

File I

Implementation

1 l3draw implementation

```
1 <*package>
2 <@@=draw>
3 \ProvidesExplPackage{l3draw}{2024-03-14}{}
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:Nnnnnn #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:Nnnnnn #2 {#3} {#4} {#5} {#6}
29     \draw_scope_end:
30 }
31 \cs_new_protected:Npn \__draw_box_use:Nnnnnn #1#2#3#4#5
32 {
33     \bool_if:NT \l_draw_bb_update_bool
34     {
35         \__draw_point_process:nn
36         { \__draw_path_update_limits:nn }
37         { \draw_point_transform:n { #2 , #3 } }
38         \__draw_point_process:nn
39         { \__draw_path_update_limits:nn }
40         { \draw_point_transform:n { #4 , #3 } }
41         \__draw_point_process:nn
42         { \__draw_path_update_limits:nn }
43         { \draw_point_transform:n { #4 , #5 } }
44         \__draw_point_process:nn
45         { \__draw_path_update_limits:nn }
46         { \draw_point_transform:n { #2 , #5 } }
47     }
48     \group_begin:
49     \hbox_set:Nn \l__draw_tmp_box
50     {
51         \use:e
52         {
53             \__draw_backend_box_use:Nnnnn #1
54             { \fp_use:N \l__draw_matrix_a_fp }
55             { \fp_use:N \l__draw_matrix_b_fp }
56             { \fp_use:N \l__draw_matrix_c_fp }
57             { \fp_use:N \l__draw_matrix_d_fp }
58         }
59     }
60     \hbox_set:Nn \l__draw_tmp_box
61     {
62         \__kernel_kern:n { \l__draw_xshift_dim }
63         \box_move_up:nn { \l__draw_yshift_dim }
64         { \box_use_drop:N \l__draw_tmp_box }
65     }
66     \box_set_ht:Nn \l__draw_tmp_box { Opt }
67     \box_set_dp:Nn \l__draw_tmp_box { Opt }
68     \box_set_wd:Nn \l__draw_tmp_box { Opt }
69     \box_use_drop:N \l__draw_tmp_box

```

```

70     \group_end:
71   }

```

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

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

```

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

```

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

```

94 </package>

```

3 l3draw-layers implementation

```

95 <*package>

```

```

96 <@@=draw>

```

3.1 User interface

```

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

```

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

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

```

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

```

(End of definition for `\l__draw_layer_tl`.)

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

```

108 \bool_new:N \l__draw_layer_close_bool

```

(End of definition for `\l__draw_layer_close_bool`.)

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

```

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

```

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

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

```

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

```

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

3.2 Internal cross-links

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

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

(End of definition for __draw_layers_insert:.)

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

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

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

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

```

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

```

4 l3draw-paths implementation

```

195 <*package>
196 <@@=draw>

```

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

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

```

\l__draw_path_tmp_tl Scratch space.
\l__draw_path_tmpa_fp 197 \tl_new:N \l__draw_path_tmp_tl
\l__draw_path_tmpb_fp 198 \fp_new:N \l__draw_path_tmpa_fp
199 \fp_new:N \l__draw_path_tmpb_fp

```

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

4.1 Tracking paths

```

\g__draw_path_lastx_dim The last point visited on a path.
\g__draw_path_lasty_dim 200 \dim_new:N \g__draw_path_lastx_dim
201 \dim_new:N \g__draw_path_lasty_dim

```

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

```

\g__draw_path_xmax_dim The limiting size of a path.
\g__draw_path_xmin_dim 202 \dim_new:N \g__draw_path_xmax_dim
\g__draw_path_ymax_dim 203 \dim_new:N \g__draw_path_xmin_dim
\g__draw_path_ymin_dim 204 \dim_new:N \g__draw_path_ymax_dim
205 \dim_new:N \g__draw_path_ymin_dim

```

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

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

```

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

```

(End of definition for `_draw_path_update_limits:nn` and `_draw_path_reset_limits:.`)

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

```

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

```

(End of definition for `_draw_path_update_last:nn`.)

4.2 Corner arcs

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

`\l__draw_corner_xarc_dim` The two arcs in use.

```

240 \dim_new:N \l__draw_corner_xarc_dim
241 \dim_new:N \l__draw_corner_yarc_dim

```

(End of definition for \l__draw_corner_xarc_dim and \l__draw_corner_yarc_dim.)

\l__draw_corner_arc_bool A flag to speed up the repeated checks.

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

(End of definition for \l__draw_corner_arc_bool.)

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

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

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

__draw_path_mark_corner: Mark up corners for arc post-processing.

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

(End of definition for __draw_path_mark_corner:.)

4.3 Basic path constructions

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

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



```

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

```

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

`\draw_path_close:` A simple wrapper.

```

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

```

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

4.4 Canvas path constructions

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

```

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

```

```

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

```

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

4.5 Computed curves

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

```

\draw_path_curveto:mn
  \_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) .

```

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

```

```

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

```

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

```

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

```

```

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

```

The auxiliary is responsible for calculating the required points. The “magic” number required to determine the length of the control vectors is well-established for a right-angle: $\frac{4}{3}(\sqrt{2} - 1) = 0.55228475$. For other cases, we follow the calculation used by `pgf` but with the second common case of 60° pre-calculated for speed.

```

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

```

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

```

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

```

```

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

```

The first control point.

```

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

```

The end point: simple arithmetic.

```

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

```

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

```

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

```

```

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

```

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

`\draw_path_arc_axes:nnnn` A simple wrapper.

```

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

```

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

`\draw_path_ellipse:nnn` Drawing an ellipse is an optimised version of drawing an arc, in particular reusing the same constant. We need to deal with the ellipse in four parts and also deal with moving to the right place, closing it and ending up back at the center. That is handled on a per-arc basis, each in a separate auxiliary for readability.

```

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

```

```

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

```

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

`\draw_path_circle:nn` A shortcut.

```

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

```

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

4.6 Rectangles

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

```

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

```

```

585         {#1} {#2}
586     }
587     {
588         \__draw_point_process:nnn \__draw_path_rectangle:nnnn
589         { (#1) + ( \l__draw_xshift_dim , \l__draw_yshift_dim ) }
590         { #2 }
591     }
592 }
593 \cs_new_protected:Npn \__draw_path_rectangle:nnnn #1#2#3#4
594 {
595     \__draw_path_update_limits:nn {#1} {#2}
596     \__draw_path_update_limits:nn { #1 + #3 } { #2 + #4 }
597     \__draw_softpath_rectangle:nnnn {#1} {#2} {#3} {#4}
598     \__draw_path_update_last:nn {#1} {#2}
599 }
600 \cs_new_protected:Npn \__draw_path_rectangle_rounded:nnnn #1#2#3#4
601 {
602     \draw_path_moveto:n { #1 + #3 , #2 + #4 }
603     \draw_path_lineto:n { #1 , #2 + #4 }
604     \draw_path_lineto:n { #1 , #2 }
605     \draw_path_lineto:n { #1 + #3 , #2 }
606     \draw_path_close:
607     \draw_path_moveto:n { #1 , #2 }
608 }

```

(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.

```

609 \cs_new_protected:Npn \draw_path_rectangle_corners:nn #1#2
610 {
611     \__draw_point_process:nnn
612     { \__draw_path_rectangle_corners:nnnnn {#1} }
613     {#1} {#2}
614 }
615 \cs_new_protected:Npn \__draw_path_rectangle_corners:nnnnn #1#2#3#4#5
616 { \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_auxi:eennnn`
`__draw_path_grid_auxii:nnnnnn`
`__draw_path_grid_auxiii:nnnnnn`
`__draw_path_grid_auxiiii:eennnn`
`__draw_path_grid_auxiv:nnnnnnnn`
`__draw_path_grid_auxiv:eennnnnn`

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

```

617 \cs_new_protected:Npn \draw_path_grid:nnnn #1#2#3#4
618 {
619     \__draw_point_process:nnn
620     {
621         \__draw_path_grid_auxi:eennnn
622         { \dim_abs:n {#1} }
623         { \dim_abs:n {#2} }
624     }
625     {#3} {#4}

