset:cn { g_draw_bb_ #1#2 _dim }
705   {
706     \dim_use:c { g__draw_path_ #1#2 _dim }
707     #3 0.5 \g__draw_linewidth_dim
708   }
709 }

```

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).

```

710 \cs_new_protected:Npn \__draw_path_use:n #1
711 {
712   \bool_set_false:N \l__draw_path_use_clip_bool
713   \bool_set_false:N \l__draw_path_use_fill_bool
714   \bool_set_false:N \l__draw_path_use_stroke_bool

```

```

715 \clist_map_inline:nn {#1}
716 {
717     \cs_if_exist:CTF { l__draw_path_use_ ##1 _ bool }
718     { \bool_set_true:c { l__draw_path_use_ ##1 _ bool } }
719     {
720         \cs_if_exist_use:cF { __draw_path_use_action_ ##1 : }
721         { \msg_error:nnn { draw } { invalid-path-action } {##1} }
722     }
723 }
724 \__draw_softpath_round_corners:
725 \bool_lazy_and:nnT
726 { \l_draw_bb_update_bool }
727 { \l__draw_path_use_stroke_bool }
728 { \__draw_path_use_stroke_bb: }
729 \__draw_softpath_use:
730 \bool_if:NT \l__draw_path_use_clip_bool
731 {
732     \__draw_backend_clip:
733     \bool_set_false:N \l_draw_bb_update_bool
734     \bool_lazy_or:nnF
735     { \l__draw_path_use_fill_bool }
736     { \l__draw_path_use_stroke_bool }
737     { \__draw_backend_discardpath: }
738 }
739 \bool_lazy_or:nnT
740 { \l__draw_path_use_fill_bool }
741 { \l__draw_path_use_stroke_bool }
742 {
743     \use:c
744     {
745         __draw_backend_
746         \bool_if:NT \l__draw_path_use_fill_bool { fill }
747         \bool_if:NT \l__draw_path_use_stroke_bool { stroke }
748         :
749     }
750 }
751 \bool_if:NT \l__draw_path_use_clear_bool
752 {
753     \__draw_softpath_clear:
754     \__draw_path_reset_limits:
755 }
756 }
757 \cs_new_protected:Npn \__draw_path_use_action_draw:
758 {
759     \bool_set_true:N \l__draw_path_use_stroke_bool
760 }
761 \cs_new_protected:Npn \__draw_path_use_action_fillstroke:
762 {
763     \bool_set_true:N \l__draw_path_use_fill_bool
764     \bool_set_true:N \l__draw_path_use_stroke_bool
765 }

```

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.

```

766 \cs_new_protected:Npn \__draw_path_use_stroke_bb:
767 {
768   \__draw_path_use_bb:NnN x { max } +
769   \__draw_path_use_bb:NnN y { max } +
770   \__draw_path_use_bb:NnN x { min } -
771   \__draw_path_use_bb:NnN y { min } -
772 }
773 \cs_new_protected:Npn \__draw_path_use_bb:NnN #1#2#3
774 {
775   \dim_compare:nNnF { \dim_use:c { g_draw_bb_ #1#2 _dim } } = { #3 -\c_max_dim }
776   {
777     \dim_gset:cn { g_draw_bb_ #1#2 _dim }
778     {
779       \use:c { dim_ #2 :nn }
780       { \dim_use:c { g_draw_bb_ #1#2 _dim } }
781       {
782         \dim_use:c { g__draw_path_ #1#2 _dim }
783         #3 0.5 \g__draw_linewidth_dim
784       }
785     }
786   }
787 }

```

(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
788 \dim_new:N \l__draw_path_lastx_dim
789 \dim_new:N \l__draw_path_lasty_dim
790 \dim_new:N \l__draw_path_xmax_dim
791 \dim_new:N \l__draw_path_xmin_dim
792 \dim_new:N \l__draw_path_ymax_dim
793 \dim_new:N \l__draw_path_ymin_dim
794 \dim_new:N \l__draw_softpath_lastx_dim
795 \dim_new:N \l__draw_softpath_lasty_dim
796 \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:
797 \cs_new_protected:Npn \draw_path_scope_begin:
798 {
799   \group_begin:
800   \dim_set_eq:NN \l__draw_path_lastx_dim \g__draw_path_lastx_dim
801   \dim_set_eq:NN \l__draw_path_lasty_dim \g__draw_path_lasty_dim
802   \dim_set_eq:NN \l__draw_path_xmax_dim \g__draw_path_xmax_dim
803   \dim_set_eq:NN \l__draw_path_xmin_dim \g__draw_path_xmin_dim
804   \dim_set_eq:NN \l__draw_path_ymax_dim \g__draw_path_ymax_dim
805   \dim_set_eq:NN \l__draw_path_ymin_dim \g__draw_path_ymin_dim
806   \dim_set_eq:NN \l__draw_softpath_lastx_dim \g__draw_softpath_lastx_dim

```

```

807     \dim_set_eq:NN \l__draw_softpath_lasty_dim \g__draw_softpath_lasty_dim
808     \__draw_path_reset_limits:
809     \__draw_softpath_save:
810 }
811 \cs_new_protected:Npn \draw_path_scope_end:
812 {
813     \__draw_softpath_restore:
814     \dim_gset_eq:NN \g__draw_softpath_lastx_dim \l__draw_softpath_lastx_dim
815     \dim_gset_eq:NN \g__draw_softpath_lasty_dim \l__draw_softpath_lasty_dim
816     \dim_gset_eq:NN \g__draw_path_xmax_dim \l__draw_path_xmax_dim
817     \dim_gset_eq:NN \g__draw_path_xmin_dim \l__draw_path_xmin_dim
818     \dim_gset_eq:NN \g__draw_path_ymax_dim \l__draw_path_ymax_dim
819     \dim_gset_eq:NN \g__draw_path_ymin_dim \l__draw_path_ymin_dim
820     \dim_gset_eq:NN \g__draw_path_lastx_dim \l__draw_path_lastx_dim
821     \dim_gset_eq:NN \g__draw_path_lasty_dim \l__draw_path_lasty_dim
822     \group_end:
823 }

```

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

4.10 Messages

```

824 \msg_new:nnnn { draw } { invalid-path-action }
825 { Invalid-action-#1'-for-path. }
826 { Paths-can-be-used-with-actions-'draw',~'clip',~'fill'~or~'stroke'. }
827 % \end{macrocode}
828 %
829 % \begin{macrocode}
830 \end{package}

```

5 l3draw-points implementation

```

831 \*package
832 \@@=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 coordinate 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

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

```

\__draw_point_process:nn
  \__draw_point_process_auxi:nn
  \__draw_point_process_auxi:en
  \__draw_point_process_auxii:nw
\__draw_point_process:nnn
  \__draw_point_process_auxiii:nnn
  \__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
833 \cs_new:Npn \__draw_point_process:nn #1#2
834   { \__draw_point_process_auxi:en {#2} {#1} }
835 \cs_new:Npn \__draw_point_process_auxi:nn #1#2
836   { \__draw_point_process_auxii:nw {#2} #1 \s__draw_stop }
837 \cs_generate_variant:Nn \__draw_point_process_auxi:nn { e }
838 \cs_new:Npn \__draw_point_process_auxii:nw #1 #2 , #3 \s__draw_stop
839   { #1 {#2} {#3} }
840 \cs_new:Npn \__draw_point_process:nnn #1#2#3
841   { \__draw_point_process_auxiii:een {#2} {#3} {#1} }
842 \cs_new:Npn \__draw_point_process_auxiii:nnn #1#2#3
843   { \__draw_point_process_auxiv:nw {#3} #1 \s__draw_mark #2 \s__draw_stop }
844 \cs_generate_variant:Nn \__draw_point_process_auxiii:nnn { ee }
845 \cs_new:Npn \__draw_point_process_auxiv:nw #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_stop
846   { #1 {#2} {#3} {#4} {#5} }
847 \cs_new:Npn \__draw_point_process:nnnn #1#2#3#4
848   { \__draw_point_process_auxv:eeen {#2} {#3} {#4} {#1} }
849 \cs_new:Npn \__draw_point_process_auxv:nnnn #1#2#3#4
850   { \__draw_point_process_auxvi:nw {#4} #1 \s__draw_mark #2 \s__draw_mark #3 \s__draw_stop }
851 \cs_generate_variant:Nn \__draw_point_process_auxv:nnnn { eee }
852 \cs_new:Npn \__draw_point_process_auxvi:nw
853   #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_mark #6 , #7 \s__draw_stop
854   { #1 {#2} {#3} {#4} {#5} {#6} {#7} }
855 \cs_new:Npn \__draw_point_process:nnnnn #1#2#3#4#5
856   { \__draw_point_process_auxvii:eeeen {#2} {#3} {#4} {#5} {#1} }
857 \cs_new:Npn \__draw_point_process_auxvii:nnnnn #1#2#3#4#5
858   {
859     \__draw_point_process_auxviii:nw
860     {#5} #1 \s__draw_mark #2 \s__draw_mark #3 \s__draw_mark #4 \s__draw_stop
861   }
862 \cs_generate_variant:Nn \__draw_point_process_auxvii:nnnnn { eeee }
863 \cs_new:Npn \__draw_point_process_auxviii:nw
864   #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_mark #6 , #7 \s__draw_mark #8 , #9 \s__draw_stop
865   { #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` Coordinates are always returned as two dimensions.

```

\__draw_point_to_dim:n
\__draw_point_to_dim:e
\__draw_point_to_dim:w

```

```

866 \cs_new:Npn \draw_point:n #1
867 { \__draw_point_to_dim:e { \fp_eval:n {#1} } }
868 \cs_new:Npn \__draw_point_to_dim:n #1
869 { \__draw_point_to_dim:w #1 }
870 \cs_generate_variant:Nn \__draw_point_to_dim:n { e }
871 \cs_new:Npn \__draw_point_to_dim:w ( #1 , ~ #2 ) { #1pt , #2pt }

```

5.3 Polar coordinates

Polar coordinates 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
872 \cs_new:Npn \draw_point_polar:nn #1#2
873 { \draw_point_polar:nnn {#1} {#1} {#2} }
874 \cs_new:Npn \draw_point_polar:nnn #1#2#3
875 { \__draw_draw_polar:enn { \fp_eval:n {#3} } {#1} {#2} }
876 \cs_new:Npn \__draw_draw_polar:nnn #1#2#3
877 { \draw_point:n { cosd(#1) * (#2) , sind(#1) * (#3) } }
878 \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 normalized 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.

```

879 \cs_new:Npn \draw_point_unit_vector:n #1
880 { \__draw_point_process:nn { \__draw_point_unit_vector:nn } { \draw_point:n {#1} } }
881 \cs_new:Npn \__draw_point_unit_vector:nn #1#2
882 {
883   \__draw_point_unit_vector:nnn
884   { \fp_eval:n { (sqrt(#1 * #1 + #2 * #2)) } }
885   {#1} {#2}
886 }
887 \cs_new:Npn \__draw_point_unit_vector:nnn #1#2#3
888 {
889   \fp_compare:nNnTF {#1} = \c_zero_fp
890   { Opt, 1pt }
891   {
892     \draw_point:n
893     { ( #2 , #3 ) / #1 }
894   }
895 }
896 \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

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

and

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

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.)

```

897 \cs_new:Npn \draw_point_intersect_lines:nnnn #1#2#3#4
898 {
899   \__draw_point_process:nnnnn
900   { \__draw_point_intersect_lines:nnnnnnnn }
901   { \draw_point:n {#1} }
902   { \draw_point:n {#2} }
903   { \draw_point:n {#3} }
904   { \draw_point:n {#4} }
905 }
```

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.

```

906 \cs_new:Npn \__draw_point_intersect_lines:nnnnnnnn #1#2#3#4#5#6#7#8
907 {
908   \__draw_point_intersect_lines_aux:eeeeee
909   { \fp_eval:n { #1 * #4 - #2 * #3 } }
910   { \fp_eval:n { #5 * #8 - #6 * #7 } }
911   { \fp_eval:n { #1 - #3 } }
912   { \fp_eval:n { #5 - #7 } }
913   { \fp_eval:n { #2 - #4 } }
914   { \fp_eval:n { #6 - #8 } }
915 }
916 \cs_new:Npn \__draw_point_intersect_lines_aux:nnnnnn #1#2#3#4#5#6
917 {
918   \draw_point:n
```



```

919      {
920      ( #2 * #3 - #1 * #4 , #2 * #5 - #1 * #6 )
921      / ( #4 * #5 - #6 * #3 )
922      }
923    }
924 \cs_generate_variant:Nn \__draw_point_intersect_lines_aux:nnnnnn { eeeeeee }

```

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

```

\draw_point_intersect_circles:nnnnn
__draw_point_intersect_circles_auxi:nnnnnnn
__draw_point_intersect_circles_auxii:nnnnnnn
__draw_point_intersect_circles_auxiii:eennnnnn
draw_point_intersect_circles_auxiii:nnnnnnn
draw_point_intersect_circles_auxiii:eennnnnn
draw_point_intersect_circles_auxiv:nnnnnnnn
draw_point_intersect_circles_auxiv:ennnnnnnn
draw_point_intersect_circles_auxv:nnnnnnnnnn
draw_point_intersect_circles_auxv:eennnnnnnnn
draw_point_intersect_circles_auxvi:nnnnnnnnn
draw_point_intersect_circles_auxvi:ennnnnnnnn
draw_point_intersect_circles_auxvii:nnnnnnnn
draw_point_intersect_circles_auxvii:eeennnnn

```

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

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.

```

925 \cs_new:Npn \draw_point_intersect_circles:nnnnn #1#2#3#4#5
926 {
927   \__draw_point_process:nnn
928   { \__draw_point_intersect_circles_auxi:nnnnnnn {#2} {#4} {#5} }
929   { \draw_point:n {#1} }
930   { \draw_point:n {#3} }
931 }
932 \cs_new:Npn \__draw_point_intersect_circles_auxi:nnnnnnn #1#2#3#4#5#6#7
933 {
934   \__draw_point_intersect_circles_auxii:eennnnnn
935   { \fp_eval:n {#1} } { \fp_eval:n {#2} } {#4} {#5} {#6} {#7} {#3}
936 }

```

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 coordinate (c, d) is no longer required: handy as we will need various intermediate values in the following.