```



```

626 }
627 \cs_new_protected:Npn \__draw_path_grid_auxi:nnnnnn #1#2#3#4#5#6
628 {
629   \dim_compare:nNnTF {#3} > {#5}
630     { \__draw_path_grid_auxiii:nnnnnn {#1} {#2} {#5} {#4} {#3} {#6} }
631     { \__draw_path_grid_auxiii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6} }
632 }
633 \cs_generate_variant:Nn \__draw_path_grid_auxi:nnnnnn { ee }
634 \cs_new_protected:Npn \__draw_path_grid_auxii:nnnnnn #1#2#3#4#5#6
635 {
636   \dim_compare:nNnTF {#4} > {#6}
637     { \__draw_path_grid_auxiiii:nnnnnn {#1} {#2} {#3} {#6} {#5} {#4} }
638     { \__draw_path_grid_auxiiii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6} }
639 }
640 \cs_new_protected:Npn \__draw_path_grid_auxiii:nnnnnn #1#2#3#4#5#6
641 {
642   \__draw_path_grid_auxiv:eennnnnn
643   { \fp_to_dim:n { #1 * ceil(#3/(#1)) } }
644   { \fp_to_dim:n { #2 * ceil(#4/(#2)) } }
645   {#1} {#2} {#3} {#4} {#5} {#6}
646 }
647 \cs_new_protected:Npn \__draw_path_grid_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
648 {
649   \dim_step_inline:nnnn
650   {#1}
651   {#3}
652   {#7}
653   {
654     \draw_path_moveto:n { ##1 , #6 }
655     \draw_path_lineto:n { ##1 , #8 }
656   }
657   \dim_step_inline:nnnn
658   {#2}
659   {#4}
660   {#8}
661   {
662     \draw_path_moveto:n { #5 , ##1 }
663     \draw_path_lineto:n { #7 , ##1 }
664   }
665 }
666 \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

```

\l__draw_path_use_clip_bool Actions to pass to the driver.
\l__draw_path_use_fill_bool
\l__draw_path_use_stroke_bool
667 \bool_new:N \l__draw_path_use_clip_bool
668 \bool_new:N \l__draw_path_use_fill_bool
669 \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.

```
670 \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`

```
671 \cs_new_protected:Npn \draw_path_use:n #1
672 {
673   \tl_if_blank:nF {#1}
674     { \__draw_path_use:n {#1} }
675 }
676 \cs_new_protected:Npn \draw_path_use_clear:n #1
677 {
678   \bool_lazy_or:nnTF
679     { \tl_if_blank_p:n {#1} }
680     { \str_if_eq_p:nn {#1} { clear } }
681     {
682       \__draw_softpath_clear:
683       \__draw_path_reset_limits:
684     }
685     { \__draw_path_use:n { #1 , clear } }
686 }
687 \cs_new_protected:Npn \draw_path_replace_bb:
688 {
689   \__draw_path_replace_bb:NnN x { max } +
690   \__draw_path_replace_bb:NnN y { max } +
691   \__draw_path_replace_bb:NnN x { min } -
692   \__draw_path_replace_bb:NnN y { min } -
693   \__draw_softpath_clear:
694   \__draw_path_reset_limits:
695 }
696 \cs_new_protected:Npn \__draw_path_replace_bb:NnN #1#2#3
697 {
698   \dim_gset:cn { g__draw_ #1#2 _dim }
699   {
700     \dim_use:c { g__draw_path_ #1#2 _dim }
701     #3 0.5 \g__draw_linewidth_dim
702   }
703 }
```

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).

```
704 \cs_new_protected:Npn \__draw_path_use:n #1
705 {
706   \bool_set_false:N \l__draw_path_use_clip_bool
707   \bool_set_false:N \l__draw_path_use_fill_bool
708   \bool_set_false:N \l__draw_path_use_stroke_bool
709   \clist_map_inline:nn {#1}
710   {
711     \cs_if_exist:ctF { l__draw_path_use_ ##1 _ bool }
712     { \bool_set_true:c { l__draw_path_use_ ##1 _ bool } }
```

```

713     {
714         \cs_if_exist_use:cF { __draw_path_use_action_ ##1 : }
715         { \msg_error:nnn { draw } { invalid-path-action } {##1} }
716     }
717 }
718 \__draw_softpath_round_corners:
719 \bool_lazy_and:nnT
720 { \l_draw_bb_update_bool }
721 { \l__draw_path_use_stroke_bool }
722 { \__draw_path_use_stroke_bb: }
723 \__draw_softpath_use:
724 \bool_if:NT \l__draw_path_use_clip_bool
725 {
726     \__draw_backend_clip:
727     \bool_set_false:N \l_draw_bb_update_bool
728     \bool_lazy_or:nnF
729     { \l__draw_path_use_fill_bool }
730     { \l__draw_path_use_stroke_bool }
731     { \__draw_backend_discardpath: }
732 }
733 \bool_lazy_or:nnT
734 { \l__draw_path_use_fill_bool }
735 { \l__draw_path_use_stroke_bool }
736 {
737     \use:c
738     {
739         __draw_backend_
740         \bool_if:NT \l__draw_path_use_fill_bool { fill }
741         \bool_if:NT \l__draw_path_use_stroke_bool { stroke }
742         :
743     }
744 }
745 \bool_if:NT \l__draw_path_use_clear_bool
746 {
747     \__draw_softpath_clear:
748     \__draw_path_reset_limits:
749 }
750 }
751 \cs_new_protected:Npn \__draw_path_use_action_draw:
752 {
753     \bool_set_true:N \l__draw_path_use_stroke_bool
754 }
755 \cs_new_protected:Npn \__draw_path_use_action_fillstroke:
756 {
757     \bool_set_true:N \l__draw_path_use_fill_bool
758     \bool_set_true:N \l__draw_path_use_stroke_bool
759 }

```

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.

```

760 \cs_new_protected:Npn \__draw_path_use_stroke_bb:
761 {
762     \__draw_path_use_bb:NnN x { max } +
763     \__draw_path_use_bb:NnN y { max } +

```

```

764     \__draw_path_use_bb:NnN x { min } -
765     \__draw_path_use_bb:NnN y { min } -
766   }
767 \cs_new_protected:Npn \__draw_path_use_bb:NnN #1#2#3
768 {
769   \dim_compare:nNnF { \dim_use:c { g__draw_ #1#2 _dim } } = { #3 -\c_max_dim }
770   {
771     \dim_gset:cn { g__draw_ #1#2 _dim }
772     {
773       \use:c { dim_ #2 :nn }
774       { \dim_use:c { g__draw_ #1#2 _dim } }
775       {
776         \dim_use:c { g__draw_path_ #1#2 _dim }
777         #3 0.5 \g__draw_linewidth_dim
778       }
779     }
780   }
781 }

```

(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_xmax_dim 782 \dim_new:N \l__draw_path_lastx_dim
\l__draw_path_xmin_dim 783 \dim_new:N \l__draw_path_lasty_dim
\l__draw_path_ymax_dim 784 \dim_new:N \l__draw_path_xmax_dim
\l__draw_path_ymin_dim 785 \dim_new:N \l__draw_path_xmin_dim
\l__draw_softpath_corners_bool 786 \dim_new:N \l__draw_path_ymax_dim
787 \dim_new:N \l__draw_path_ymin_dim
788 \dim_new:N \l__draw_softpath_lastx_dim
789 \dim_new:N \l__draw_softpath_lasty_dim
790 \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:
791 \cs_new_protected:Npn \draw_path_scope_begin:
792 {
793   \group_begin:
794     \dim_set_eq:NN \l__draw_path_lastx_dim \g__draw_path_lastx_dim
795     \dim_set_eq:NN \l__draw_path_lasty_dim \g__draw_path_lasty_dim
796     \dim_set_eq:NN \l__draw_path_xmax_dim \g__draw_path_xmax_dim
797     \dim_set_eq:NN \l__draw_path_xmin_dim \g__draw_path_xmin_dim
798     \dim_set_eq:NN \l__draw_path_ymax_dim \g__draw_path_ymax_dim
799     \dim_set_eq:NN \l__draw_path_ymin_dim \g__draw_path_ymin_dim
800     \dim_set_eq:NN \l__draw_softpath_lastx_dim \g__draw_softpath_lastx_dim
801     \dim_set_eq:NN \l__draw_softpath_lasty_dim \g__draw_softpath_lasty_dim
802     \__draw_path_reset_limits:
803     \__draw_softpath_save:
804   }

```

```

805 \cs_new_protected:Npn \draw_path_scope_end:
806 {
807   \__draw_softpath_restore:
808   \dim_gset_eq:NN \g__draw_softpath_lastx_dim \l__draw_softpath_lastx_dim
809   \dim_gset_eq:NN \g__draw_softpath_lasty_dim \l__draw_softpath_lasty_dim
810   \dim_gset_eq:NN \g__draw_path_xmax_dim \l__draw_path_xmax_dim
811   \dim_gset_eq:NN \g__draw_path_xmin_dim \l__draw_path_xmin_dim
812   \dim_gset_eq:NN \g__draw_path_ymax_dim \l__draw_path_ymax_dim
813   \dim_gset_eq:NN \g__draw_path_ymin_dim \l__draw_path_ymin_dim
814   \dim_gset_eq:NN \g__draw_path_lastx_dim \l__draw_path_lastx_dim
815   \dim_gset_eq:NN \g__draw_path_lasty_dim \l__draw_path_lasty_dim
816   \group_end:
817 }

```

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

4.10 Messages

```

818 \msg_new:nmmn { draw } { invalid-path-action }
819 { Invalid-action-#1'-for-path. }
820 { Paths-can-be-used-with-actions-'draw',~'clip',~'fill'~or~'stroke'. }
821 %   \end{macrocode}
822 %
823 %   \begin{macrocode}
824 </package>

```

5 I3draw-points implementation

```

825 <*package>
826 <@@=draw>

```

This sub-module covers more-or-less the same ideas as `pgfcorepoints.code.tex`, though the approach taken to returning values is different: point expressions here are processed by expansion and return a co-ordinate pair in the form $\langle x \rangle \langle y \rangle$. Equivalents of following `pgf` functions are deliberately omitted:

- `\pgfpointorigin`: Can be given explicitly as `Opt,Opt`.
- `\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`, `\pgfpointospherical`: The usefulness of these commands is not currently clear.
- `\pgfpointborderrectangle`, `\pgfpointborderellipse`: To be revisited once the semantics and use cases are clear.

- `\pgfqpoint`, `\pgfqpointscale`, `\pgfqpointpolar`, `\pgfqpointxy`, `\pgfqpointxyz`: The expandable approach taken in the code here, along with the absolute requirement for ε -TeX, means it is likely many use cases for these commands may be covered in other ways. This may be revisited as higher-level structures are constructed.

5.1 Support functions

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

```

\__draw_point_process:nn
  \__draw_point_process_auxi:nn
  \__draw_point_process_auxi:en
  \__draw_point_process_auxii:nw
\__draw_point_process:nnn
  \__draw_point_process_auxiii:nn
  \__draw_point_process_auxiii:een
  \__draw_point_process_auxiv:nw
\__draw_point_process:nnnn
  \__draw_point_process_auxv:nnnn
  \__draw_point_process_auxv:eeen
  \__draw_point_process_auxvi:nw
\__draw_point_process:nnnnn
  \__draw_point_process_auxvii:nnnnn
  \__draw_point_process_auxvii:eeeen
  \__draw_point_process_auxviii:nw
827 \cs_new:Npn \__draw_point_process:nn #1#2
828 {
829   \__draw_point_process_auxi:en
830   { \draw_point:n {#2} }
831   {#1}
832 }
833 \cs_new:Npn \__draw_point_process_auxi:nn #1#2
834 { \__draw_point_process_auxii:nw {#2} #1 \s__draw_stop }
835 \cs_generate_variant:Nn \__draw_point_process_auxi:nn { e }
836 \cs_new:Npn \__draw_point_process_auxii:nw #1 #2 , #3 \s__draw_stop
837 { #1 {#2} {#3} }
838 \cs_new:Npn \__draw_point_process:nnn #1#2#3
839 {
840   \__draw_point_process_auxiii:een
841   { \draw_point:n {#2} }
842   { \draw_point:n {#3} }
843   {#1}
844 }
845 \cs_new:Npn \__draw_point_process_auxiii:nnn #1#2#3
846 { \__draw_point_process_auxiv:nw {#3} #1 \s__draw_mark #2 \s__draw_stop }
847 \cs_generate_variant:Nn \__draw_point_process_auxiii:nnn { ee }
848 \cs_new:Npn \__draw_point_process_auxiv:nw #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_stop
849 { #1 {#2} {#3} {#4} {#5} }
850 \cs_new:Npn \__draw_point_process:nnnn #1#2#3#4
851 {
852   \__draw_point_process_auxv:eeen
853   { \draw_point:n {#2} }
854   { \draw_point:n {#3} }
855   { \draw_point:n {#4} }
856   {#1}
857 }
858 \cs_new:Npn \__draw_point_process_auxv:nnnn #1#2#3#4
859 { \__draw_point_process_auxvi:nw {#4} #1 \s__draw_mark #2 \s__draw_mark #3 \s__draw_stop }
860 \cs_generate_variant:Nn \__draw_point_process_auxv:nnnn { eee }
861 \cs_new:Npn \__draw_point_process_auxvi:nw
862   #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_mark #6 , #7 \s__draw_stop
863   { #1 {#2} {#3} {#4} {#5} {#6} {#7} }
864 \cs_new:Npn \__draw_point_process:nnnnn #1#2#3#4#5
865 {
866   \__draw_point_process_auxvii:eeeen
867   { \draw_point:n {#2} }
868   { \draw_point:n {#3} }

```

```

869     { \draw_point:n {#4} }
870     { \draw_point:n {#5} }
871     {#1}
872   }
873 \cs_new:Npn \__draw_point_process_auxvii:nnnnn #1#2#3#4#5
874 {
875   \__draw_point_process_auxviii:nw
876   {#5} #1 \s__draw_mark #2 \s__draw_mark #3 \s__draw_mark #4 \s__draw_stop
877 }
878 \cs_generate_variant:Nn \__draw_point_process_auxvii:nnnnn { eeee }
879 \cs_new:Npn \__draw_point_process_auxviii:nw
880 #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_mark #6 , #7 \s__draw_mark #8 , #9 \s__draw_stop
881 { #1 {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9} }

```

(End of definition for `__draw_point_process:nn` and others.)

5.2 Basic points

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

```

\__draw_point_to_dim:n 882 \cs_new:Npn \draw_point:n #1
\__draw_point_to_dim:e 883 { \__draw_point_to_dim:e { \fp_eval:n {#1} } }
\__draw_point_to_dim:w 884 \cs_new:Npn \__draw_point_to_dim:n #1
885 { \__draw_point_to_dim:w #1 }
886 \cs_generate_variant:Nn \__draw_point_to_dim:n { e }
887 \cs_new:Npn \__draw_point_to_dim:w ( #1 , ~ #2 ) { #1pt , #2pt }

```

5.3 Polar co-ordinates

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

```

\draw_point_polar:nn 888 \cs_new:Npn \draw_point_polar:nn #1#2
\draw_point_polar:nnn 889 { \draw_point_polar:nnn {#1} {#1} {#2} }
\__draw_draw_polar:nnn 890 \cs_new:Npn \draw_point_polar:nnn #1#2#3
\__draw_draw_polar:enn 891 { \__draw_draw_polar:enn { \fp_eval:n {#3} } {#1} {#2} }
892 \cs_new:Npn \__draw_draw_polar:nnn #1#2#3
893 { \draw_point:n { cosd(#1) * (#2) , sind(#1) * (#3) } }
894 \cs_generate_variant:Nn \__draw_draw_polar:nnn { e }

```

5.4 Point expression arithmetic

These functions all take point expressions as arguments.

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

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

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

```

895 \cs_new:Npn \draw_point_unit_vector:n #1
896 { \__draw_point_process:nn { \__draw_point_unit_vector:n } {#1} }
897 \cs_new:Npn \__draw_point_unit_vector:nn #1#2
898 {

```

```

899   \__draw_point_unit_vector:nnn
900     { \fp_eval:n { (sqrt(#1 * #1 + #2 * #2)) } }
901     {#1} {#2}
902   }
903 \cs_new:Npn \__draw_point_unit_vector:nnn #1#2#3
904   {
905     \fp_compare:nNnTF {#1} = \c_zero_fp
906       { Opt, 1pt }
907       {
908         \draw_point:n
909           { ( #2 , #3 ) / #1 }
910       }
911   }
912 \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.)

```

913 \cs_new:Npn \draw_point_intersect_lines:nnnn #1#2#3#4
914   {
915     \__draw_point_process:nnnnn
916       { \__draw_point_intersect_lines:nnnnnnn }
917       {#1} {#2} {#3} {#4}
918   }

```

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

```

#1 x1
#2 y1
#3 x2
#4 y2
#5 x3
#6 y3
#7 x4
#8 y4

```


so now just have to do all of the calculation.