```
937 \cs_new:Npn \__draw_point_intersect_circles_auxii:nnnnnnn #1#2#3#4#5#6#7
938 {
939   \__draw_point_intersect_circles_auxiii:ennnnnnn
940   { \fp_eval:n { #5 - #3 } }
941   { \fp_eval:n { #6 - #4 } }
942   {#1} {#2} {#3} {#4} {#7}
943 }
944 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxii:nnnnnnn { ee }
945 \cs_new:Npn \__draw_point_intersect_circles_auxiii:nnnnnnn #1#2#3#4#5#6#7
946 {
947   \__draw_point_intersect_circles_auxiv:ennnnnnnn
948   { \fp_eval:n { sqrt( #1 * #1 + #2 * #2 ) } }
949   {#1} {#2} {#3} {#4} {#5} {#6} {#7}
950 }
951 \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.

```
952 \cs_new:Npn \__draw_point_intersect_circles_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
953 {
954   \__draw_point_intersect_circles_auxv:ennnnnnnnn
955   { \fp_eval:n { 1 / #1 } }
956   { \fp_eval:n { #4 * #4 } }
957   {#1} {#2} {#3} {#5} {#6} {#7} {#8}
958 }
959 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxiv:nnnnnnnn { e }
960 \cs_new:Npn \__draw_point_intersect_circles_auxv:nnnnnnnnn #1#2#3#4#5#6#7#8#9
961 {
962   \__draw_point_intersect_circles_auxvi:ennnnnnnnn
963   { \fp_eval:n { 0.5 * #1 * ( #2 + #3 * #3 - #6 * #6 ) } }
964   {#1} {#2} {#4} {#5} {#7} {#8} {#9}
965 }
966 \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.

```

967 \cs_new:Npn \__draw_point_intersect_circles_auxvi:nnnnnnnn #1#2#3#4#5#6#7#8
968 {
969   \__draw_point_intersect_circles_auxvii:eeennnnn
970   { \fp_eval:n { #1 * #2 } }
971   { \int_if_odd:nTF {#8} { 1 } { -1 } }
972   { \fp_eval:n { sqrt ( #3 - #1 * #1 ) * #2 } }
973   {#4} {#5} {#6} {#7}
974 }
975 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxvi:nnnnnnnn { e }
976 \cs_new:Npn \__draw_point_intersect_circles_auxvii:nnnnnnnn #1#2#3#4#5#6#7
977 {
978   \draw_point:n
979   { #6 + #4 * #1 + #2 * #3 * #5 , #7 + #5 * #1 + -1 * #2 * #3 * #4 }
980 }
981 \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.

```

982 \cs_new:Npn \draw_point_intersect_line_circle:nnnnn #1#2#3#4#5
983 {
984   \__draw_point_process:nnnnn
985   { \__draw_point_intersect_line_circle_auxi:nnnnnnnn {#4} {#5} }
986   { \draw_point:n {#1} }

```

```

987     { \draw_point:n {#2} }
988     { \draw_point:n {#3} }
989   }
990   \cs_new:Npn \__draw_point_intersect_line_circle_auxi:nnnnnnnn #1#2#3#4#5#6#7#8
991   {
992     \__draw_point_intersect_line_circle_auxii:ennnnnnnn
993     { \fp_eval:n {#1} } {#3} {#4} {#5} {#6} {#7} {#8} {#2}
994   }

```

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

```

#1 r
#2 x_1
#3 y_1
#4 x_2
#5 y_2
#6 x_3
#7 y_3
#8 n

```

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

```

995   \cs_new:Npn \__draw_point_intersect_line_circle_auxii:nnnnnnnn #1#2#3#4#5#6#7#8
996   {
997     \__draw_point_intersect_line_circle_auxiii:eeennnnnn
998     { \fp_eval:n { (#4-#2)*(#4-#2)+(#5-#3)*(#5-#3) } }
999     { \fp_eval:n { 2*((#4-#2)*(#2-#6)+(#5-#3)*(#3-#7)) } }
1000     { \fp_eval:n { (#6*#6+#7*#7)+(#2*#2+#3*#3)-(2*(#6*#2+#7*#3))-(#1*#1) } }
1001     {#2} {#3} {#4} {#5} {#6} {#7} {#8}
1002   }
1003   \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 .

```

1004   \cs_new:Npn \__draw_point_intersect_line_circle_auxiii:nnnnnnnn #1#2#3#4#5#6#7#8
1005   {
1006     \__draw_point_intersect_line_circle_auxiv:eeennnnnn
1007     { \fp_eval:n { #2 * #2 - 4 * #1 * #3 } }
1008     { \int_if_odd:nTF {#8} { 1 } { -1 } }
1009     {#1} {#2} {#4} {#5} {#6} {#7}
1010   }
1011   \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 \pm (the usage of n)

#6 x_1

#7 y_1

#8 x_2

#9 y_2

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

```

1012 \cs_new:Npn \__draw_point_intersect_line_circle_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
1013 {
1014   \__draw_point_intersect_line_circle_auxv:ennnn
1015   { \fp_eval:n { (-1 * #4 + #2 * sqrt(#1)) / (2 * #3) } }
1016   {#5} {#6} {#7} {#8}
1017 }
1018 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxiv:nnnnnnnn { ee }
1019 \cs_new:Npn \__draw_point_intersect_line_circle_auxv:nnnnn #1#2#3#4#5
1020 {
1021   \draw_point:n
1022   { #2 + #1 * (#4 - #2), #3 + #1 * (#5 - #3) }
1023 }
1024 \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
\_draw_point_interpolate_line_aux:nnnnn
\_draw_point_interpolate_line_aux:ennnn
\_draw_point_interpolate_line_aux:nnnnnn
\_draw_point_interpolate_line_aux:ennnnn
1025 \cs_new:Npn \draw_point_interpolate_line:nnn #1#2#3
1026 {
1027   \__draw_point_process:nnn
1028   { \__draw_point_interpolate_line_aux:ennnn { \fp_eval:n {#1} } }
1029   { \draw_point:n {#2} }
1030   { \draw_point:n {#3} }
1031 }
1032 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnn #1#2#3#4#5
1033 {
1034   \__draw_point_interpolate_line_aux:ennnnn { \fp_eval:n { 1 - #1 } }
1035   {#1} {#2} {#3} {#4} {#5}
1036 }
1037 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnn { e }
1038 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnnn #1#2#3#4#5#6
1039 { \draw_point:n { #2 * #3 + #1 * #5 , #2 * #4 + #1 * #6 } }
1040 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnnn { e }

```

Same idea but using the normalized 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
\_draw_point_interpolate_distance:nnnnn
\_draw_point_interpolate_distance:nnnnnn
\_draw_point_interpolate_distance:ennnnn
1041 \cs_new:Npn \draw_point_interpolate_distance:nnn #1#2#3
1042 {
1043   \__draw_point_process:nn
1044   { \__draw_point_interpolate_distance:nnnn {#1} {#3} }
1045   {#2}

```

```

1046 }
1047 \cs_new:Npn \__draw_point_interpolate_distance:nnnn #1#2#3#4
1048 {
1049   \__draw_point_process:nn
1050   {
1051     \__draw_point_interpolate_distance:ennnn
1052     { \fp_eval:n {#1} } {#3} {#4}
1053   }
1054   { \draw_point_unit_vector:n { ( #2 ) - ( #3 , #4 ) } }
1055 }
1056 \cs_new:Npn \__draw_point_interpolate_distance:nnnnn #1#2#3#4#5
1057 { \draw_point:n { #2 + #1 * #4 , #3 + #1 * #5 } }
1058 \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:ennnnnnnnnn

```

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 coordinate expansion.

```

1059 \cs_new:Npn \draw_point_interpolate_arcaxes:nnnnnn #1#2#3#4#5#6
1060 {
1061   \__draw_point_process:nnnn
1062   { \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn {#1} {#5} {#6} }
1063   { \draw_point:n {#2} }
1064   { \draw_point:n {#3} }
1065   { \draw_point:n {#4} }
1066 }
1067 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
1068 {
1069   \__draw_point_interpolate_arcaxes_auxii:ennnnnnnnnn
1070   { \fp_eval:n {#1} } {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1071 }

```

At this stage, the three coordinate 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.

```

1072 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
1073 {

```

```

1074     \_draw_point_interpolate_arcaxes_auxiii:ennnnnnn
1075     { \fp_eval:n { #1 * (#3) + ( 1 - #1 ) * (#2) } }
1076     {#4} {#5} {#6} {#7} {#8} {#9}
1077   }
1078   \cs_generate_variant:Nn \_draw_point_interpolate_arcaxes_auxii:nnnnnnnnnn { e }
1079   \cs_new:Npn \_draw_point_interpolate_arcaxes_auxiii:nnnnnnnn #1#2#3#4#5#6#7
1080   {
1081     \_draw_point_interpolate_arcaxes_auxiv:ennnnnnnnn
1082     { \fp_eval:n { cosd (#1) } }
1083     { \fp_eval:n { sind (#1) } }
1084     {#2} {#3} {#4} {#5} {#6} {#7}
1085   }
1086   \cs_generate_variant:Nn \_draw_point_interpolate_arcaxes_auxiii:nnnnnnnn { e }
1087   \cs_new:Npn \_draw_point_interpolate_arcaxes_auxiv:nnnnnnnnnn #1#2#3#4#5#6#7#8
1088   {
1089     \draw_point:n
1090     { #3 + #1 * #5 + #2 * #7 , #4 + #1 * #6 + #2 * #8 }
1091   }
1092   \cs_generate_variant:Nn \_draw_point_interpolate_arcaxes_auxiv:nnnnnnnnnn { ee }

```

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

```

\_draw_point_interpolate_curve:nnnnnn
draw_point_interpolate_curve_auxi:nnnnnnnnnn
draw_point_interpolate_curve_auxii:nnnnnnnnnn
draw_point_interpolate_curve_auxiii:ennnnnnnnn
\_draw_point_interpolate_curve_auxiiii:nnnnnnnn
\_draw_point_interpolate_curve_auxiii:ennnnnnn
\_draw_point_interpolate_curve_auxiv:nnnnnnnn
\_draw_point_interpolate_curve_auxv:nnw
\_draw_point_interpolate_curve_auxv:eev
\_draw_point_interpolate_curve_auxvi:n
draw_point_interpolate_curve_auxvii:nnnnnnnnnn
draw_point_interpolate_curve_auxviii:nnnnnnnn
draw_point_interpolate_curve_auxviii:ennnnnnn

```

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.

```

1093   \cs_new:Npn \draw_point_interpolate_curve:nnnnnn #1#2#3#4#5
1094   {
1095     \_draw_point_process:nnnnnn
1096     { \_draw_point_interpolate_curve_auxi:nnnnnnnnnn {#1} }
1097     { \draw_point:n {#2} }
1098     { \draw_point:n {#3} }
1099     { \draw_point:n {#4} }
1100     { \draw_point:n {#5} }
1101   }
1102   \cs_new:Npn \_draw_point_interpolate_curve_auxi:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
1103   {
1104     \_draw_point_interpolate_curve_auxii:ennnnnnnnn
1105     { \fp_eval:n {#1} }
1106     {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1107   }

```

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 coordinates

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

In the second stage, we can drop the final point

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

and for the final stage only need one set of calculations

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

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

```

1108 \cs_new:Npn \__draw_point_interpolate_curve_auxii:nnnnnnnnn
1109   #1#2#3#4#5#6#7#8#9
1110   {
1111     \__draw_point_interpolate_curve_auxiii:ennnnnn
1112     { \fp_eval:n { 1 - #1 } }
1113     {#1}
1114     { {#2} {#3} } { {#4} {#5} } { {#6} {#7} } { {#8} {#9} }
1115   }
1116 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxii:nnnnnnnnn { e }
1117 % \begin{macrocode}
1118 % We need to do the first cycle, but haven't got enough arguments to keep
1119 % everything in play at once. So here we use a bit of argument re-ordering
1120 % and a single auxiliary to get the job done.
1121 % \begin{macrocode}
1122 \cs_new:Npn \__draw_point_interpolate_curve_auxiii:nnnnnn #1#2#3#4#5#6
1123   {
1124     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #3 #4
1125     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #4 #5
1126     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #5 #6
1127     \prg_do_nothing:
1128     \__draw_point_interpolate_curve_auxvi:n { {#1} {#2} }
1129   }
1130 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxiii:nnnnnn { e }
1131 \cs_new:Npn \__draw_point_interpolate_curve_auxiv:nnnnnn #1#2#3#4#5#6
1132   {
1133     \__draw_point_interpolate_curve_auxv:eev
1134     { \fp_eval:n { #1 * #3 + #2 * #5 } }
1135     { \fp_eval:n { #1 * #4 + #2 * #6 } }
1136   }

```



```

1137 \cs_new:Npn \__draw_point_interpolate_curve_auxv:nnw
1138   #1#2#3 \prg_do_nothing: #4#5
1139   {
1140     #3
1141     \prg_do_nothing:
1142     #4 { #5 {#1} {#2} }
1143   }
1144 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxv:nnw { ee }
1145 % \begin{macrocode}
1146 % Get the arguments back into the right places and to the second and
1147 % third cycles directly.
1148 % \begin{macrocode}
1149 \cs_new:Npn \__draw_point_interpolate_curve_auxvi:n #1
1150   { \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1 }
1151 \cs_new:Npn \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1#2#3#4#5#6#7#8
1152   {
1153     \__draw_point_interpolate_curve_auxviii:eeeeenn
1154     { \fp_eval:n { #1 * #5 + #2 * #3 } }
1155     { \fp_eval:n { #1 * #6 + #2 * #4 } }
1156     { \fp_eval:n { #1 * #7 + #2 * #5 } }
1157     { \fp_eval:n { #1 * #8 + #2 * #6 } }
1158     {#1} {#2}
1159   }
1160 \cs_new:Npn \__draw_point_interpolate_curve_auxviii:nnnnnn #1#2#3#4#5#6
1161   {
1162     \draw_point:n
1163     { #5 * #3 + #6 * #1 , #5 * #4 + #6 * #2 }
1164   }
1165 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxviii:nnnnnn { eeee }

```

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

5.7 Vector support

As well as coordinates 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

```

```

1172 \cs_new_protected:Npn \draw_xvec:n #1
1173   { \__draw_vec:nn { x } {#1} }
1174 \cs_new_protected:Npn \draw_yvec:n #1
1175   { \__draw_vec:nn { y } {#1} }
1176 \cs_new_protected:Npn \draw_zvec:n #1
1177   { \__draw_vec:nn { z } {#1} }

```

```

1178 \cs_new_protected:Npn \__draw_vec:nn #1#2
1179 { \__draw_point_process:nn { \__draw_vec:nnn {#1} } { \draw_point:n {#2} } }
1180 \cs_new_protected:Npn \__draw_vec:nnn #1#2#3
1181 {
1182   \dim_set:cn { l__draw_ #1 vec_x_dim } {#2}
1183   \dim_set:cn { l__draw_ #1 vec_y_dim } {#3}
1184 }

```

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

Initialize the vectors.

```

1185 \draw_xvec:n { 1cm , 0cm }
1186 \draw_yvec:n { 0cm , 1cm }
1187 \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 1188 \cs_new:Npn \draw_point_vec:nn #1#2
\__draw_point_vec:ee 1189 { \__draw_point_vec:ee { \fp_eval:n {#1} } { \fp_eval:n {#2} } }
\draw_point_vec:nnn 1190 \cs_new:Npn \__draw_point_vec:nn #1#2
\__draw_point_vec:nnn 1191 {
\__draw_point_vec:eee 1192   \draw_point:n
1193   {
1194     #1 * \l__draw_xvec_x_dim + #2 * \l__draw_yvec_x_dim ,
1195     #1 * \l__draw_xvec_y_dim + #2 * \l__draw_yvec_y_dim
1196   }
1197 }
1198 \cs_generate_variant:Nn \__draw_point_vec:nn { ee }
1199 \cs_new:Npn \draw_point_vec:nnn #1#2#3
1200 {
1201   \__draw_point_vec:eee
1202   { \fp_eval:n {#1} } { \fp_eval:n {#2} } { \fp_eval:n {#3} }
1203 }
1204 \cs_new:Npn \__draw_point_vec:nnn #1#2#3
1205 {
1206   \draw_point:n
1207   {
1208     #1 * \l__draw_xvec_x_dim
1209     + #2 * \l__draw_yvec_x_dim
1210     + #3 * \l__draw_zvec_x_dim
1211     ,
1212     #1 * \l__draw_xvec_y_dim
1213     + #2 * \l__draw_yvec_y_dim
1214     + #3 * \l__draw_zvec_y_dim
1215   }
1216 }
1217 \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 Much the same as the core polar approach.