```

919 \cs_new:Npn \__draw_point_intersect_lines:nnnnnnn #1#2#3#4#5#6#7#8
920 {
921   \__draw_point_intersect_lines_aux:eeeeee
922   { \fp_eval:n { #1 * #4 - #2 * #3 } }
923   { \fp_eval:n { #5 * #8 - #6 * #7 } }
924   { \fp_eval:n { #1 - #3 } }
925   { \fp_eval:n { #5 - #7 } }
926   { \fp_eval:n { #2 - #4 } }
927   { \fp_eval:n { #6 - #8 } }
928 }
929 \cs_new:Npn \__draw_point_intersect_lines_aux:nnnnn #1#2#3#4#5#6
930 {
931   \draw_point:n
932   {
933     ( #2 * #3 - #1 * #4 , #2 * #5 - #1 * #6 )
934     / ( #4 * #5 - #6 * #3 )
935   }
936 }
937 \cs_generate_variant:Nn \__draw_point_intersect_lines_aux:nnnnn { 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:nnmn
__draw_point_intersect_circles_auxi:nnnnnnn
__draw_point_intersect_circles_auxii:nnnnnnn
__draw_point_intersect_circles_auxiii:eeennnnn
__draw_point_intersect_circles_auxiiii:nnnnnnn
__draw_point_intersect_circles_auxv:nnnnnnnnn
__draw_point_intersect_circles_auxvi:nnnnnnnnn
__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.

```

938 \cs_new:Npn \draw_point_intersect_circles:nnnnn #1#2#3#4#5
939 {
940   \__draw_point_process:nnn
941   { \__draw_point_intersect_circles_auxi:nnnnnnn {#2} {#4} {#5} }
942   {#1} {#3}

```

```

943 }
944 \cs_new:Npn \__draw_point_intersect_circles_auxi:nnnnnnn #1#2#3#4#5#6#7
945 {
946   \__draw_point_intersect_circles_auxii:eennnnn
947   { \fp_eval:n {#1} } { \fp_eval:n {#2} } {#4} {#5} {#6} {#7} {#3}
948 }

```

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

```

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

```

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

```

949 \cs_new:Npn \__draw_point_intersect_circles_auxii:nnnnnnn #1#2#3#4#5#6#7
950 {
951   \__draw_point_intersect_circles_auxiii:eennnnn
952   { \fp_eval:n { #5 - #3 } }
953   { \fp_eval:n { #6 - #4 } }
954   {#1} {#2} {#3} {#4} {#7}
955 }
956 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxii:nnnnnnn { ee }
957 \cs_new:Npn \__draw_point_intersect_circles_auxiii:nnnnnnn #1#2#3#4#5#6#7
958 {
959   \__draw_point_intersect_circles_auxiv:ennnnnnn
960   { \fp_eval:n { sqrt( #1 * #1 + #2 * #2 ) } }
961   {#1} {#2} {#3} {#4} {#5} {#6} {#7}
962 }
963 \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.

```

964 \cs_new:Npn \__draw_point_intersect_circles_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
965 {
966   \__draw_point_intersect_circles_auxv:ennnnnnnn
967   { \fp_eval:n { 1 / #1 } }
968   { \fp_eval:n { #4 * #4 } }
969   {#1} {#2} {#3} {#5} {#6} {#7} {#8}
970 }
971 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxiv:nnnnnnnn { e }
972 \cs_new:Npn \__draw_point_intersect_circles_auxv:nnnnnnnnn #1#2#3#4#5#6#7#8#9
973 {
974   \__draw_point_intersect_circles_auxvi:ennnnnnnn
975   { \fp_eval:n { 0.5 * #1 * ( #2 + #3 * #3 - #6 * #6 ) } }
976   {#1} {#2} {#4} {#5} {#7} {#8} {#9}
977 }
978 \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.

```

979 \cs_new:Npn \__draw_point_intersect_circles_auxvi:nnnnnnnn #1#2#3#4#5#6#7#8
980   {
981     \__draw_point_intersect_circles_auxvii:eeennnn
982     { \fp_eval:n { #1 * #2 } }
983     { \int_if_odd:nTF {#8} { 1 } { -1 } }
984     { \fp_eval:n { sqrt ( #3 - #1 * #1 ) * #2 } }
985     {#4} {#5} {#6} {#7}
986   }
987 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxvi:nnnnnnnn { e }
988 \cs_new:Npn \__draw_point_intersect_circles_auxvii:nnnnnnnn #1#2#3#4#5#6#7
989   {
990     \draw_point:n
991     { #6 + #4 * #1 + #2 * #3 * #5 , #7 + #5 * #1 + -1 * #2 * #3 * #4 }
992   }
993 \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:nnmn
w_point_intersect_line_circle_auxi:nnnnnnnn
_point_intersect_line_circle_auxii:nnnnnnnn
_point_intersect_line_circle_auxiii:ennnnnnnn
_point_intersect_line_circle_auxiiii:nnnnnnnn
_point_intersect_line_circle_auxv:eeennnnnn
_point_intersect_line_circle_auxvi:nnnnnnnn
_point_intersect_line_circle_auxvii:eeennnnnn
draw_point_intersect_line_circle_auxviii:nnnnn
draw_point_intersect_line_circle_auxviiii: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

$$P_{2x} = x_1 + \mu_2 \times (x_2 - x_1)$$

$$P_{2y} = y_1 + \mu_2 \times (y_2 - y_1)$$

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

```

994 \cs_new:Npn \draw_point_intersect_line_circle:nnnnn #1#2#3#4#5
995 {
996   \__draw_point_process:nnnn
997   { \__draw_point_intersect_line_circle_auxi:nnnnnnnn {#4} {#5} }
998   {#1} {#2} {#3}
999 }
1000 \cs_new:Npn \__draw_point_intersect_line_circle_auxi:nnnnnnnn #1#2#3#4#5#6#7#8
1001 {
1002   \__draw_point_intersect_line_circle_auxii:ennnnnnnn
1003   { \fp_eval:n {#1} } {#3} {#4} {#5} {#6} {#7} {#8} {#2}
1004 }

```

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

```

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

```

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

```

1005 \cs_new:Npn \__draw_point_intersect_line_circle_auxii:nnnnnnnn #1#2#3#4#5#6#7#8
1006 {
1007   \__draw_point_intersect_line_circle_auxiii:eeennnnnn
1008   { \fp_eval:n { (#4-#2)*(#4-#2)+(#5-#3)*(#5-#3) } }
1009   { \fp_eval:n { 2*((#4-#2)*(#2-#6)+(#5-#3)*(#3-#7)) } }
1010   { \fp_eval:n { (#6*#6+#7*#7)+(#2*#2+#3*#3)-(2*(#6*#2+#7*#3))-(#1*#1) } }
1011   {#2} {#3} {#4} {#5} {#8}
1012 }
1013 \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 .

```

1014 \cs_new:Npn \__draw_point_intersect_line_circle_auxiii:nnnnnnnn #1#2#3#4#5#6#7#8
1015 {
1016   \__draw_point_intersect_line_circle_auxiv:eennnnnnn
1017   { \fp_eval:n { #2 * #2 - 4 * #1 * #3 } }
1018   { \int_if_odd:nTF {#8} { 1 } { -1 } }
1019   {#1} {#2} {#4} {#5} {#6} {#7}
1020 }
1021 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxiii:nnnnnnnn { eee }

```

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

```

#1 a
#2 b
#3 c
#4 d
#5 ±(the usage of n)
#6 x1
#7 y1
#8 x2
#9 y2

```

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

```

1022 \cs_new:Npn \__draw_point_intersect_line_circle_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
1023   {
1024     \__draw_point_intersect_line_circle_auxv:ennnn
1025     { \fp_eval:n { (-1 * #4 + #2 * sqrt(#1)) / (2 * #3) } }
1026     {#5} {#6} {#7} {#8}
1027   }
1028 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxiv:nnnnnnnn { ee }
1029 \cs_new:Npn \__draw_point_intersect_line_circle_auxv:nnnnn #1#2#3#4#5
1030   {
1031     \draw_point:n
1032     { #2 + #1 * (#4 - #2), #3 + #1 * (#5 - #3) }
1033   }
1034 \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 1035 \cs_new:Npn \draw_point_interpolate_line:nnn #1#2#3
\_draw_point_interpolate_line_aux:nnnnn 1036   {
\_draw_point_interpolate_line_aux:ennnn 1037     \__draw_point_process:nnn
\_draw_point_interpolate_line_aux:nnnnnn 1038     { \__draw_point_interpolate_line_aux:ennnn { \fp_eval:n {#1} } }
\_draw_point_interpolate_line_aux:ennnnnn 1039     {#2} {#3}
1040   }
1041 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnn #1#2#3#4#5
1042   {
1043     \__draw_point_interpolate_line_aux:ennnnn { \fp_eval:n { 1 - #1 } }
1044     {#1} {#2} {#3} {#4} {#5}
1045   }
1046 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnn { e }
1047 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnnn #1#2#3#4#5#6
1048   { \draw_point:n { #2 * #3 + #1 * #5 , #2 * #4 + #1 * #6 } }
1049 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnnn { e }

```

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

```

\draw_point_interpolate_distance:nnn
\__draw_point_interpolate_distance:nnnnn
\__draw_point_interpolate_distance:nnnnnn
\__draw_point_interpolate_distance:ennnnn
1050 \cs_new:Npn \draw_point_interpolate_distance:nnn #1#2#3
1051 {
1052   \__draw_point_process:nn
1053   { \__draw_point_interpolate_distance:nnnn {#1} {#3} }
1054   {#2}
1055 }
1056 \cs_new:Npn \__draw_point_interpolate_distance:nnnn #1#2#3#4
1057 {
1058   \__draw_point_process:nn
1059   {
1060     \__draw_point_interpolate_distance:ennnn
1061     { \fp_eval:n {#1} } {#3} {#4}
1062   }
1063   { \draw_point_unit_vector:n { ( #2 ) - ( #3 , #4 ) } }
1064 }
1065 \cs_new:Npn \__draw_point_interpolate_distance:nnnnn #1#2#3#4#5
1066 { \draw_point:n { #2 + #1 * #4 , #3 + #1 * #5 } }
1067 \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:nnnnnnnnn
\__draw_point_interpolate_arcaxes_auxii:nnnnnnnnn
\__draw_point_interpolate_arcaxes_auxii:ennnnnnnnn
\__draw_point_interpolate_arcaxes_auxiii:nnnnnnnnn
\__draw_point_interpolate_arcaxes_auxiii:ennnnnnnnn
\__draw_point_interpolate_arcaxes_auxiv:nnnnnnnnn
\__draw_point_interpolate_arcaxes_auxiv:ennnnnnnnn

```

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

```

1068 \cs_new:Npn \draw_point_interpolate_arcaxes:nnnnnn #1#2#3#4#5#6
1069 {
1070   \__draw_point_process:nnnnn
1071   { \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnn {#1} {#5} {#6} }
1072   {#2} {#3} {#4}
1073 }
1074 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnn #1#2#3#4#5#6#7#8#9
1075 {
1076   \__draw_point_interpolate_arcaxes_auxii:ennnnnnnnn
1077   { \fp_eval:n {#1} } {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1078 }

```

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

- #1 p
- #2 θ_1
- #3 θ_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.

```

1079 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnn #1#2#3#4#5#6#7#8#9
1080 {
1081   \__draw_point_interpolate_arcaxes_auxiii:ennnnnn
1082   { \fp_eval:n { #1 * (#3) + ( 1 - #1 ) * (#2) } }
1083   {#4} {#5} {#6} {#7} {#8} {#9}
1084 }
1085 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnn { e }
1086 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnn #1#2#3#4#5#6#7
1087 {
1088   \__draw_point_interpolate_arcaxes_auxiv:eennnnnn
1089   { \fp_eval:n { cosd (#1) } }
1090   { \fp_eval:n { sind (#1) } }
1091   {#2} {#3} {#4} {#5} {#6} {#7}
1092 }
1093 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnn { e }
1094 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
1095 {
1096   \draw_point:n
1097   { #3 + #1 * #5 + #2 * #7 , #4 + #1 * #6 + #2 * #8 }
1098 }
1099 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnn { ee }

```

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

```

\draw_point_interpolate_curve:nnnnn
draw_point_interpolate_curve_auxi:nnnnnnnnn
raw_point_interpolate_curve_auxii:nnnnnnnnn
\draw_point_interpolate_curve_auxiii:nnnnnnn
\draw_point_interpolate_curve_auxiiii:ennnnnn
\draw_point_interpolate_curve_auxiv:nnnnnnn
\draw_point_interpolate_curve_auxv:nnw
\draw_point_interpolate_curve_auxv:eev
\draw_point_interpolate_curve_auxvi:n
raw_point_interpolate_curve_auxvii:nnnnnnnnn
draw_point_interpolate_curve_auxviii:nnnnnnn
draw_point_interpolate_curve_auxviii:eennnn

```

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.

```

1100 \cs_new:Npn \draw_point_interpolate_curve:nnnnnn #1#2#3#4#5
1101 {
1102   \__draw_point_process:nnnnn
1103   { \__draw_point_interpolate_curve_auxi:nnnnnnnnn {#1} }
1104   {#2} {#3} {#4} {#5}
1105 }
1106 \cs_new:Npn \__draw_point_interpolate_curve_auxi:nnnnnnnnn #1#2#3#4#5#6#7#8#9
1107 {
1108   \__draw_point_interpolate_curve_auxii:ennnnnnnn
1109   { \fp_eval:n {#1} }
1110   {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1111 }

```

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

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

In the second stage, we can drop the final point

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

and for the final stage only need one set of calculations

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

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

```

1112 \cs_new:Npn \__draw_point_interpolate_curve_auxii:nnnnnnnn
1113   #1#2#3#4#5#6#7#8#9
1114   {
1115     \__draw_point_interpolate_curve_auxiii:ennnnn
1116     { \fp_eval:n { 1 - #1 } }
1117     {#1}
1118     { {#2} {#3} } { {#4} {#5} } { {#6} {#7} } { {#8} {#9} }
1119   }
1120 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxii:nnnnnnnn { e }
1121 % \begin{macrocode}
1122 % We need to do the first cycle, but haven't got enough arguments to keep
1123 % everything in play at once. So here we use a bit of argument re-ordering
1124 % and a single auxiliary to get the job done.
1125 % \begin{macrocode}
1126 \cs_new:Npn \__draw_point_interpolate_curve_auxiii:nnnnnn #1#2#3#4#5#6
1127   {
1128     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #3 #4
1129     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #4 #5
1130     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #5 #6
1131     \prg_do_nothing:
1132     \__draw_point_interpolate_curve_auxvi:n { {#1} {#2} }
1133   }
1134 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxiii:nnnnnn { e }
1135 \cs_new:Npn \__draw_point_interpolate_curve_auxiv:nnnnnn #1#2#3#4#5#6
1136   {
1137     \__draw_point_interpolate_curve_auxv:eev
1138     { \fp_eval:n { #1 * #3 + #2 * #5 } }

```



```

1139     { \fp_eval:n { #1 * #4 + #2 * #6 } }
1140   }
1141 \cs_new:Npn \__draw_point_interpolate_curve_auxv:nnw
1142   #1#2#3 \prg_do_nothing: #4#5
1143   {
1144     #3
1145     \prg_do_nothing:
1146     #4 { #5 {#1} {#2} }
1147   }
1148 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxv:nnw { ee }
1149 %   \begin{macrocode}
1150 %   Get the arguments back into the right places and to the second and
1151 %   third cycles directly.
1152 %   \begin{macrocode}
1153 \cs_new:Npn \__draw_point_interpolate_curve_auxvi:n #1
1154   { \__draw_point_interpolate_curve_auxvii:nnnnnnn #1 }
1155 \cs_new:Npn \__draw_point_interpolate_curve_auxvii:nnnnnnn #1#2#3#4#5#6#7#8
1156   {
1157     \__draw_point_interpolate_curve_auxviii:eeeenn
1158     { \fp_eval:n { #1 * #5 + #2 * #3 } }
1159     { \fp_eval:n { #1 * #6 + #2 * #4 } }
1160     { \fp_eval:n { #1 * #7 + #2 * #5 } }
1161     { \fp_eval:n { #1 * #8 + #2 * #6 } }
1162     {#1} {#2}
1163   }
1164 \cs_new:Npn \__draw_point_interpolate_curve_auxviii:nnnnnn #1#2#3#4#5#6
1165   {
1166     \draw_point:n
1167     { #5 * #3 + #6 * #1 , #5 * #4 + #6 * #2 }
1168   }
1169 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxviii:nnnnnn { eeee }