```

\draw_point_vec_polar:nnn 1218 \cs_new:Npn \draw_point_vec_polar:nn #1#2
\__draw_point_vec_polar:nnn 1219 { \draw_point_vec_polar:nnn {#1} {#1} {#2} }
\__draw_point_vec_polar:enn 1220 \cs_new:Npn \draw_point_vec_polar:nnn #1#2#3
1221 { \__draw_draw_vec_polar:enn { \fp_eval:n {#3} } {#1} {#2} }
1222 \cs_new:Npn \__draw_draw_vec_polar:nnn #1#2#3

```

```

1223 {
1224   \draw_point:n
1225   {
1226     cosd(#1) * (#2) * \l__draw_xvec_x_dim ,
1227     sind(#1) * (#3) * \l__draw_yvec_y_dim
1228   }
1229 }
1230 \cs_generate_variant:Nn \__draw_draw_vec_polar:nnn { e }

```

(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.

```

\__draw_point_transform:nn
1231 \cs_new:Npn \draw_point_transform:n #1
1232 {
1233   \__draw_point_process:nn
1234   { \__draw_point_transform:nn } { \draw_point:n {#1} }
1235 }
1236 \cs_new:Npn \__draw_point_transform:nn #1#2
1237 {
1238   \bool_if:NTF \l__draw_matrix_active_bool
1239   {
1240     \draw_point:n
1241     {
1242       (
1243         \l__draw_matrix_a_fp * #1
1244         + \l__draw_matrix_c_fp * #2
1245         + \l__draw_xshift_dim
1246       )
1247       ,
1248       (
1249         \l__draw_matrix_b_fp * #1
1250         + \l__draw_matrix_d_fp * #2
1251         + \l__draw_yshift_dim
1252       )
1253     }
1254   }
1255   {
1256     \draw_point:n
1257     {
1258       (#1, #2)
1259       + ( \l__draw_xshift_dim , \l__draw_yshift_dim )
1260     }
1261   }
1262 }

```

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

`__draw_point_transform_noshift:n` A version with no shift: used for internal purposes.

```

\__draw_point_transform_noshift:nn
1263 \cs_new:Npn \__draw_point_transform_noshift:n #1

```

```

1264 {
1265   \__draw_point_process:nn
1266   { \__draw_point_transform_noshift:nn }
1267   { \draw_point:n {#1} }
1268 }
1269 \cs_new:Npn \__draw_point_transform_noshift:nn #1#2
1270 {
1271   \bool_if:NTF \l__draw_matrix_active_bool
1272   {
1273     \draw_point:n
1274     {
1275       (
1276         \l__draw_matrix_a_fp * #1
1277         + \l__draw_matrix_c_fp * #2
1278       )
1279       ,
1280       (
1281         \l__draw_matrix_b_fp * #1
1282         + \l__draw_matrix_d_fp * #2
1283       )
1284     }
1285   }
1286   { \draw_point:n { (#1, #2) } }
1287 }

```

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

```
1288 \endpackage
```

6 l3draw-scopes implementation

```
1289 \begin{package}
```

```
1290 \begin{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

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

```

\g_draw_bb_xmin_dim
\g_draw_bb_ymax_dim
\g_draw_bb_ymin_dim
1291 \dim_new:N \g_draw_bb_xmax_dim
1292 \dim_new:N \g_draw_bb_xmin_dim
1293 \dim_new:N \g_draw_bb_ymax_dim
1294 \dim_new:N \g_draw_bb_ymin_dim

```

(End of definition for `\g_draw_bb_xmax_dim` and others. These variables are documented on page ??.)

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

```
1295 \bool_new:N \l_draw_bb_update_bool
```

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

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

```
1296 \box_new:N \l__draw_main_box
1297 \box_new:N \l__draw_layer_main_box

(End of definition for \l__draw_layer_main_box.)
```

`\g_draw_id_int` The drawing number.

```
1298 \int_new:N \g_draw_id_int

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

`__draw_reset_bb:` A simple auxiliary.

```
1299 \cs_new_protected:Npn \__draw_reset_bb:
1300 {
1301   \dim_gset:Nn \g_draw_bb_xmax_dim { -\c_max_dim }
1302   \dim_gset:Nn \g_draw_bb_xmin_dim { \c_max_dim }
1303   \dim_gset:Nn \g_draw_bb_ymax_dim { -\c_max_dim }
1304   \dim_gset:Nn \g_draw_bb_ymin_dim { \c_max_dim }
1305 }

(End of definition for \__draw_reset_bb:.)
```

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

```
1306 \cs_new_protected:Npn \draw_begin:
1307 {
1308   \group_begin:
1309   \int_gincr:N \g_draw_id_int
1310   \hbox_set:Nw \l__draw_main_box
1311   \__draw_backend_begin:
1312   \__draw_reset_bb:
1313   \__draw_path_reset_limits:
1314   \bool_set_true:N \l_draw_bb_update_bool
1315   \draw_transform_matrix_reset:
1316   \draw_transform_shift_reset:
1317   \__draw_softpath_clear:
1318   \draw_set_linewidth:n { \l_draw_default_linewidth_dim }
1319   \color_ensure_current:
1320   \draw_set_nonzero_rule:
1321   \draw_set_cap_but:
1322   \draw_set_join_miter:
1323   \draw_set_miterlimit:n { 10 }
1324   \draw_set_dash_pattern:nn { } { 0cm }
1325   \hbox_set:Nw \l__draw_layer_main_box
1326   \__draw_record_origin:
1327 }
1328 \cs_new_protected:Npn \draw_end:
1329 {
1330   \__draw_baseline_finalize:w
1331   \exp_args:NNNV \hbox_set_end:
```

```

1332         \clist_set:Nn \l_draw_layers_clist \l_draw_layers_clist
1333         \__draw_layers_insert:
1334         \__draw_backend_end:
1335         \hbox_set_end:
1336         \dim_compare:nNnT \g_draw_bb_xmin_dim = \c_max_dim
1337         {
1338             \dim_gzero:N \g_draw_bb_xmax_dim
1339             \dim_gzero:N \g_draw_bb_xmin_dim
1340             \dim_gzero:N \g_draw_bb_ymax_dim
1341             \dim_gzero:N \g_draw_bb_ymin_dim
1342         }
1343         \__draw_finalize:
1344         \box_set_wd:Nn \l__draw_main_box
1345         { \g_draw_bb_xmax_dim - \g_draw_bb_xmin_dim }
1346         \mode_leave_vertical:
1347         \box_use_drop:N \l__draw_main_box
1348         \group_end:
1349     }

```

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

`__draw_record_origin:` Used to log the absolute location of a drawing. Ideally this would not need two `\savepos:` we need to sort an “always left-to-right” box. At present, this functionality is only available in L^AT_EX.

```

1350 \cs_new_protected:Npe \__draw_record_origin:
1351 {
1352     \hbox_to_wd:nn { Opt }
1353     {
1354         \tex_savepos:D
1355         \cs_if_exist:NT \@expl@finalise@setup@@
1356         {
1357             \exp_not:N \property_record:en
1358             { draw . \exp_not:N \int_use:N \exp_not:N \g_draw_id_int }
1359             { xpos , ypos , abspage }
1360         }
1361         \tex_savepos:D
1362     }
1363 }
1364 \cs_generate_variant:Nn \property_record:nn { e }

```

(End of definition for `__draw_record_origin:`.)

`__draw_finalize:` Finalizing the (vertical) size of the output depends on whether we have an explicit baseline 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.

```

1365 \cs_new_protected:Npn \__draw_finalize:
1366 {
1367     \hbox_set:Nn \l__draw_main_box
1368     {
1369         \skip_horizontal:n { -\g_draw_bb_xmin_dim }
1370         \box_move_down:nn
1371         { \g_draw_bb_ymin_dim }

```

```

1372         { \box_use_drop:N \l__draw_main_box }
1373     }
1374     \box_set_dp:Nn \l__draw_main_box { Opt }
1375     \box_set_ht:Nn \l__draw_main_box
1376         { \g_draw_bb_ymax_dim - \g_draw_bb_ymin_dim }
1377 }
1378 \cs_new_protected:Npn \__draw_finalize_baseline:n #1
1379 {
1380     \hbox_set:Nn \l__draw_main_box
1381     {
1382         \skip_horizontal:n { -\g_draw_bb_xmin_dim }
1383         \box_move_down:nn
1384             {#1}
1385         { \box_use_drop:N \l__draw_main_box }
1386     }
1387     \box_set_dp:Nn \l__draw_main_box
1388     {
1389         \dim_max:nn
1390             { #1 - \g_draw_bb_ymin_dim }
1391             { Opt }
1392     }
1393     \box_set_ht:Nn \l__draw_main_box
1394         { \g_draw_bb_ymax_dim - #1 }
1395 }

```

(End of definition for __draw_finalize: and __draw_finalize_baseline:n.)

6.2 Baseline position

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

```

\l__draw_baseline_dim
1396 \bool_new:N \l__draw_baseline_bool
1397 \dim_new:N \l__draw_baseline_dim

```

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

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

```

1398 \cs_new_protected:Npn \draw_set_baseline:n #1
1399 {
1400     \bool_set_true:N \l__draw_baseline_bool
1401     \dim_set:Nn \l__draw_baseline_dim { \fp_to_dim:n {#1} }
1402 }

```

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

__draw_baseline_finalize: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.

```

1403 \cs_new_protected:Npn \__draw_baseline_finalize:w #1 \__draw_finalize:
1404 {
1405     \bool_if:NTF \l__draw_baseline_bool
1406     {
1407         \use:e
1408         {
1409             \exp_not:n {#1}
1410             \__draw_finalize_baseline:n { \dim_use:N \l__draw_baseline_dim }

```

```

1411     }
1412   }
1413   { #1 \_draw_finalize: }
1414 }

```

(End of definition for _draw_baseline_finalize:w.)

6.3 Scopes

\l_draw_linewidth_dim Storage for local variables.

```

\l\_draw_fill_color_tl 1415 \dim_new:N \l\_draw_linewidth_dim
\l\_draw_stroke_color_tl 1416 \tl_new:N \l\_draw_fill_color_tl
1417 \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 T_EX) scope, also deal with global data structures.

```

\draw_scope_begin: 1418 \cs_new_protected:Npn \draw_scope_begin:
1419 {
1420   \_draw_backend_scope_begin:
1421   \group_begin:
1422     \dim_set_eq:NN \l\_draw_linewidth_dim \g\_draw_linewidth_dim
1423     \draw_path_scope_begin:
1424   }
1425 \cs_new_protected:Npn \draw_scope_end:
1426 {
1427   \draw_path_scope_end:
1428   \dim_gset_eq:NN \g\_draw_linewidth_dim \l\_draw_linewidth_dim
1429   \group_end:
1430   \_draw_backend_scope_end:
1431 }

```

(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 1432 \dim_new:N \l\_draw_xmax_dim
\l\_draw_ymax_dim 1433 \dim_new:N \l\_draw_xmin_dim
\l\_draw_ymin_dim 1434 \dim_new:N \l\_draw_ymax_dim
1435 \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: 1436 \cs_new_protected:Npn \_draw_scope_bb_begin:
1437 {
1438   \group_begin:
1439     \dim_set_eq:NN \l\_draw_xmax_dim \g\_draw_bb_xmax_dim
1440     \dim_set_eq:NN \l\_draw_xmin_dim \g\_draw_bb_xmin_dim
1441     \dim_set_eq:NN \l\_draw_ymax_dim \g\_draw_bb_ymax_dim
1442     \dim_set_eq:NN \l\_draw_ymin_dim \g\_draw_bb_ymin_dim
1443     \_draw_reset_bb:
1444   }
1445 \cs_new_protected:Npn \_draw_scope_bb_end:
1446 {

```



```

1447         \dim_gset_eq:NN \g_draw_bb_xmax_dim \l__draw_xmax_dim
1448         \dim_gset_eq:NN \g_draw_bb_xmin_dim \l__draw_xmin_dim
1449         \dim_gset_eq:NN \g_draw_bb_ymax_dim \l__draw_ymax_dim
1450         \dim_gset_eq:NN \g_draw_bb_ymin_dim \l__draw_ymin_dim
1451     \group_end:
1452 }

```

(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:
1453 \cs_new_protected:Npn \draw_suspend_begin:
1454 {
1455     \__draw_scope_bb_begin:
1456     \draw_path_scope_begin:
1457     \draw_transform_matrix_reset:
1458     \draw_transform_shift_reset:
1459     \__draw_layers_save:
1460 }
1461 \cs_new_protected:Npn \draw_suspend_end:
1462 {
1463     \__draw_layers_restore:
1464     \draw_path_scope_end:
1465     \__draw_scope_bb_end:
1466 }

```

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

```

1467 </package>

```

7 l3draw-softpath implementation

```

1468 <*package>
1469 <@@=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.

```

1470 \tl_new:N \g__draw_softpath_main_tl

```

(End of definition for `\g__draw_softpath_main_tl.`)

```

\l__draw_softpath_tmp_tl Scratch space.
1471 \tl_new:N \l__draw_softpath_tmp_tl

(End of definition for \l__draw_softpath_tmp_tl.)

\g__draw_softpath_corners_bool Allow for optimized path use.
1472 \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 1473 \cs_new_protected:Npn \__draw_softpath_add:n
\__draw_softpath_add:e 1474 { \tl_build_gput_right:Nn \g__draw_softpath_main_tl }
1475 \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: 1476 \cs_new_protected:Npn \__draw_softpath_use:
1477 {
1478   \tl_build_get_intermediate:NN
1479   \g__draw_softpath_main_tl
1480   \l__draw_softpath_tmp_tl
1481   \l__draw_softpath_tmp_tl
1482 }
1483 \cs_new_protected:Npn \__draw_softpath_clear:
1484 {
1485   \tl_build_gbegin:N \g__draw_softpath_main_tl
1486   \bool_gset_false:N \g__draw_softpath_corners_bool
1487 }

(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: 1488 \cs_new_protected:Npn \__draw_softpath_save:
1489 {
1490   \tl_build_gend:N \g__draw_softpath_main_tl
1491   \tl_set_eq:NN
1492   \l__draw_softpath_main_tl
1493   \g__draw_softpath_main_tl
1494   \bool_set_eq:NN
1495   \l__draw_softpath_corners_bool
1496   \g__draw_softpath_corners_bool
1497   \__draw_softpath_clear:
1498 }
1499 \cs_new_protected:Npn \__draw_softpath_restore:
1500 {
1501   \__draw_softpath_clear:
1502   \__draw_softpath_add:o \l__draw_softpath_main_tl
1503   \bool_gset_eq:NN
1504   \g__draw_softpath_corners_bool
1505   \l__draw_softpath_corners_bool
1506 }

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

```

\g__draw_softpath_lastx_dim \g__draw_softpath_lasty_dim	For tracking the end of the path (to close it). 1507 \dim_new:N \g__draw_softpath_lastx_dim 1508 \dim_new:N \g__draw_softpath_lasty_dim <i>(End of definition for \g__draw_softpath_lastx_dim and \g__draw_softpath_lasty_dim.)</i>
\g__draw_softpath_move_bool	Track if moving a point should update the close position. 1509 \bool_new:N \g__draw_softpath_move_bool 1510 \bool_gset_true:N \g__draw_softpath_move_bool <i>(End of definition for \g__draw_softpath_move_bool.)</i>
__draw_softpath_closepath: __draw_softpath_curveto:nnnnnn __draw_softpath_lineto:nn __draw_softpath_moveto:nn __draw_softpath_rectangle:nnnn __draw_softpath_roundpoint:nn __draw_softpath_roundpoint:VV	The various parts of a path expressed as the appropriate soft path functions. 1511 \cs_new_protected:Npn __draw_softpath_closepath: 1512 { 1513 __draw_softpath_add:e 1514 { 1515 __draw_softpath_close_op:nn 1516 { \dim_use:N \g__draw_softpath_lastx_dim } 1517 { \dim_use:N \g__draw_softpath_lasty_dim } 1518 } 1519 } 1520 \cs_new_protected:Npn __draw_softpath_curveto:nnnnnn #1#2#3#4#5#6 1521 { 1522 __draw_softpath_add:n 1523 { 1524 __draw_softpath_curveto_opi:nn {#1} {#2} 1525 __draw_softpath_curveto_opii:nn {#3} {#4} 1526 __draw_softpath_curveto_opiii:nn {#5} {#6} 1527 } 1528 } 1529 \cs_new_protected:Npn __draw_softpath_lineto:nn #1#2 1530 { 1531 __draw_softpath_add:n 1532 { __draw_softpath_lineto_op:nn {#1} {#2} } 1533 } 1534 \cs_new_protected:Npn __draw_softpath_moveto:nn #1#2 1535 { 1536 __draw_softpath_add:n 1537 { __draw_softpath_moveto_op:nn {#1} {#2} } 1538 \bool_if:NT \g__draw_softpath_move_bool 1539 { 1540 \dim_gset:Nn \g__draw_softpath_lastx_dim {#1} 1541 \dim_gset:Nn \g__draw_softpath_lasty_dim {#2} 1542 } 1543 } 1544 \cs_new_protected:Npn __draw_softpath_rectangle:nnnn #1#2#3#4 1545 { 1546 __draw_softpath_add:n 1547 { 1548 __draw_softpath_rectangle_opi:nn {#1} {#2} 1549 __draw_softpath_rectangle_opii:nn {#3} {#4} 1550 } 1551 }

```

1552 \cs_new_protected:Npn \__draw_softpath_roundpoint:nn #1#2
1553 {
1554   \__draw_softpath_add:n
1555   { \__draw_softpath_roundpoint_op:nn {#1} {#2} }
1556   \bool_gset_true:N \g__draw_softpath_corners_bool
1557 }
1558 \cs_generate_variant:Nn \__draw_softpath_roundpoint:nn { VV }

```

(End of definition for __draw_softpath_closepath: 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.

```

1559 \cs_new_protected:Npn \__draw_softpath_close_op:nn #1#2
1560 { \__draw_backend_closepath: }
1561 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nn #1#2
1562 { \__draw_softpath_curveto_opi:nnNnnNnn {#1} {#2} }
1563 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nnNnnNnn #1#2#3#4#5#6#7#8
1564 { \__draw_backend_curveto:nnnnnn {#1} {#2} {#4} {#5} {#7} {#8} }
1565 \cs_new_protected:Npn \__draw_softpath_curveto_opii:nn #1#2
1566 { \__draw_softpath_curveto_opii:nn }
1567 \cs_new_protected:Npn \__draw_softpath_curveto_opiii:nn #1#2
1568 { \__draw_softpath_curveto_opiii:nn }
1569 \cs_new_protected:Npn \__draw_softpath_lineto_op:nn #1#2
1570 { \__draw_backend_lineto:nn {#1} {#2} }
1571 \cs_new_protected:Npn \__draw_softpath_moveto_op:nn #1#2
1572 { \__draw_backend_moveto:nn {#1} {#2} }
1573 \cs_new_protected:Npn \__draw_softpath_roundpoint_op:nn #1#2
1574 { \__draw_softpath_roundpoint_op:nn }
1575 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nn #1#2
1576 { \__draw_softpath_rectangle_opi:nnNnn {#1} {#2} }
1577 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nnNnn #1#2#3#4#5
1578 { \__draw_backend_rectangle:nnnn {#1} {#2} {#4} {#5} }
1579 \cs_new_protected:Npn \__draw_softpath_rectangle_opii:nn #1#2
1580 { \__draw_softpath_rectangle_opii:nn }

```

(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.

```

1581 \tl_new:N \l__draw_softpath_main_tl

```

(End of definition for \l__draw_softpath_main_tl.)

\l__draw_softpath_part_tl Data structures.

```

1582 \tl_new:N \l__draw_softpath_part_tl
1583 \tl_new:N \l__draw_softpath_curve_end_tl

```

(End of definition for \l__draw_softpath_part_tl.)

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

```

\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

```

```

1584 \fp_new:N \l__draw_softpath_lastx_fp
1585 \fp_new:N \l__draw_softpath_lasty_fp
1586 \dim_new:N \l__draw_softpath_corneri_dim
1587 \dim_new:N \l__draw_softpath_cornerii_dim
1588 \tl_new:N \l__draw_softpath_first_tl
1589 \tl_new:N \l__draw_softpath_move_tl

```

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

`\c__draw_softpath_arc_fp` The magic constant.

```

1590 \fp_const:Nn \c__draw_softpath_arc_fp { 4/3 * (sqrt(2) - 1) }

```

(End of definition for `\c__draw_softpath_arc_fp`.)

`__draw_softpath_round_corners:` 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.

```

\__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:

```

```

1591 \cs_new_protected:Npn \__draw_softpath_round_corners:
1592 {
1593   \bool_if:NT \g__draw_softpath_corners_bool
1594   {
1595     \group_begin:
1596       \tl_clear:N \l__draw_softpath_main_tl
1597       \tl_clear:N \l__draw_softpath_part_tl
1598       \fp_zero:N \l__draw_softpath_lastx_fp
1599       \fp_zero:N \l__draw_softpath_lasty_fp
1600       \tl_clear:N \l__draw_softpath_first_tl
1601       \tl_clear:N \l__draw_softpath_move_tl
1602       \tl_build_gend:N \g__draw_softpath_main_tl
1603       \exp_after:wN \__draw_softpath_round_loop:Nnn
1604         \g__draw_softpath_main_tl
1605         \q__draw_recursion_tail ? ?
1606         \q__draw_recursion_stop
1607     \group_end:
1608   }
1609   \bool_gset_false:N \g__draw_softpath_corners_bool
1610 }

```

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.

```

1611 \cs_new_protected:Npn \__draw_softpath_round_loop:Nnn #1#2#3
1612 {
1613   \__draw_if_recursion_tail_stop_do:Nn #1 { \__draw_softpath_round_end: }
1614   \token_if_eq_meaning:NNTF #1 \__draw_softpath_roundpoint_op:nn
1615   { \__draw_softpath_round_action:nn {#2} {#3} }
1616   {
1617     \tl_if_empty:NT \l__draw_softpath_first_tl
1618     { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1619     \fp_set:Nn \l__draw_softpath_lastx_fp {#2}
1620     \fp_set:Nn \l__draw_softpath_lasty_fp {#3}

```

```

1621     \token_if_eq_meaning:NNTF #1 \__draw_softpath_moveto_op:nn
1622     {
1623         \tl_put_right:No \l__draw_softpath_main_tl
1624         \l__draw_softpath_move_tl
1625         \tl_put_right:No \l__draw_softpath_main_tl
1626         \l__draw_softpath_part_tl
1627         \tl_set:Nn \l__draw_softpath_move_tl { #1 {#2} {#3} }
1628         \tl_clear:N \l__draw_softpath_first_tl
1629         \tl_clear:N \l__draw_softpath_part_tl
1630     }
1631     { \tl_put_right:Nn \l__draw_softpath_part_tl { #1 {#2} {#3} } }
1632     \__draw_softpath_round_loop:Nnn
1633 }
1634 }
1635 \cs_new_protected:Npn \__draw_softpath_round_action:nn #1#2
1636 {
1637     \dim_set:Nn \l__draw_softpath_corneri_dim {#1}
1638     \dim_set:Nn \l__draw_softpath_cornerii_dim {#2}
1639     \bool_lazy_and:nnTF
1640     { \dim_compare_p:nNn \l__draw_softpath_corneri_dim = { Opt } }
1641     { \dim_compare_p:nNn \l__draw_softpath_cornerii_dim = { Opt } }
1642     { \__draw_softpath_round_loop:Nnn }
1643     { \__draw_softpath_round_action:Nnn }
1644 }

```

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.

```

1645 \cs_new_protected:Npn \__draw_softpath_round_action:Nnn #1#2#3
1646 {
1647     \tl_if_empty:NT \l__draw_softpath_first_tl
1648     { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1649     \token_if_eq_meaning:NNTF #1 \__draw_softpath_curveto_opi:nn
1650     { \__draw_softpath_round_action_curveto:NnnNnn }
1651     {
1652         \token_if_eq_meaning:NNTF #1 \__draw_softpath_close_op:nn
1653         { \__draw_softpath_round_action_close: }
1654         {
1655             \token_if_eq_meaning:NNTF #1 \__draw_softpath_lineto_op:nn
1656             { \__draw_softpath_round_lookahead:NnnNnn }
1657             { \__draw_softpath_round_loop:Nnn }
1658         }
1659     }
1660     #1 {#2} {#3}
1661 }

```

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.

```

1662 \cs_new_protected:Npn \__draw_softpath_round_action_curveto:NnnNnn
1663 #1#2#3#4#5#6
1664 {
1665     \tl_put_right:Nn \l__draw_softpath_part_tl
1666     { #1 {#2} {#3} #4 {#5} {#6} }
1667     \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1668     \fp_set:Nn \l__draw_softpath_lasty_fp {#6}

```

```

1669     \__draw_softpath_round_lookahead:NnnNnn
1670   }
1671   \cs_new_protected:Npn \__draw_softpath_round_action_close:
1672   {
1673     \bool_lazy_and:nnTF
1674     { ! \tl_if_empty_p:N \l__draw_softpath_first_tl }
1675     { ! \tl_if_empty_p:N \l__draw_softpath_move_tl }
1676     {
1677       \exp_after:wN \__draw_softpath_round_close:nn
1678       \l__draw_softpath_first_tl
1679     }
1680     { \__draw_softpath_round_loop:Nnn }
1681   }

```

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.

```

1682   \cs_new_protected:Npn \__draw_softpath_round_lookahead:NnnNnn #1#2#3#4#5#6
1683   {
1684     \bool_lazy_any:nTF
1685     {
1686       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_lineto_op:nn }
1687       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_curveto_opi:nn }
1688       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_close_op:nn }
1689     }
1690     {
1691       \__draw_softpath_round_calc:NnnNnn
1692       \__draw_softpath_round_loop:Nnn
1693       {#5} {#6}
1694     }
1695     {
1696       \token_if_eq_meaning:NNTF #4 \__draw_softpath_roundpoint_op:nn
1697       { \__draw_softpath_round_roundpoint:NnnNnnNnn }
1698       { \__draw_softpath_round_loop:Nnn }
1699     }
1700     #1 {#2} {#3}
1701     #4 {#5} {#6}
1702   }
1703   \cs_new_protected:Npn \__draw_softpath_round_roundpoint:NnnNnnNnn
1704   #1#2#3#4#5#6#7#8#9
1705   {
1706     \__draw_softpath_round_calc:NnnNnn
1707     \__draw_softpath_round_loop:Nnn
1708     {#8} {#9}
1709     #1 {#2} {#3}
1710     #4 {#5} {#6} #7 {#8} {#9}
1711   }

```

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.