```

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

5.7 Vector support

As well as co-ordinates relative to the drawing

```

\l__draw_xvec_x_dim Base vectors to map to the underlying two-dimensional drawing space.
\l__draw_xvec_y_dim 1170 \dim_new:N \l__draw_xvec_x_dim
\l__draw_yvec_x_dim 1171 \dim_new:N \l__draw_xvec_y_dim
\l__draw_yvec_y_dim 1172 \dim_new:N \l__draw_yvec_x_dim
\l__draw_zvec_x_dim 1173 \dim_new:N \l__draw_yvec_y_dim
\l__draw_zvec_y_dim 1174 \dim_new:N \l__draw_zvec_x_dim
1175 \dim_new:N \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 1176 \cs_new_protected:Npn \draw_xvec:n #1
\draw_zvec:n 1177 { \__draw_vec:nn { x } {#1} }
\__draw_vec:nn 1178 \cs_new_protected:Npn \draw_yvec:n #1
\__draw_vec:nnn 1179 { \__draw_vec:nn { y } {#1} }

```

```

1180 \cs_new_protected:Npn \draw_zvec:n #1
1181 { \__draw_vec:nn { z } {#1} }
1182 \cs_new_protected:Npn \__draw_vec:nn #1#2
1183 {
1184   \__draw_point_process:nn { \__draw_vec:nnn {#1} } {#2}
1185 }
1186 \cs_new_protected:Npn \__draw_vec:nnn #1#2#3
1187 {
1188   \dim_set:cn { l__draw_ #1 vec_x_dim } {#2}
1189   \dim_set:cn { l__draw_ #1 vec_y_dim } {#3}
1190 }

```

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

Initialise the vectors.

```

1191 \draw_xvec:n { 1cm , 0cm }
1192 \draw_yvec:n { 0cm , 1cm }
1193 \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 1194 \cs_new:Npn \draw_point_vec:nn #1#2
\__draw_point_vec:ee 1195 { \__draw_point_vec:ee { \fp_eval:n {#1} } { \fp_eval:n {#2} } }
\draw_point_vec:nnn 1196 \cs_new:Npn \__draw_point_vec:nn #1#2
\__draw_point_vec:nnn 1197 {
\__draw_point_vec:eee 1198   \draw_point:n
1199   {
1200     #1 * \l__draw_xvec_x_dim + #2 * \l__draw_yvec_x_dim ,
1201     #1 * \l__draw_xvec_y_dim + #2 * \l__draw_yvec_y_dim
1202   }
1203 }
1204 \cs_generate_variant:Nn \__draw_point_vec:nn { ee }
1205 \cs_new:Npn \draw_point_vec:nnn #1#2#3
1206 {
1207   \__draw_point_vec:eee
1208   { \fp_eval:n {#1} } { \fp_eval:n {#2} } { \fp_eval:n {#3} }
1209 }
1210 \cs_new:Npn \__draw_point_vec:nnn #1#2#3
1211 {
1212   \draw_point:n
1213   {
1214     #1 * \l__draw_xvec_x_dim
1215     + #2 * \l__draw_yvec_x_dim
1216     + #3 * \l__draw_zvec_x_dim
1217     ,
1218     #1 * \l__draw_xvec_y_dim
1219     + #2 * \l__draw_yvec_y_dim
1220     + #3 * \l__draw_zvec_y_dim
1221   }
1222 }
1223 \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 1224 \cs_new:Npn \draw_point_vec_polar:nn #1#2
\__draw_point_vec_polar:nnn
\__draw_point_vec_polar:enn

```

```

1225 { \draw_point_vec_polar:nnn {#1} {#1} {#2} }
1226 \cs_new:Npn \draw_point_vec_polar:nnn #1#2#3
1227 { \__draw_draw_vec_polar:enn { \fp_eval:n {#3} } {#1} {#2} }
1228 \cs_new:Npn \__draw_draw_vec_polar:nnn #1#2#3
1229 {
1230   \draw_point:n
1231   {
1232     cosd(#1) * (#2) * \l__draw_xvec_x_dim ,
1233     sind(#1) * (#3) * \l__draw_yvec_y_dim
1234   }
1235 }
1236 \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.

```

1237 \cs_new:Npn \draw_point_transform:n #1
1238 {
1239   \__draw_point_process:nn
1240   { \__draw_point_transform:nn } {#1}
1241 }
1242 \cs_new:Npn \__draw_point_transform:nn #1#2
1243 {
1244   \bool_if:NTF \l__draw_matrix_active_bool
1245   {
1246     \draw_point:n
1247     {
1248       (
1249         \l__draw_matrix_a_fp * #1
1250         + \l__draw_matrix_c_fp * #2
1251         + \l__draw_xshift_dim
1252       )
1253       ,
1254       (
1255         \l__draw_matrix_b_fp * #1
1256         + \l__draw_matrix_d_fp * #2
1257         + \l__draw_yshift_dim
1258       )
1259     }
1260   }
1261   {
1262     \draw_point:n
1263     {
1264       (#1, #2)
1265       + ( \l__draw_xshift_dim , \l__draw_yshift_dim )
1266     }
1267   }
1268 }

```

(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
1269 \cs_new:Npn \__draw_point_transform_noshift:n #1
1270 {
1271   \__draw_point_process:nn
1272   { \__draw_point_transform_noshift:nn } {#1}
1273 }
1274 \cs_new:Npn \__draw_point_transform_noshift:nn #1#2
1275 {
1276   \bool_if:NTF \l__draw_matrix_active_bool
1277   {
1278     \draw_point:n
1279     {
1280       (
1281         \l__draw_matrix_a_fp * #1
1282         + \l__draw_matrix_c_fp * #2
1283       )
1284       ,
1285       (
1286         \l__draw_matrix_b_fp * #1
1287         + \l__draw_matrix_d_fp * #2
1288       )
1289     }
1290   }
1291   { \draw_point:n { (#1, #2) } }
1292 }

```

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

```
1293 </package>
```

6 I3draw-scopes implementation

```
1294 <*package>
```

```
1295 <@@=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_xmax_dim Used to track the overall (official) size of the image created: may not actually be the
\g__draw_xmin_dim natural size of the content.
```

```
\g__draw_ymax_dim 1296 \dim_new:N \g__draw_xmax_dim
```

```
\g__draw_ymin_dim 1297 \dim_new:N \g__draw_xmin_dim
```

```
1298 \dim_new:N \g__draw_ymax_dim
```

```
1299 \dim_new:N \g__draw_ymin_dim
```

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

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

```
1300 \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.

```
1301 \box_new:N \l__draw_main_box
1302 \box_new:N \l__draw_layer_main_box
```

(End of definition for `\l__draw_layer_main_box`.)

`\g__draw_id_int` The drawing number.

```
1303 \int_new:N \g__draw_id_int
```

(End of definition for `\g__draw_id_int`.)

`__draw_reset_bb:` A simple auxiliary.

```
1304 \cs_new_protected:Npn \__draw_reset_bb:
1305 {
1306   \dim_gset:Nn \g__draw_xmax_dim { -\c_max_dim }
1307   \dim_gset:Nn \g__draw_xmin_dim { \c_max_dim }
1308   \dim_gset:Nn \g__draw_ymax_dim { -\c_max_dim }
1309   \dim_gset:Nn \g__draw_ymin_dim { \c_max_dim }
1310 }
```

(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

```
1311 \cs_new_protected:Npn \draw_begin:
1312 {
1313   \group_begin:
1314     \int_gincr:N \g__draw_id_int
1315     \hbox_set:Nw \l__draw_main_box
1316     \__draw_backend_begin:
1317     \__draw_reset_bb:
1318     \__draw_path_reset_limits:
1319     \bool_set_true:N \l_draw_bb_update_bool
1320     \draw_transform_matrix_reset:
1321     \draw_transform_shift_reset:
1322     \__draw_softpath_clear:
1323     \draw_linewidth:n { \l_draw_default_linewidth_dim }
1324     \color_select:n { . }
1325     \draw_nonzero_rule:
1326     \draw_cap_but:
1327     \draw_join_miter:
1328     \draw_miterlimit:n { 10 }
1329     \draw_dash_pattern:nn { } { 0cm }
1330     \hbox_set:Nw \l__draw_layer_main_box
1331 }
1332 \cs_new_protected:Npn \draw_end:
1333 {
1334   \__draw_baseline_finalise:w
```