```

1712 \cs_new_protected:Npn \__draw_softpath_round_calc:NnnNnn #1#2#3#4#5#6
1713 {
1714   \tl_set:Nc \l__draw_softpath_curve_end_tl
1715   {
1716     \draw_point_interpolate_distance:nnn
1717     \l__draw_softpath_cornerii_dim
1718     { #5 , #6 } { #2 , #3 }
1719   }
1720   \tl_put_right:Nc \l__draw_softpath_part_tl
1721   {
1722     \exp_not:N #4
1723     \__draw_softpath_round_calc:eVnnnn
1724     {
1725       \draw_point_interpolate_distance:nnn
1726       \l__draw_softpath_corneri_dim
1727       { #5 , #6 }
1728       {
1729         \l__draw_softpath_lastx_fp ,
1730         \l__draw_softpath_lasty_fp
1731       }
1732     }
1733     \l__draw_softpath_curve_end_tl
1734     {#5} {#6} {#2} {#3}
1735   }
1736   \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1737   \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1738   #1
1739 }

```

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

```

1740 \cs_new:Npn \__draw_softpath_round_calc:nnnnnn #1#2#3#4#5#6
1741 {
1742   \__draw_softpath_round_calc:nnnnw {#3} {#4} {#5} {#6}
1743   #1 \s__draw_mark #2 \s__draw_stop
1744 }
1745 \cs_generate_variant:Nn \__draw_softpath_round_calc:nnnnnn { eV }

```

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

```

1746 \cs_new:Npn \__draw_softpath_round_calc:nnnnw
1747 #1#2#3#4 #5 , #6 \s__draw_mark #7 , #8 \s__draw_stop
1748 {
1749   {#5} {#6}
1750   \exp_not:N \__draw_softpath_curveto_opi:nn
1751   {
1752     \fp_to_dim:n
1753     { #5 + \c__draw_softpath_arc_fp * ( #1 - #5 ) }
1754   }
1755   {
1756     \fp_to_dim:n
1757     { #6 + \c__draw_softpath_arc_fp * ( #2 - #6 ) }

```



```

1758     }
1759     \exp_not:N \__draw_softpath_curveto_opii:nn
1760     {
1761         \fp_to_dim:n
1762         { #7 + \c__draw_softpath_arc_fp * ( #1 - #7 ) }
1763     }
1764     {
1765         \fp_to_dim:n
1766         { #8 + \c__draw_softpath_arc_fp * ( #2 - #8 ) }
1767     }
1768     \exp_not:N \__draw_softpath_curveto_opiii:nn
1769     {#7} {#8}
1770 }

```

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.

```

1771 \cs_new_protected:Npn \__draw_softpath_round_close:nn #1#2
1772 {
1773     \use:e
1774     {
1775         \__draw_softpath_round_calc:NnnNnn
1776         {
1777             \tl_set:Nc \exp_not:N \l__draw_softpath_move_tl
1778             {
1779                 \__draw_softpath_moveto_op:nn
1780                 \exp_not:N \exp_after:wN
1781                 \exp_not:N \__draw_softpath_round_close:w
1782                 \exp_not:N \l__draw_softpath_curve_end_tl
1783                 \s__draw_stop
1784             }
1785             \use:e
1786             {
1787                 \exp_not:N \exp_not:N \exp_not:N \use_i:nnnn
1788                 {
1789                     \__draw_softpath_round_loop:Nnn
1790                     \__draw_softpath_close_op:nn
1791                     \exp_not:N \exp_after:wN
1792                     \exp_not:N \__draw_softpath_round_close:w
1793                     \exp_not:N \l__draw_softpath_curve_end_tl
1794                     \s__draw_stop
1795                 }
1796             }
1797         }
1798         {#1} {#2}
1799         \__draw_softpath_lineto_op:nn
1800         \exp_after:wN \use_none:n \l__draw_softpath_move_tl
1801     }
1802 }
1803 \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.

```

1804 \cs_new_protected:Npn \__draw_softpath_round_end:
1805 {

```

```

1806 \tl_put_right:No \l__draw_softpath_main_tl
1807 \l__draw_softpath_move_tl
1808 \tl_put_right:No \l__draw_softpath_main_tl
1809 \l__draw_softpath_part_tl
1810 \tl_build_gbegin:N \g__draw_softpath_main_tl
1811 \__draw_softpath_add:o \l__draw_softpath_main_tl
1812 }

```

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

```

1813 </package>

```

8 l3draw-state implementation

```

1814 <*package>

```

```

1815 <@@=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.

```

1816 \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.

```

1817 \dim_new:N \l_draw_default_linewidth_dim
1818 \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_set_linewidth:n` Set the linewidth: we need a wrapper as this has to pass to the driver layer.

```

1819 \cs_new_protected:Npn \draw_set_linewidth:n #1
1820 {
1821   \dim_gset:Nn \g__draw_linewidth_dim { \fp_to_dim:n {#1} }
1822   \__draw_backend_linewidth:n \g__draw_linewidth_dim
1823 }

```

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

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

```

\l__draw_tmp_seq
1824 \cs_new_protected:Npn \draw_set_dash_pattern:nn #1#2
1825 {
1826   \group_begin:
1827     \seq_set_from_clist:Nn \l__draw_tmp_seq {#1}
1828     \seq_set_map:NNn \l__draw_tmp_seq \l__draw_tmp_seq
1829       { \fp_to_dim:n {##1} }
1830     \use:e
1831     {
1832       \__draw_backend_dash_pattern:nn
1833       { \seq_use:Nn \l__draw_tmp_seq { , } }

```

```

1834         { \fp_to_dim:n {#2} }
1835     }
1836     \group_end:
1837 }
1838 \seq_new:N \l__draw_tmp_seq

```

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

`\draw_set_miterlimit:n` Pass through to the driver layer.

```

1839 \cs_new_protected:Npn \draw_set_miterlimit:n #1
1840 { \exp_args:Ne \__draw_backend_miterlimit:n { \fp_eval:n {#1} } }

```

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

`\draw_set_cap_but`: All straight wrappers.

```

\draw_set_cap_rectangle: 1841 \cs_new_protected:Npn \draw_set_cap_but: { \__draw_backend_cap_but: }
\draw_set_cap_round:      1842 \cs_new_protected:Npn \draw_set_cap_rectangle: { \__draw_backend_cap_rectangle: }
\draw_set_evenodd_rule:   1843 \cs_new_protected:Npn \draw_set_cap_round: { \__draw_backend_cap_round: }
\draw_set_nonzero_rule:   1844 \cs_new_protected:Npn \draw_set_evenodd_rule: { \__draw_backend_evenodd_rule: }
\draw_set_join_bevel:     1845 \cs_new_protected:Npn \draw_set_nonzero_rule: { \__draw_backend_nonzero_rule: }
\draw_set_join_miter:     1846 \cs_new_protected:Npn \draw_set_join_bevel: { \__draw_backend_join_bevel: }
\draw_set_join_round:     1847 \cs_new_protected:Npn \draw_set_join_miter: { \__draw_backend_join_miter: }
                          1848 \cs_new_protected:Npn \draw_set_join_round: { \__draw_backend_join_round: }

```

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

```

1849 \</package>

```

9 l3draw-transforms implementation

```

1850 \*package>

```

```

1851 \@@=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.
- `\pgftransformlineatime`, `\pgftransformarcaxesatime`, `\pgftransformcurveatime`: Need to look at the use cases for these to fully understand them.
- `\pgftransformarrow`: Likely to be done when other arrow functions are added.
- `\pgftransformationadjustments`: Used mainly by CircuiTiKZ although also for shapes, likely needs more use cases before addressing.
- `\pgflowlevelsynccm`, `\pgflowlevel`: Likely to be added when use cases are encountered in other parts of the code.
- `\pgfviewboxscope`: Seems very specialized, need to understand the requirements here.

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

```

1852 \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 1853 \fp_new:N \l__draw_matrix_a_fp
\l__draw_matrix_c_fp 1854 \fp_new:N \l__draw_matrix_b_fp
\l__draw_xshift_dim 1855 \fp_new:N \l__draw_matrix_c_fp
\l__draw_yshift_dim 1856 \fp_new:N \l__draw_matrix_d_fp
1857 \dim_new:N \l__draw_xshift_dim
1858 \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: 1859 \cs_new_protected:Npn \draw_transform_matrix_reset:
1860 {
1861 \fp_set:Nn \l__draw_matrix_a_fp { 1 }
1862 \fp_zero:N \l__draw_matrix_b_fp
1863 \fp_zero:N \l__draw_matrix_c_fp
1864 \fp_set:Nn \l__draw_matrix_d_fp { 1 }
1865 \bool_set_false:N \l__draw_matrix_active_bool
1866 }
1867 \cs_new_protected:Npn \draw_transform_shift_reset:
1868 {
1869 \dim_zero:N \l__draw_xshift_dim
1870 \dim_zero:N \l__draw_yshift_dim
1871 }
1872 \draw_transform_matrix_reset:
1873 \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 1874 \cs_new_protected:Npn \draw_transform_matrix_absolute:nmmn #1#2#3#4
1875 {
1876 \fp_set:Nn \l__draw_matrix_a_fp {#1}
1877 \fp_set:Nn \l__draw_matrix_b_fp {#2}
1878 \fp_set:Nn \l__draw_matrix_c_fp {#3}
1879 \fp_set:Nn \l__draw_matrix_d_fp {#4}
1880 \bool_lazy_all:nTF
1881 {
1882 { \fp_compare_p:nNn \l__draw_matrix_a_fp = \c_one_fp }
1883 { \fp_compare_p:nNn \l__draw_matrix_b_fp = \c_zero_fp }
1884 { \fp_compare_p:nNn \l__draw_matrix_c_fp = \c_zero_fp }
1885 { \fp_compare_p:nNn \l__draw_matrix_d_fp = \c_one_fp }
1886 }
1887 { \bool_set_false:N \l__draw_matrix_active_bool }
1888 { \bool_set_true:N \l__draw_matrix_active_bool }
1889 }
1890 \cs_new_protected:Npn \draw_transform_shift_absolute:n #1
1891 {
1892 _draw_point_process:nn
1893 { _draw_transform_shift_absolute:nn } { \draw_point:n {#1} }

```

1894 }
1895 \cs_new_protected:Npn \__draw_transform_shift_absolute:nn #1#2
1896 {
1897   \dim_set:Nn \l__draw_xshift_dim {#1}
1898   \dim_set:Nn \l__draw_yshift_dim {#2}
1899 }

```

(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
1900 \cs_new_protected:Npn \draw_transform_matrix:nnnn #1#2#3#4
1901 {
1902   \use:e
1903   {
1904     \__draw_transform:nnnn
1905     { \fp_eval:n {#1} }
1906     { \fp_eval:n {#2} }
1907     { \fp_eval:n {#3} }
1908     { \fp_eval:n {#4} }
1909   }
1910 }
1911 \cs_new_protected:Npn \__draw_transform:nnnn #1#2#3#4
1912 {
1913   \use:e
1914   {
1915     \draw_transform_matrix_absolute:nnnn
1916     { #1 * \l__draw_matrix_a_fp + #2 * \l__draw_matrix_c_fp }
1917     { #1 * \l__draw_matrix_b_fp + #2 * \l__draw_matrix_d_fp }
1918     { #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_c_fp }
1919     { #3 * \l__draw_matrix_b_fp + #4 * \l__draw_matrix_d_fp }
1920   }
1921 }
1922 \cs_new_protected:Npn \draw_transform_shift:n #1
1923 {
1924   \__draw_point_process:nn
1925   { \__draw_transform_shift:nn } { \draw_point:n {#1} }
1926 }
1927 \cs_new_protected:Npn \__draw_transform_shift:nn #1#2
1928 {
1929   \__draw_transform_shift:nnnn
1930   \l__draw_xshift_dim
1931   \l__draw_yshift_dim
1932   {#1} {#2}
1933 }

```

(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 non-zero starting point here.

```

1934 \cs_new_protected:Npn \__draw_transform_shift:nnnn #1#2#3#4
1935 {
1936   \dim_set:Nn \l__draw_xshift_dim

```

```

1937 {
1938   \fp_to_dim:n
1939   {
1940     #1 +
1941     ( #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_c_fp )
1942   }
1943 }
1944 \dim_set:Nn \l__draw_yshift_dim
1945 {
1946   \fp_to_dim:n
1947   {
1948     #2 +
1949     ( #3 * \l__draw_matrix_b_fp + #4 * \l__draw_matrix_d_fp )
1950   }
1951 }
1952 }

```

(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 1953 \cs_new_protected:Npn \draw_transform_matrix_invert:
\__draw_transform_invert:e 1954 {
\draw_transform_shift_invert: 1955   \bool_if:NT \l__draw_matrix_active_bool
1956   {
1957     \__draw_transform_invert:e
1958     {
1959       \fp_eval:n
1960       {
1961         1 /
1962         (
1963           \l__draw_matrix_a_fp * \l__draw_matrix_d_fp
1964           - \l__draw_matrix_b_fp * \l__draw_matrix_c_fp
1965         )
1966       }
1967     }
1968   }
1969 }
1970 \cs_new_protected:Npn \__draw_transform_invert:n #1
1971 {
1972   \fp_set:Nn \l__draw_matrix_a_fp
1973   { \l__draw_matrix_d_fp * #1 }
1974   \fp_set:Nn \l__draw_matrix_b_fp
1975   { -\l__draw_matrix_b_fp * #1 }
1976   \fp_set:Nn \l__draw_matrix_c_fp
1977   { -\l__draw_matrix_c_fp * #1 }
1978   \fp_set:Nn \l__draw_matrix_d_fp
1979   { \l__draw_matrix_a_fp * #1 }
1980 }
1981 \cs_generate_variant:Nn \__draw_transform_invert:n { e }
1982 \cs_new_protected:Npn \draw_transform_shift_invert:
1983 {
1984   \dim_set:Nn \l__draw_xshift_dim { -\l__draw_xshift_dim }
1985   \dim_set:Nn \l__draw_yshift_dim { -\l__draw_yshift_dim }
1986 }

```

(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.

```

1987 \cs_new_protected:Npn \draw_transform_triangle:nnn #1#2#3
1988 {
1989   \__draw_point_process:nnnn
1990   { \__draw_transform_triangle:nnnnnn }
1991   { \draw_point:n {#1} }
1992   { \draw_point:n {#2} }
1993   { \draw_point:n {#3} }
1994 }
1995 \cs_new_protected:Npn \__draw_transform_triangle:nnnnnn #1#2#3#4#5#6
1996 {
1997   \use:e
1998   {
1999     \draw_transform_matrix_absolute:nnnn
2000     { #3 - #1 }
2001     { #4 - #2 }
2002     { #5 - #1 }
2003     { #6 - #2 }
2004     \draw_transform_shift_absolute:n { #1 , #2 }
2005   }
2006 }

```

(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 2007 \cs_new_protected:Npn \draw_transform_scale:n #1
\draw_transform_yscale:n 2008 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { #1 } }
\draw_transform_xshift:n 2009 \cs_new_protected:Npn \draw_transform_xscale:n #1
\draw_transform_yshift:n 2010 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { 1 } }
\draw_transform_xslant:n 2011 \cs_new_protected:Npn \draw_transform_yscale:n #1
\draw_transform_yslant:n 2012 { \draw_transform_matrix:nnnn { 1 } { 0 } { 0 } { #1 } }
2013 \cs_new_protected:Npn \draw_transform_xshift:n #1
2014 { \draw_transform_shift:n { #1 , Opt } }
2015 \cs_new_protected:Npn \draw_transform_yshift:n #1
2016 { \draw_transform_shift:n { Opt , #1 } }
2017 \cs_new_protected:Npn \draw_transform_xslant:n #1
2018 { \draw_transform_matrix:nnnn { 1 } { 0 } { #1 } { 1 } }
2019 \cs_new_protected:Npn \draw_transform_yslant:n #1
2020 { \draw_transform_matrix:nnnn { 1 } { #1 } { 0 } { 1 } }

```

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

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

```

\__draw_transform_rotate:n 2021 \cs_new_protected:Npn \draw_transform_rotate:n #1
\__draw_transform_rotate:e 2022 { \__draw_transform_rotate:e { \fp_eval:n {#1} } }
\__draw_transform_rotate:nn 2023 \cs_new_protected:Npn \__draw_transform_rotate:n #1
\__draw_transform_rotate:ee 2024 {
2025   \__draw_transform_rotate:ee
2026   { \fp_eval:n { cosd(#1) } }
2027   { \fp_eval:n { sind(#1) } }
2028 }

```

```

2029 \cs_generate_variant:Nn \_draw_transform_rotate:n { e }
2030 \cs_new_protected:Npn \_draw_transform_rotate:nn #1#2
2031   { \draw_transform_matrix:nnnn {#1} {#2} { -#2 } { #1 } }
2032 \cs_generate_variant:Nn \_draw_transform_rotate:nn { ee }

```

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

```

2033 \endpackage

```


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` ... 194, 829, 1117, 1121, 1145, 1148

bool commands:

`\bool_gset_eq:NN` 1503

`\bool_gset_false:N` 1486, 1609

`\bool_gset_true:N` 1510, 1556

`\bool_if:NTF`
 . 33, 137, 217, 256, 730, 746, 747,
 751, 1238, 1271, 1405, 1538, 1593, 1955

`\bool_lazy_all:nTF` 1880

`\bool_lazy_and:nnTF`
 248, 725, 1639, 1673

`\bool_lazy_any:nTF` 1684

`\bool_lazy_or:nnTF` . 583, 684, 734, 739

`\bool_new:N` 108, 243, 673, 674, 675,
 676, 796, 1295, 1396, 1472, 1509, 1852

`\bool_set_eq:NN` 1494

`\bool_set_false:N` 118,
 251, 712, 713, 714, 733, 1865, 1887

`\bool_set_true:N` 120, 252,
 718, 759, 763, 764, 1314, 1400, 1888

box commands:

`\box_dp:N` 17, 22, 89

`\box_gclear:N` 169

`\box_gset_eq:NN` 179

`\box_gset_wd:Nn` 122, 155

`\box_ht:N` 17, 22, 91

`\box_if_exist:NTF` 115

`\box_move_down:nn` 1370, 1383

`\box_move_up:nn` 63

`\box_new:N` ... 13, 102, 103, 1296, 1297

`\box_set_dp:Nn` 67, 1374, 1387

`\box_set_eq:NN` 167

`\box_set_ht:Nn` 66, 1375, 1393

`\box_set_wd:Nn` 68, 150, 1344

`\box_use_drop:N` 64,
 69, 124, 151, 156, 1347, 1372, 1385

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

C

clist commands:

`\clist_map_inline:Nn` .. 146, 163, 175

`\clist_map_inline:nn` 715

`\clist_new:N` 109, 111

`\clist_set:Nn` 110, 1332

coffin commands:

`\coffin_typeset:Nnnnn` 86

`\coffin_wd:N` 88

color commands:

`\color_ensure_current:` 1319

cs commands:

`\cs_generate_variant:Nn`
 445, 639, 672,
 837, 844, 851, 862, 870, 878, 896,
 924, 944, 951, 959, 966, 975, 981,
 1003, 1011, 1018, 1024, 1037, 1040,
 1058, 1078, 1086, 1092, 1116, 1130,
 1144, 1165, 1198, 1217, 1230, 1364,
 1475, 1558, 1745, 1981, 2029, 2032

`\cs_if_exist:NTF` 717, 1355

`\cs_if_exist_use:NTF` .. 428, 437, 720

`\cs_new:Npn` 538, 548, 558, 568, 833,
 835, 838, 840, 842, 845, 847, 849,
 852, 855, 857, 863, 866, 868, 871,
 872, 874, 876, 879, 881, 887, 897,
 906, 916, 925, 932, 937, 945, 952,
 960, 967, 976, 982, 990, 995, 1004,
 1012, 1019, 1025, 1032, 1038, 1041,
 1047, 1056, 1059, 1067, 1072, 1079,
 1087, 1093, 1102, 1108, 1122, 1131,
 1137, 1149, 1151, 1160, 1188, 1190,
 1199, 1204, 1218, 1220, 1222, 1231,
 1236, 1263, 1269, 1740, 1746, 1803

`\cs_new_protected:Npe` 1350

`\cs_new_protected:Npn`
 14, 19, 24, 31, 72, 77,
 82, 97, 112, 135, 144, 161, 173, 207,
 229, 236, 244, 254, 263, 269, 275,
 281, 288, 299, 307, 312, 314, 316,
 327, 334, 370, 372, 383, 389, 419,
 446, 475, 481, 487, 492, 500, 509,
 516, 524, 579, 581, 597, 604, 613,
 620, 622, 633, 640, 646, 653, 677,
 682, 693, 702, 710, 757, 761, 766,
 773, 797, 811, 1172, 1174, 1176,
 1178, 1180, 1299, 1306, 1328, 1365,
 1378, 1398, 1403, 1418, 1425, 1436,
 1445, 1453, 1461, 1473, 1476, 1483,
 1488, 1499, 1511, 1520, 1529, 1534,
 1544, 1552, 1559, 1561, 1563, 1565,
 1567, 1569, 1571, 1573, 1575, 1577,
 1579, 1591, 1611, 1635, 1645, 1662,
 1671, 1682, 1703, 1712, 1771, 1804,
 1819, 1824, 1839, 1841, 1842, 1843,
 1844, 1845, 1846, 1847, 1848, 1859,
 1867, 1874, 1890, 1895, 1900, 1911,

1922, 1927, 1934, 1953, 1970, 1982,
1987, 1995, 2007, 2009, 2011, 2013,
2015, 2017, 2019, 2021, 2023, 2030

D

dim commands:

\dim_abs:n 627, 628
\dim_compare:nNnTF 635, 642, 775, 1336
\dim_compare_p:nNn 249, 250, 1640, 1641
\dim_gset:Nn 209, 211, 213,
215, 219, 221, 223, 225, 231, 232,
233, 234, 238, 239, 704, 777, 1301,
1302, 1303, 1304, 1540, 1541, 1821
\dim_gset_eq:NN
..... 814, 815, 816, 817, 818, 819,
820, 821, 1428, 1447, 1448, 1449, 1450
\dim_gzero:N .. 1338, 1339, 1340, 1341
\dim_horizontal:N 62
\dim_max:nn .. 210, 214, 220, 224, 1389
\dim_min:nn 212, 216, 222, 226
\dim_new:N 201, 202,
203, 204, 205, 206, 241, 242, 788,
789, 790, 791, 792, 793, 794, 795,
1166, 1167, 1168, 1169, 1170, 1171,
1291, 1292, 1293, 1294, 1397, 1415,
1432, 1433, 1434, 1435, 1507, 1508,
1586, 1587, 1816, 1817, 1857, 1858
\dim_set:Nn 246, 247,
1182, 1183, 1401, 1637, 1638, 1818,
1897, 1898, 1936, 1944, 1984, 1985
\dim_set_eq:NN
..... 800, 801, 802, 803, 804, 805,
806, 807, 1422, 1439, 1440, 1441, 1442
\dim_step_inline:nnnn 655, 663
\dim_use:N
..... 706, 775, 780, 782, 1410, 1516, 1517
\dim_zero:N 1869, 1870
\c_max_dim 231, 232, 233,
234, 775, 1301, 1302, 1303, 1304, 1336

draw commands:

\l_draw_bb_update_bool
..... 33, 217, 726, 733, 1295, 1314
\g_draw_bb_xmax_dim 219,
220, 1291, 1301, 1338, 1345, 1439, 1447
\g_draw_bb_xmin_dim
..... 221, 222, 1291, 1302, 1336,
1339, 1345, 1369, 1382, 1440, 1448
\g_draw_bb_ymax_dim 223, 224, 1291,
1303, 1340, 1376, 1394, 1441, 1449
\g_draw_bb_ymin_dim
..... 225, 226, 1291, 1304,
1341, 1371, 1376, 1390, 1442, 1450
\draw_begin: 1306, 1306
\draw_box_use:N 14, 14

\draw_box_use:Nn 14, 19
\draw_coffin_use:Nnn 36, 72, 72
\draw_coffin_use:Nnnn 72, 77
\l_draw_default_linewidth_dim ...
..... 127, 1318, 1817
\draw_end: 1306, 1328
\g_draw_id_int 1298, 1309, 1358
\draw_layer_begin:n 112, 112
\draw_layer_end: 112, 135
\draw_layer_new:n 97, 97
\l_draw_layers_clist
..... 109, 146, 163, 175, 1332
\draw_path_arc:nnn 370, 370, 513
\draw_path_arc:nnnn ... 370, 371, 372
\draw_path_arc_axes:nnnn ... 509, 509
\draw_path_canvas_curveto:nnn ...
..... 312, 316
\draw_path_canvas_lineto:n . 312, 314
\draw_path_canvas_moveto:n . 312, 312
\draw_path_circle:nn 579, 579
\draw_path_close: 307, 307, 610
\draw_path_corner_arc:nn ... 244, 244
\draw_path_curveto:nn 327, 327
\draw_path_curveto:nnn 263, 288
\draw_path_ellipse:nnn . 516, 516, 580
\draw_path_grid:nnnn 622, 622
\draw_path_lineto:n
..... 263, 275, 607, 608, 609, 661, 669
\draw_path_moveto:n
..... 263, 263, 606, 611, 660, 668
\draw_path_rectangle:nn 581, 581, 621
\draw_path_rectangle_corners:nn .
..... 613, 613
\draw_path_replace_bb: 677, 693
\draw_path_scope_begin:
..... 797, 797, 1423, 1456
\draw_path_scope_end:
..... 797, 811, 1427, 1464
\draw_path_use:n 677, 677
\draw_path_use_clear:n 677, 682
\draw_point:n
..... 313, 315, 323, 324, 325, 588, 589,
593, 594, 617, 618, 630, 631, 866,
866, 877, 880, 892, 901, 902, 903,
904, 918, 929, 930, 978, 986, 987,
988, 1021, 1029, 1030, 1039, 1057,
1063, 1064, 1065, 1089, 1097, 1098,
1099, 1100, 1162, 1179, 1192, 1206,
1224, 1234, 1240, 1256, 1267, 1273,
1286, 1893, 1925, 1991, 1992, 1993
\draw_point_interpolate_arcaxes:nnnnnn
..... 1059, 1059
\draw_point_interpolate_curve:nnnnn
..... 1093

\draw_point_interpolate_curve:nnnnnn	1093	\draw_set_linewidth:n	127, 1318, 1819, 1819
\draw_point_interpolate_curve_-auxi:nnnnnnnnnn	1093	\draw_set_miterlimit:n	1323, 1839, 1839
\draw_point_interpolate_curve_-auxii:nnnnnnnnnn	1093	\draw_set_nonzero_rule:	1320, 1841, 1845
\draw_point_interpolate_curve_-auxiii:nnnnnn	1093	\draw_suspend_begin:	1453, 1453
\draw_point_interpolate_curve_-auxiv:nnnnnn	1093	\draw_suspend_end:	1453, 1461
\draw_point_interpolate_curve_-auxv:nnw	1093	\draw_transform_matrix:nnnn	1900, 1900, 2008, 2010, 2012, 2018, 2020, 2031
\draw_point_interpolate_curve_-auxvi:n	1093	\draw_transform_matrix_absolute:nnnn	1874, 1874, 1915, 1999
\draw_point_interpolate_curve_-auxvii:nnnnnnnn	1093	\draw_transform_matrix_invert:	1953, 1953
\draw_point_interpolate_curve_-auxviii:nnnnnn	1093	\draw_transform_matrix_reset:	1315, 1457, 1859, 1859, 1872
\draw_point_interpolate_distance:nnn	1041, 1041, 1716, 1725	\draw_transform_rotate:n	2021, 2021
\draw_point_interpolate_line:nnn	1025, 1025	\draw_transform_scale:n	2007, 2007
\draw_point_intersect_circles:nnnnn	925, 925	\draw_transform_shift:n	27, 1900, 1922, 2014, 2016
\draw_point_intersect_line_-circle:nnnnn	982, 982	\draw_transform_shift_absolute:n	1874, 1890, 2004
\draw_point_intersect_lines:nnnn	897, 897	\draw_transform_shift_invert:	1953, 1982
\draw_point_polar:nn	872, 872	\draw_transform_shift_reset:	1316, 1458, 1859, 1867, 1873
\draw_point_polar:nnn	453, 459, 463, 469, 872, 873, 874	\draw_transform_triangle:nnn	512, 1987, 1987
\draw_point_transform:n	37, 40, 43, 46, 267, 279, 295, 296, 297, 331, 332, 458, 462, 520, 1231, 1231	\draw_transform_xscale:n	2007, 2009
\draw_point_unit_vector:n	879, 879, 1054	\draw_transform_xshift:n	2007, 2013
\draw_point_vec:nn	1188, 1188	\draw_transform_xslant:n	2007, 2017
\draw_point_vec:nnn	1188, 1199	\draw_transform_yscale:n	2007, 2011
\draw_point_vec_polar:nn	1218, 1218	\draw_transform_yshift:n	2007, 2015
\draw_point_vec_polar:nnn	1218, 1219, 1220	\draw_transform_yslant:n	2007, 2019
\draw_scope_begin:	26, 1418, 1418	\draw_xvec:n	1172, 1172, 1185
\draw_scope_end:	29, 1425	\draw_yvec:n	1172, 1174, 1186
\draw_set_baseline:n	1398, 1398	\draw_zvec:n	1172, 1176, 1187
\draw_set_cap_but:	1321, 1841, 1841	draw internal commands:	
\draw_set_cap_rectangle:	1841, 1842	__draw_backend_begin:	1311
\draw_set_cap_round:	1841, 1843	__draw_backend_box_use:Nnnnn	53
\draw_set_dash_pattern:nn	1324, 1824, 1824	__draw_backend_cap_but:	1841
\draw_set_evenodd_rule:	1841, 1844	__draw_backend_cap_rectangle:	1842
\draw_set_join_bevel:	1841, 1846	__draw_backend_cap_round:	1843
\draw_set_join_miter:	1322, 1841, 1847	__draw_backend_clip:	732
\draw_set_join_round:	1841, 1848	__draw_backend_closepath:	1560
		__draw_backend_curveto:nnnnnn	1564
		__draw_backend_dash_pattern:nn	1832
		__draw_backend_discardpath:	737
		__draw_backend_end:	1334
		__draw_backend_evenodd_rule:	1844
		__draw_backend_join_bevel:	1846
		__draw_backend_join_miter:	1847
		__draw_backend_join_round:	1848