```

1335         \exp_args:NNNV \hbox_set_end:
1336         \clist_set:Nn \l_draw_layers_clist \l_draw_layers_clist
1337         \__draw_layers_insert:
1338         \__draw_backend_end:
1339     \hbox_set_end:
1340     \dim_compare:nNnT \g__draw_xmin_dim = \c_max_dim
1341     {
1342         \dim_gzero:N \g__draw_xmax_dim
1343         \dim_gzero:N \g__draw_xmin_dim
1344         \dim_gzero:N \g__draw_ymax_dim
1345         \dim_gzero:N \g__draw_ymin_dim
1346     }
1347     \__draw_finalise:
1348     \box_set_wd:Nn \l__draw_main_box
1349     { \g__draw_xmax_dim - \g__draw_xmin_dim }
1350     \mode_leave_vertical:
1351     \box_use_drop:N \l__draw_main_box
1352 \group_end:
1353 }

```

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

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

```

1354 \cs_new_protected:Npn \__draw_finalise:
1355 {
1356     \hbox_set:Nn \l__draw_main_box
1357     {
1358         \skip_horizontal:n { -\g__draw_xmin_dim }
1359         \box_move_down:nn
1360         { \g__draw_ymin_dim }
1361         { \box_use_drop:N \l__draw_main_box }
1362     }
1363     \box_set_dp:Nn \l__draw_main_box { Opt }
1364     \box_set_ht:Nn \l__draw_main_box
1365     { \g__draw_ymax_dim - \g__draw_ymin_dim }
1366 }
1367 \cs_new_protected:Npn \__draw_finalise_baseline:n #1
1368 {
1369     \hbox_set:Nn \l__draw_main_box
1370     {
1371         \skip_horizontal:n { -\g__draw_xmin_dim }
1372         \box_move_down:nn
1373         { #1 }
1374         { \box_use_drop:N \l__draw_main_box }
1375     }
1376     \box_set_dp:Nn \l__draw_main_box
1377     {
1378         \dim_max:nn
1379         { #1 - \g__draw_ymin_dim }
1380         { Opt }

```

```

1381     }
1382     \box_set_ht:Nn \l__draw_main_box
1383     { \g__draw_ymax_dim - #1 }
1384 }

```

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

6.2 Baseline position

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

```

\l__draw_baseline_dim
1385 \bool_new:N \l__draw_baseline_bool
1386 \dim_new:N \l__draw_baseline_dim

```

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

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

```

1387 \cs_new_protected:Npn \draw_baseline:n #1
1388 {
1389     \bool_set_true:N \l__draw_baseline_bool
1390     \dim_set:Nn \l__draw_baseline_dim { \fp_to_dim:n {#1} }
1391 }

```

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

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

```

1392 \cs_new_protected:Npn \__draw_baseline_finalise:w #1 \__draw_finalise:
1393 {
1394     \bool_if:NTF \l__draw_baseline_bool
1395     {
1396         \use:e
1397         {
1398             \exp_not:n {#1}
1399             \__draw_finalise_baseline:n { \dim_use:N \l__draw_baseline_dim }
1400         }
1401     }
1402     { #1 \__draw_finalise: }
1403 }

```

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

6.3 Scopes

`\l__draw_linewidth_dim` Storage for local variables.

```

\l__draw_fill_color_tl
\l__draw_stroke_color_tl
1404 \dim_new:N \l__draw_linewidth_dim
1405 \tl_new:N \l__draw_fill_color_tl
1406 \tl_new:N \l__draw_stroke_color_tl

```

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

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

```

\draw_scope_begin: 1407 \cs_new_protected:Npn \draw_scope_begin:
1408 {
1409   \__draw_backend_scope_begin:
1410   \group_begin:
1411     \dim_set_eq:NN \l__draw_linewidth_dim \g__draw_linewidth_dim
1412     \draw_path_scope_begin:
1413   }
1414 \cs_new_protected:Npn \draw_scope_end:
1415 {
1416   \draw_path_scope_end:
1417   \dim_gset_eq:NN \g__draw_linewidth_dim \l__draw_linewidth_dim
1418   \group_end:
1419   \__draw_backend_scope_end:
1420 }

```

(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 1421 \dim_new:N \l__draw_xmax_dim
\l__draw_ymax_dim 1422 \dim_new:N \l__draw_xmin_dim
\l__draw_ymin_dim 1423 \dim_new:N \l__draw_ymax_dim
1424 \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: 1425 \cs_new_protected:Npn \__draw_scope_bb_begin:
1426 {
1427   \group_begin:
1428     \dim_set_eq:NN \l__draw_xmax_dim \g__draw_xmax_dim
1429     \dim_set_eq:NN \l__draw_xmin_dim \g__draw_xmin_dim
1430     \dim_set_eq:NN \l__draw_ymax_dim \g__draw_ymax_dim
1431     \dim_set_eq:NN \l__draw_ymin_dim \g__draw_ymin_dim
1432     \__draw_reset_bb:
1433   }
1434 \cs_new_protected:Npn \__draw_scope_bb_end:
1435 {
1436   \dim_gset_eq:NN \g__draw_xmax_dim \l__draw_xmax_dim
1437   \dim_gset_eq:NN \g__draw_xmin_dim \l__draw_xmin_dim
1438   \dim_gset_eq:NN \g__draw_ymax_dim \l__draw_ymax_dim
1439   \dim_gset_eq:NN \g__draw_ymin_dim \l__draw_ymin_dim
1440   \group_end:
1441 }

```

(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: 1442 \cs_new_protected:Npn \draw_suspend_begin:
1443 {
1444   \__draw_scope_bb_begin:
1445   \draw_path_scope_begin:
1446   \draw_transform_matrix_reset:
1447   \draw_transform_shift_reset:

```



```

1448     \__draw_layers_save:
1449   }
1450 \cs_new_protected:Npn \draw_suspend_end:
1451 {
1452   \__draw_layers_restore:
1453   \draw_path_scope_end:
1454   \__draw_scope_bb_end:
1455 }

```

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

```

1456 </package>

```

7 I3draw-softpath implementation

```

1457 <*package>

```

```

1458 <@@=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.

```

1459 \tl_new:N \g__draw_softpath_main_tl

```

(End of definition for `\g__draw_softpath_main_tl`.)

`\l__draw_softpath_tmp_tl` Scratch space.

```

1460 \tl_new:N \l__draw_softpath_tmp_tl

```

(End of definition for `\l__draw_softpath_tmp_tl`.)

`\g__draw_softpath_corners_bool` Allow for optimised path use.

```

1461 \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`

`__draw_softpath_add:e`

```

1462 \cs_new_protected:Npn \__draw_softpath_add:n

```

```

1463 { \tl_build_gput_right:Nn \g__draw_softpath_main_tl }

```

```

1464 \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: 1465 \cs_new_protected:Npn \__draw_softpath_use:
1466 {
1467   \tl_build_get_intermediate:NN
1468   \g__draw_softpath_main_tl
1469   \l__draw_softpath_tmp_tl
1470   \l__draw_softpath_tmp_tl
1471 }
1472 \cs_new_protected:Npn \__draw_softpath_clear:
1473 {
1474   \tl_build_gbegin:N \g__draw_softpath_main_tl
1475   \bool_gset_false:N \g__draw_softpath_corners_bool
1476 }

(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: 1477 \cs_new_protected:Npn \__draw_softpath_save:
1478 {
1479   \tl_build_gend:N \g__draw_softpath_main_tl
1480   \tl_set_eq:NN
1481   \l__draw_softpath_main_tl
1482   \g__draw_softpath_main_tl
1483   \bool_set_eq:NN
1484   \l__draw_softpath_corners_bool
1485   \g__draw_softpath_corners_bool
1486   \__draw_softpath_clear:
1487 }
1488 \cs_new_protected:Npn \__draw_softpath_restore:
1489 {
1490   \__draw_softpath_clear:
1491   \__draw_softpath_add:o \l__draw_softpath_main_tl
1492   \bool_gset_eq:NN
1493   \g__draw_softpath_corners_bool
1494   \l__draw_softpath_corners_bool
1495 }

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

```

```

\g__draw_softpath_lastx_dim For tracking the end of the path (to close it).
\g__draw_softpath_lasty_dim 1496 \dim_new:N \g__draw_softpath_lastx_dim
1497 \dim_new:N \g__draw_softpath_lasty_dim