__draw_backend_lineto:nn	1570	\l__draw_matrix_a_fp	54,
__draw_backend_linewidth:n	..	1822			1243, 1276, 1853, 1861, 1876, 1882,
__draw_backend_miterlimit:n	.	1840			1916, 1918, 1941, 1963, 1972, 1979
__draw_backend_moveto:nn	1572	\l__draw_matrix_active_bool	
__draw_backend_nonzero_rule:	.	1845		 585, 1238,
__draw_backend_rectangle:nnnn		1578			1271, 1852, 1865, 1887, 1888, 1955
__draw_backend_scope_begin:	...		\l__draw_matrix_b_fp	55,
	154, 1420			1249, 1281, 1853, 1862, 1877, 1883,
__draw_backend_scope_end:		157, 1430			1917, 1919, 1949, 1964, 1974, 1975
\l__draw_baseline_bool		\l__draw_matrix_c_fp	56,
	1396, 1400, 1405			1244, 1277, 1853, 1863, 1878, 1884,
\l__draw_baseline_dim		1396, 1401, 1410			1916, 1918, 1941, 1964, 1976, 1977
__draw_baseline_finalize:w		\l__draw_matrix_d_fp	57,
	1330, 1403, 1403			1250, 1282, 1856, 1864, 1879, 1885,
__draw_box_use:Nnnnn	14			1917, 1919, 1949, 1963, 1973, 1978
__draw_box_use:nNnnnn	21, 24, 79	__draw_path_arc:nnnn	.	370, 376, 383
__draw_box_use:Nnnnnnn		16, 28, 31, 74	__draw_path_arc:nnNnn	
__draw_box_use:nNnnnnnn	14		 370, 386, 387, 389
__draw_coffin_use:nNnn		72, 74, 79, 82	\c__draw_path_arc_60_fp	370
\l__draw_corner_arc_bool		\c__draw_path_arc_90_fp	370
	243, 251, 252, 256, 584	__draw_path_arc_add:nnnn	370
\l__draw_corner_xarc_dim		__draw_path_arc_aux_add:nn	
	241, 246, 249, 259		 477, 483, 495, 500
\l__draw_corner_yarc_dim		__draw_path_arc_auxi:nnnnNnn	...	
	241, 247, 250, 260		 370, 397, 404, 412, 419, 445
__draw_draw_polar:nnn		__draw_path_arc_auxii:nnnNnnnn	.	
	872, 875, 876, 878		 370, 423, 446
__draw_draw_vec_polar:nnn		__draw_path_arc_auxiii:nn	
	1221, 1222, 1230		 370, 450, 475
\l__draw_fill_color_tl	1415	__draw_path_arc_auxiv:nnnn	
__draw_finalize: 370, 456, 481
	1343, 1365, 1365, 1403, 1413	__draw_path_arc_auxv:nn		370, 466, 487
__draw_finalize_baseline:n		__draw_path_arc_auxvi:nn	
	1365, 1378, 1410		 370, 489, 492
__draw_if_recursion_tail_stop-			\l__draw_path_arc_delta_fp	370
do:Nn	9, 9, 1613	\l__draw_path_arc_start_fp	370
\l__draw_layer_close_bool		__draw_path_curveto:nnnn	
	108, 118, 120, 137		 327, 330, 334
\l__draw_layer_main_box		__draw_path_curveto:nnnnnn	
	150, 151, 1296, 1325		 263, 293,
\l__draw_layer_tl	106, 117, 121			299, 321, 341, 471, 540, 550, 560, 570
\g__draw_layers_clist	109	\c__draw_path_curveto_a_fp	327
__draw_layers_insert:		144, 144, 1333	\c__draw_path_curveto_b_fp	327
__draw_layers_restore:		161, 173, 1463	__draw_path_ellipse:nnnnnn	
__draw_layers_save:	.	161, 161, 1459		 516, 519, 524
\g__draw_linewidth_dim		__draw_path_ellipse_arci:nnnnnn		
		707, 783, 1422, 1428, 1816, 1821, 1822		 516, 530, 538
\l__draw_linewidth_dim		__draw_path_ellipse_arcii:nnnnnn		
	1415, 1422, 1428		 516, 531, 548
\l__draw_main_box		__draw_path_ellipse_arciiii:nnnnnn		
		1296, 1310, 1344, 1347, 1367, 1372,		 516, 532, 558
		1374, 1375, 1380, 1385, 1387, 1393	__draw_path_ellipse_arciv:nnnnnn		
				 516, 533, 568
			\c__draw_path_ellipse_fp	516

__draw_path_grid_auxi:nnnnnn ...	\l__draw_path_use_fill_bool
..... 622, 626, 633, 639 673, 713, 735, 740, 746, 763
__draw_path_grid_auxii:nnnnnn ...	__draw_path_use_stroke_bb:
..... 622, 636, 637, 640 677, 728, 766
__draw_path_grid_auxiii:nnnnnn ...	\l__draw_path_use_stroke_bool ...
..... 622, 643, 644, 646	673, 714, 727, 736, 741, 747, 759, 764
__draw_path_grid_auxiiii:nnnnnn 622	\g__draw_path_xmax_dim
__draw_path_grid_auxiv:nnnnnnnn 203, 209, 210, 231, 802, 816
..... 622, 648, 653, 672	\l__draw_path_xmax_dim . 788, 802, 816
\g__draw_path_lastx_dim	\g__draw_path_xmin_dim
.... 201, 238, 345, 478, 484, 800, 820 203, 211, 212, 232, 803, 817
\l__draw_path_lastx_dim 788, 800, 820	\l__draw_path_xmin_dim . 788, 803, 817
\g__draw_path_lasty_dim	\g__draw_path_ymax_dim
.... 201, 239, 352, 479, 485, 801, 821 203, 213, 214, 233, 804, 818
\l__draw_path_lasty_dim 788, 801, 821	\l__draw_path_ymax_dim . 788, 804, 818
__draw_path_lineto:nn	\g__draw_path_ymin_dim
..... 263, 278, 281, 315 203, 215, 216, 234, 805, 819
__draw_path_mark_corner:	\l__draw_path_ymin_dim . 788, 805, 819
254, 254, 283, 292, 309, 320, 340, 411	__draw_point_interpolate_-
__draw_path_moveto:nn	arcaxes_auxi:nnnnnnnnnn
..... 263, 266, 269, 313, 528, 536 1059, 1062, 1067
__draw_path_rectangle:nnnn	__draw_point_interpolate_-
..... 581, 592, 597	arcaxes_auxii:nnnnnnnnnn
__draw_path_rectangle_corners:nnnn 1059, 1069, 1072, 1078
..... 613	__draw_point_interpolate_-
__draw_path_rectangle_corners:nnnnnn	arcaxes_auxiii:nnnnnnnn
..... 616, 620 1059, 1074, 1079, 1086
__draw_path_rectangle_rounded:nnnn	__draw_point_interpolate_-
..... 581, 587, 604	arcaxes_auxiv:nnnnnnnnnn
__draw_path_replace_bb:NnN 1059, 1081, 1087, 1092
..... 677, 695, 696, 697, 698, 702	__draw_point_interpolate_curve_-
__draw_path_reset_limits:	auxi:nnnnnnnnnn
... 207, 229, 689, 700, 754, 808, 1313 1096, 1102
\l__draw_path_tmp_tl	__draw_point_interpolate_curve_-
.... 198, 448, 471, 490, 494, 498, 502	auxii:nnnnnnnnnn . 1104, 1108, 1116
\l__draw_path_tmpa_fp	__draw_point_interpolate_curve_-
..... 198, 336, 346, 358	auxiii:nnnnnnnn ... 1111, 1122, 1130
\l__draw_path_tmpb_fp	__draw_point_interpolate_curve_-
..... 198, 337, 353, 362	auxiv:nnnnnnnn 1124, 1125, 1126, 1131
__draw_path_update_last:nn	__draw_point_interpolate_curve_-
..... 236, 236, 273, 286, 305, 602	auxv:nnw
__draw_path_update_limits:nn ...	1133, 1137, 1144
..... 36, 39, 42, 45, 207,	__draw_point_interpolate_curve_-
207, 271, 284, 301, 302, 303, 599, 600	auxvi:n
__draw_path_use:n . 677, 680, 691, 710	1128, 1149
__draw_path_use_action_draw: ...	__draw_point_interpolate_curve_-
..... 677, 757	auxvii:nnnnnnnnnn
__draw_path_use_action_fillstroke:	1150, 1151
..... 677, 761	__draw_point_interpolate_curve_-
__draw_path_use_bb:NnN	auxviii:nnnnnnnn .. 1153, 1160, 1165
..... 677, 768, 769, 770, 771, 773	__draw_point_interpolate_-
\l__draw_path_use_clear_bool 676, 751	distance:nnnn
\l__draw_path_use_clip_bool	1044, 1047
..... 673, 712, 730	__draw_point_interpolate_-
	distance:nnnnnn
	1041, 1051, 1056, 1058
	__draw_point_interpolate_-
	distance:nnnnnn
	1041

_draw_point_interpolate_line_-	_draw_point_process_auxiv:nw ..
aux:nnnnn .. 1025 , 1028 , 1032 , 1037 833 , 843 , 845
_draw_point_interpolate_line_-	_draw_point_process_auxv:nnnn ..
aux:nnnnnn .. 1025 , 1034 , 1038 , 1040 833 , 848 , 849 , 851
_draw_point_intersect_circles_-	_draw_point_process_auxvi:nw ..
auxi:nnnnnnn 925 , 928 , 932 833 , 850 , 852
_draw_point_intersect_circles_-	_draw_point_process_auxvii:nnnnn
auxii:nnnnnnn .. 925 , 934 , 937 , 944 833 , 856 , 857 , 862
_draw_point_intersect_circles_-	_draw_point_process_auxviii:nw
auxiii:nnnnnnn .. 925 , 939 , 945 , 951 833 , 859 , 863
_draw_point_intersect_circles_-	_draw_point_to_dim:n
auxiv:nnnnnnnn .. 925 , 947 , 952 , 959 866 , 867 , 868 , 870
_draw_point_intersect_circles_-	_draw_point_to_dim:w .. 866 , 869 , 871
auxv:nnnnnnnnn .. 925 , 954 , 960 , 966	_draw_point_transform:nn
_draw_point_intersect_circles_- 1231 , 1234 , 1236
auxvi:nnnnnnnn .. 925 , 962 , 967 , 975	_draw_point_transform_noshift:n
_draw_point_intersect_circles_- 452 , 468 , 521 , 522 , 1263 , 1263
auxvii:nnnnnnn .. 925 , 969 , 976 , 981	_draw_point_transform_noshift:nn
_draw_point_intersect_line_- 1263 , 1266 , 1269
circle_auxi:nnnnnnnn 982 , 985 , 990	_draw_point_unit_vector:nn
_draw_point_intersect_line_- 879 , 880 , 881
circle_auxii:nnnnnnnn	_draw_point_unit_vector:nnn
..... 982 , 992 , 995 , 1003 879 , 883 , 887 , 896
_draw_point_intersect_line_-	_draw_point_vec:nn
circle_auxiii:nnnnnnnn 1188 , 1189 , 1190 , 1198
..... 982 , 997 , 1004 , 1011	_draw_point_vec:nnn
_draw_point_intersect_line_- 1188 , 1201 , 1204 , 1217
circle_auxiv:nnnnnnnn	_draw_point_vec_polar:nnn .. 1218
..... 982 , 1006 , 1012 , 1018	_draw_record_origin:
_draw_point_intersect_line_- 1326 , 1350 , 1350
circle_auxv:nnnnn	_draw_reset_bb:
..... 982 , 1014 , 1019 , 1024 1299 , 1299 , 1312 , 1443
_draw_point_intersect_lines:nnnnnn	_draw_scope_bb_begin:
..... 897 1436 , 1436 , 1455
_draw_point_intersect_lines:nnnnnnnn	_draw_scope_bb_end: 1436 , 1445 , 1465
..... 897 , 900 , 906	_draw_softpath_add:n
_draw_point_intersect_lines_-	... 1473 , 1473 , 1475 , 1502 , 1513 ,
aux:nnnnnn	1522 , 1531 , 1536 , 1546 , 1554 , 1811
..... 897 , 908 , 916 , 924	_c__draw_softpath_arc_fp
_draw_point_process:nn 1590 , 1753 , 1757 , 1762 , 1766
..... 35 , 38 , 41 , 44 , 265 , 277 , 313 ,	_draw_softpath_clear:
315 , 449 , 465 , 833 , 833 , 880 , 1043 , 688 ,
1049 , 1179 , 1233 , 1265 , 1892 , 1924 699 , 753 , 1317 , 1476 , 1483 , 1497 , 1501
_draw_point_process:nnn 329 , 455 ,	_draw_softpath_close_op:nn
587 , 592 , 615 , 624 , 833 , 840 , 927 , 1027	... 1515 , 1559 , 1559 , 1652 , 1688 , 1790
_draw_point_process:nnnn .. 290 ,	_draw_softpath_closepath:
318 , 518 , 833 , 847 , 984 , 1061 , 1989 310 , 535 , 1511 , 1511
_draw_point_process:nnnnn	_l__draw_softpath_corneri_dim
..... 833 , 855 , 899 , 1095 1584 , 1637 , 1640 , 1726
_draw_point_process_auxi:nn	_l__draw_softpath_cornerii_dim
..... 833 , 834 , 835 , 837 1584 , 1638 , 1641 , 1717
_draw_point_process_auxii:nw ..	_g__draw_softpath_corners_bool ..
..... 833 , 836 , 838 1472 ,
_draw_point_process_auxiii:nnn 1486 , 1496 , 1504 , 1556 , 1593 , 1609
..... 833 , 841 , 842 , 844	

\l__draw_softpath_corners_bool . .	__draw_softpath_rectangle_-
..... 788, 1495, 1505	opi:nn 1548, 1559, 1575
\l__draw_softpath_curve_end_tl . .	__draw_softpath_rectangle_-
..... 1583, 1714, 1733, 1782, 1793	opi:nnNnn 1559, 1576, 1577
__draw_softpath_curveto:nnnnnn .	__draw_softpath_rectangle_-
..... 304, 1511, 1520	opii:nn 1549, 1559, 1579, 1580
__draw_softpath_curveto_opi:nn .	__draw_softpath_restore:
.. 1524, 1559, 1561, 1649, 1687, 1750 813, 1488, 1499
__draw_softpath_curveto_-	__draw_softpath_round_action:nn
opi:nnNnnNnn 1559, 1562, 1563 1591, 1615, 1635
__draw_softpath_curveto_opii:nn	__draw_softpath_round_action:Nnn
..... 1525, 1559, 1565, 1566, 1759 1591, 1643, 1645
__draw_softpath_curveto_-	__draw_softpath_round_action_-
opiii:nn 1526, 1559, 1567, 1568, 1768	close: 1591, 1653, 1671
\l__draw_softpath_first_tl	__draw_softpath_round_action_-
..... 1584, 1600, 1617,	curveto:NnnNnn .. 1591, 1650, 1662
1618, 1628, 1647, 1648, 1674, 1678	__draw_softpath_round_calc:NnnNnn
\g__draw_softpath_lastx_dim 1591, 1691, 1706, 1712, 1775
..... 806, 814, 1507, 1516, 1540	__draw_softpath_round_calc:nnnnnn
\l__draw_softpath_lastx_dim 1591, 1723, 1740, 1745
..... 794, 806, 814	__draw_softpath_round_calc:nnnnnw
\l__draw_softpath_lastx_fp 1591, 1742, 1746
.. 1584, 1598, 1619, 1667, 1729, 1736	__draw_softpath_round_close:nn .
\g__draw_softpath_lasty_dim 1591, 1677, 1771
..... 807, 815, 1507, 1517, 1541	__draw_softpath_round_close:w . .
\l__draw_softpath_lasty_dim 1591, 1781, 1792, 1803
..... 795, 807, 815	__draw_softpath_round_corners: .
\l__draw_softpath_lasty_fp 724, 1591, 1591
.. 1584, 1599, 1620, 1668, 1730, 1737	__draw_softpath_round_end:
__draw_softpath_lineto:nn 1591, 1613, 1804
..... 285, 1511, 1529	__draw_softpath_round_lookahead:NnnNnn
__draw_softpath_lineto_op:nn 1591, 1656, 1669, 1682
.. 1532, 1559, 1569, 1655, 1686, 1799	__draw_softpath_round_loop:Nnn .
\g__draw_softpath_main_tl 1591, 1603, 1611, 1632, 1642,
..... 1470, 1474, 1479,	1657, 1680, 1692, 1698, 1707, 1789
1485, 1490, 1493, 1602, 1604, 1810	__draw_softpath_round_roundpoint:NnnNnnNnn
\l__draw_softpath_main_tl 1591, 1697, 1703
..... 20, 1492, 1502, 1581,	__draw_softpath_roundpoint:nn . .
1596, 1623, 1625, 1806, 1808, 1811 258, 1511, 1552, 1558
\g__draw_softpath_move_bool	__draw_softpath_roundpoint_-
..... 1509, 1538	op:nn
\l__draw_softpath_move_tl 1555, 1559, 1573, 1574, 1614, 1696
..... 1584, 1601,	__draw_softpath_save: 809, 1488, 1488
1624, 1627, 1675, 1777, 1800, 1807	\l__draw_softpath_tmp_tl
__draw_softpath_moveto:nn 1471, 1480, 1481
..... 272, 1511, 1534	__draw_softpath_use: 729, 1476, 1476
__draw_softpath_moveto_op:nn . . .	\l__draw_stroke_color_tl 1415
..... 1537, 1559, 1571, 1621, 1779	\l__draw_tmp_box 13, 49, 60,
\l__draw_softpath_part_tl	64, 66, 67, 68, 69, 85, 87, 88, 89, 90, 91
..... 1582, 1597,	\l__draw_tmp_seq 1824
1626, 1629, 1631, 1665, 1720, 1809	__draw_transform:nnnn
__draw_softpath_rectangle:nnnn 1900, 1904, 1911
..... 601, 1511, 1544	__draw_transform_invert:n
 1953, 1957, 1970, 1981

`__draw_transform_rotate:n` 2021, 2022, 2023, 2029
`__draw_transform_rotate:nn` 2021, 2025, 2030, 2032
`__draw_transform_shift:nn` 1900, 1925, 1927
`__draw_transform_shift:nnnn` 1929, 1934, 1934
`__draw_transform_shift_absolute:nn` 1874, 1893, 1895
`__draw_transform_triangle:nnnnnn` 1990, 1995
`__draw_vec:nn` 1172, 1173, 1175, 1177, 1178
`__draw_vec:nnn` 1172, 1179, 1180
`\l__draw_xmax_dim` 1432, 1439, 1447
`\l__draw_xmin_dim` 1432, 1440, 1448
`\l__draw_xshift_dim` 62, 593, 1245, 1259, 1853, 1869, 1897, 1930, 1936, 1984
`\l__draw_xvec_x_dim` 1166, 1194, 1208, 1226
`\l__draw_xvec_y_dim` 1166, 1195, 1212
`\l__draw_ymax_dim` 1432, 1441, 1449
`\l__draw_ymin_dim` 1432, 1442, 1450
`\l__draw_yshift_dim` 63, 593, 1251, 1259, 1853, 1870, 1898, 1931, 1944, 1985
`\l__draw_yvec_x_dim` 1166, 1194, 1209
`\l__draw_yvec_y_dim` 1166, 1195, 1213, 1227
`\l__draw_zvec_x_dim` 1166, 1210
`\l__draw_zvec_y_dim` 1166, 1214

E

`\end` 192, 827
exp commands:
`\exp_after:wN` 471, 489, 1603, 1677, 1780, 1791, 1800
`\exp_args:Ne` 1840
`\exp_args:NNNV` 1331
`\exp_not:N` 1357, 1358, 1722, 1750, 1759, 1768, 1777, 1780, 1781, 1782, 1787, 1791, 1792, 1793
`\exp_not:n` 1409

F

fp commands:
`\fp_compare:nNnTF` 385, 395, 889
`\fp_compare_p:nNn` 1882, 1883, 1884, 1885
`\fp_const:Nn` 368, 369, 507, 508, 578, 1590

G

group commands:
`\group_begin:` 48, 84, 114, 125, 511, 799, 1308, 1421, 1438, 1595, 1826
`\group_end:` 70, 92, 139, 142, 514, 822, 1348, 1429, 1451, 1607, 1836

H

hbox commands:
`\hbox_gset:Nw` 123
`\hbox_gset_end:` 140
`\hbox_set:Nn` 49, 60, 85, 1367, 1380
`\hbox_set:Nw` 1310, 1325
`\hbox_set_end:` 1331, 1335
`\hbox_to_wd:nn` 1352

I

int commands:
`\int_gincr:N` 1309
`\int_if_odd:nTF` 971, 1008
`\int_new:N` 1298
`\int_use:N` 1358

`\fp_eval:n` 377, 378, 399, 406, 415, 867, 875, 884, 909, 910, 911, 912, 913, 914, 935, 940, 941, 948, 955, 956, 963, 970, 972, 993, 998, 999, 1000, 1007, 1015, 1028, 1034, 1052, 1070, 1075, 1082, 1083, 1105, 1112, 1134, 1135, 1154, 1155, 1156, 1157, 1189, 1202, 1221, 1840, 1905, 1906, 1907, 1908, 1959, 2022, 2026, 2027
`\fp_new:N` 199, 200, 505, 506, 1584, 1585, 1853, 1854, 1855, 1856
`\fp_set:Nn` 336, 337, 391, 392, 472, 473, 1619, 1620, 1667, 1668, 1736, 1737, 1861, 1864, 1876, 1877, 1878, 1879, 1972, 1974, 1976, 1978
`\fp_to_decimal:N` 398, 405, 413
`\fp_to_dim:n` 246, 247, 343, 350, 357, 361, 379, 380, 426, 435, 503, 529, 541, 542, 543, 544, 545, 546, 551, 552, 553, 554, 555, 556, 561, 562, 563, 564, 565, 566, 571, 572, 573, 574, 575, 576, 649, 650, 1401, 1752, 1756, 1761, 1765, 1821, 1829, 1834, 1938, 1946
`\fp_use:N` 54, 55, 56, 57, 578
`\fp_while_do:nNnn` 393
`\fp_zero:N` 1598, 1599, 1862, 1863
`\c_one_fp` 1882, 1885
`\c_zero_fp` 889, 1883, 1884

K		<code>\ProvidesExplPackage</code> 3
kernel internal commands:		
	<code>__kernel_quark_new_test:N</code> 9	
M		Q
mode commands:		quark commands:
	<code>\mode_leave_vertical:</code> 1346	<code>\quark_new:N</code> 7, 8
msg commands:		quark internal commands:
	<code>\msg_error:nnn</code> 100, 131, 132, 721	<code>\q__draw_recursion_stop</code> 7, 1606
	<code>\msg_new:nnn</code> 187	<code>\q__draw_recursion_tail</code> 7, 1605
	<code>\msg_new:nnnn</code> 184, 189, 824	
P		S
<code>\pgfextractx</code> 21		scan commands:
<code>\pgfextracty</code> 21		<code>\scan_new:N</code> 5, 6
<code>\pgfgetlastxy</code> 21		scan internal commands:
<code>\pgfgettransform</code> 51		<code>\s__draw_mark</code> 5,
<code>\pgfgettransformentries</code> 51		843, 845, 850, 853, 860, 864, 1743, 1747
<code>\pgfinnerlinewidth</code> 50		<code>\s__draw_stop</code>
<code>\pgflowlevel</code> 51		.. 5, 836, 838, 843, 845, 850, 853,
<code>\pgflowlevelsynccm</code> 51		860, 864, 1743, 1747, 1783, 1794, 1803
<code>\pgfpatharcto</code> 6		seq commands:
<code>\pgfpatharctoprecomputed</code> 6		<code>\seq_new:N</code> 1838
<code>\pgfpathcosine</code> 6		<code>\seq_set_from_clist:Nn</code> 1827
<code>\pgfpathcurvebetweentime</code> 6		<code>\seq_set_map:NNn</code> 1828
<code>\pgfpathcurvebetweentimecontinue</code> 6		<code>\seq_use:Nn</code> 1833
<code>\pgfpathparabola</code> 6		skip commands:
<code>\pgfpathsine</code> 6		<code>\skip_horizontal:n</code> 1369, 1382
<code>\pgfpointadd</code> 21		str commands:
<code>\pgfpointborderellipse</code> 22		<code>\str_if_eq:nnTF</code>
<code>\pgfpointborderrectangle</code> 22	 99, 117, 130, 148, 165, 177
<code>\pgfpointcylindrical</code> 21		<code>\str_if_eq_p:nn</code> 686
<code>\pgfpointdiff</code> 21		
<code>\pgfpointorigin</code> 21		T
<code>\pgfpointscale</code> 21		TeX and L ^A T _E X 2 _ε commands:
<code>\pgfpointspherical</code> 21		<code>\@expl@finalise@setup@@</code> 1355
<code>\pgfqpoint</code> 22		<code>\savepos</code> 38
<code>\pgfqpointpolar</code> 22		tex commands:
<code>\pgfqpointscale</code> 22		<code>\tex_savepos:D</code> 1354, 1361
<code>\pgfqpointxy</code> 22		tl commands:
<code>\pgfqpointxyz</code> 22		<code>\tl_build_gbegin:N</code> 1485, 1810
<code>\pgfsetinnerlinewidth</code> 50		<code>\tl_build_gend:N</code> 1490, 1602
<code>\pgfsetinnerstrokecolor</code> 50		<code>\tl_build_get_intermediate:NN</code> . 1478
<code>\pgftext</code> 36		<code>\tl_build_gput_right:Nn</code> 1474
<code>\pgftransformarcaxesattime</code> 51		<code>\tl_clear:N</code>
<code>\pgftransformarrow</code> 51		448, 1596, 1597, 1600, 1601, 1628, 1629
<code>\pgftransformationadjustments</code> 51		<code>\tl_if_blank:nTF</code> 679
<code>\pgftransformcurveattime</code> 51		<code>\tl_if_blank_p:n</code> 685
<code>\pgftransformlineattime</code> 51		<code>\tl_if_empty:NTF</code> 1617, 1647
<code>\pgfviewboxscope</code> 51		<code>\tl_if_empty_p:N</code> 1674, 1675
prg commands:		<code>\tl_new:N</code> 106, 198, 1416, 1417, 1470,
	<code>\prg_do_nothing:</code> 1127, 1138, 1141	1471, 1581, 1582, 1583, 1588, 1589
property commands:		<code>\tl_put_right:Nn</code> .. 498, 502, 1623,
	<code>\property_record:nn</code> 1357, 1364	1625, 1631, 1665, 1720, 1806, 1808
		<code>\tl_set:Nn</code> 107,
		121, 494, 1618, 1627, 1648, 1714, 1777
		<code>\tl_set_eq:NN</code> 1491

token commands:		\use:n	51, 338, 374, 421, 526, 1407,
\token_if_eq_meaning:NNTF		1773, 1785, 1830, 1902, 1913, 1997
..	1614, 1621, 1649, 1652, 1655, 1696	\use_i:nn 21
\token_if_eq_meaning_p:NN	\use_i:nnnn 1787
.....	1686, 1687, 1688	\use_ii:nn 21
		\use_none:n 1800
	U		
use commands:			
\use:N		743, 779