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

```

```

\g__draw_softpath_move_bool Track if moving a point should update the close position.
1498 \bool_new:N \g__draw_softpath_move_bool
1499 \bool_gset_true:N \g__draw_softpath_move_bool

(End of definition for \g__draw_softpath_move_bool.)

```

```

\__draw_softpath_closepath: The various parts of a path expressed as the appropriate soft path functions.
  \__draw_softpath_curveto:nnnnnn 1500 \cs_new_protected:Npn \__draw_softpath_closepath:
\__draw_softpath_lineto:nn 1501 {
\__draw_softpath_moveto:nn 1502   \__draw_softpath_add:e
  \__draw_softpath_rectangle:nnnn 1503   {
  \__draw_softpath_roundpoint:nn 1504     \__draw_softpath_close_op:nn
  \__draw_softpath_roundpoint:VV 1505     { \dim_use:N \g__draw_softpath_lastx_dim }
  \__draw_softpath_roundpoint:VV 1506     { \dim_use:N \g__draw_softpath_lasty_dim }
  1507   }
  1508 }
1509 \cs_new_protected:Npn \__draw_softpath_curveto:nnnnnn #1#2#3#4#5#6
1510 {
1511   \__draw_softpath_add:n
1512   {
1513     \__draw_softpath_curveto_opi:nn {#1} {#2}
1514     \__draw_softpath_curveto_opii:nn {#3} {#4}
1515     \__draw_softpath_curveto_opiii:nn {#5} {#6}
1516   }
1517 }
1518 \cs_new_protected:Npn \__draw_softpath_lineto:nn #1#2
1519 {
1520   \__draw_softpath_add:n
1521   { \__draw_softpath_lineto_op:nn {#1} {#2} }
1522 }
1523 \cs_new_protected:Npn \__draw_softpath_moveto:nn #1#2
1524 {
1525   \__draw_softpath_add:n
1526   { \__draw_softpath_moveto_op:nn {#1} {#2} }
1527   \bool_if:NT \g__draw_softpath_move_bool
1528   {
1529     \dim_gset:Nn \g__draw_softpath_lastx_dim {#1}
1530     \dim_gset:Nn \g__draw_softpath_lasty_dim {#2}
1531   }
1532 }
1533 \cs_new_protected:Npn \__draw_softpath_rectangle:nnnn #1#2#3#4
1534 {
1535   \__draw_softpath_add:n
1536   {
1537     \__draw_softpath_rectangle_opi:nn {#1} {#2}
1538     \__draw_softpath_rectangle_opii:nn {#3} {#4}
1539   }
1540 }
1541 \cs_new_protected:Npn \__draw_softpath_roundpoint:nn #1#2
1542 {
1543   \__draw_softpath_add:n
1544   { \__draw_softpath_roundpoint_op:nn {#1} {#2} }
1545   \bool_gset_true:N \g__draw_softpath_corners_bool
1546 }
1547 \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
  \__draw_softpath_curveto_opi:nn
  \__draw_softpath_curveto_opii:nn
  \__draw_softpath_curveto_opiii:nn
  \__draw_softpath_lineto_op:nn
  \__draw_softpath_moveto_op:nn
  \__draw_softpath_roundpoint_op:nn
  \__draw_softpath_rectangle_opi:nn
  \__draw_softpath_rectangle_opii:nn
  \__draw_softpath_curveto_opi:nnNnnNnn
  \__draw_softpath_rectangle_opi:nnNnn

```

tokens for curves have to be different in meaning to a round point, hence being quark-like.

```

1548 \cs_new_protected:Npn \__draw_softpath_close_op:nn #1#2
1549 { \__draw_backend_closepath: }
1550 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nn #1#2
1551 { \__draw_softpath_curveto_opi:nnNnnNnn {#1} {#2} }
1552 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nnNnnNnn #1#2#3#4#5#6#7#8
1553 { \__draw_backend_curveto:nnnnnn {#1} {#2} {#4} {#5} {#7} {#8} }
1554 \cs_new_protected:Npn \__draw_softpath_curveto_opii:nn #1#2
1555 { \__draw_softpath_curveto_opii:nn }
1556 \cs_new_protected:Npn \__draw_softpath_curveto_opiii:nn #1#2
1557 { \__draw_softpath_curveto_opiii:nn }
1558 \cs_new_protected:Npn \__draw_softpath_lineto_op:nn #1#2
1559 { \__draw_backend_lineto:nn {#1} {#2} }
1560 \cs_new_protected:Npn \__draw_softpath_moveto_op:nn #1#2
1561 { \__draw_backend_moveto:nn {#1} {#2} }
1562 \cs_new_protected:Npn \__draw_softpath_roundpoint_op:nn #1#2
1563 { \__draw_softpath_roundpoint_op:nn }
1564 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nn #1#2
1565 { \__draw_softpath_rectangle_opi:nnNnn {#1} {#2} }
1566 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nnNnn #1#2#3#4#5
1567 { \__draw_backend_rectangle:nnnn {#1} {#2} {#4} {#5} }
1568 \cs_new_protected:Npn \__draw_softpath_rectangle_opii:nn #1#2
1569 { \__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.

```
1570 \tl_new:N \l__draw_softpath_main_tl
```

(End of definition for \l__draw_softpath_main_tl.)

`\l__draw_softpath_part_tl` Data structures.

```
1571 \tl_new:N \l__draw_softpath_part_tl
1572 \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 co-ordinate.

```

\l__draw_softpath_lasty_fp 1573 \fp_new:N \l__draw_softpath_lastx_fp
  \l__draw_softpath_corneri_dim 1574 \fp_new:N \l__draw_softpath_lasty_fp
  \l__draw_softpath_cornerii_dim 1575 \dim_new:N \l__draw_softpath_corneri_dim
\l__draw_softpath_first_tl 1576 \dim_new:N \l__draw_softpath_cornerii_dim
  \l__draw_softpath_move_tl 1577 \tl_new:N \l__draw_softpath_first_tl
  1578 \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.
 1579 `\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
`_draw_softpath_round_loop:Nnn` such, we avoid this entirely if we know there are no corners to deal with. Assuming there
`_draw_softpath_round_action:nn` is work to do, we recover the existing path and start a loop.
`_draw_softpath_round_action:Nnn` 1580 `\cs_new_protected:Npn _draw_softpath_round_corners:`
`_draw_softpath_round_action_curveto:NnnNnn` 1581 `{`
`_draw_softpath_round_action_close:` 1582 `\bool_if:NT \g__draw_softpath_corners_bool`
`_draw_softpath_round_lookahead:NnnNnn` 1583 `{`
`_draw_softpath_round_roundpoint:NnnNnnNnn` 1584 `\group_begin:`
`_draw_softpath_round_calc:NnnNnn` 1585 `\tl_clear:N \l__draw_softpath_main_tl`
`_draw_softpath_round_calc:nnnnnn` 1586 `\tl_clear:N \l__draw_softpath_part_tl`
`_draw_softpath_round_calc:eVnnnn` 1587 `\fp_zero:N \l__draw_softpath_lastx_fp`
`_draw_softpath_round_calc:nnnnw` 1588 `\fp_zero:N \l__draw_softpath_lasty_fp`
`_draw_softpath_round_close:nn` 1589 `\tl_clear:N \l__draw_softpath_first_tl`
`_draw_softpath_round_close:w` 1590 `\tl_clear:N \l__draw_softpath_move_tl`
`_draw_softpath_round_end:` 1591 `\tl_build_gend:N \g__draw_softpath_main_tl`
 1592 `\exp_after:wN _draw_softpath_round_loop:Nnn`
 1593 `\g__draw_softpath_main_tl`
 1594 `\q__draw_recursion_tail ? ?`
 1595 `\q__draw_recursion_stop`
 1596 `\group_end:`
 1597 `}`
 1598 `\bool_gset_false:N \g__draw_softpath_corners_bool`
 1599 `}`

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.

1600 `\cs_new_protected:Npn _draw_softpath_round_loop:Nnn #1#2#3`
 1601 `{`
 1602 `_draw_if_recursion_tail_stop_do:Nn #1 { _draw_softpath_round_end: }`
 1603 `\token_if_eq_meaning:NNTF #1 _draw_softpath_roundpoint_op:nn`
 1604 `{ _draw_softpath_round_action:nn {#2} {#3} }`
 1605 `{`
 1606 `\tl_if_empty:NT \l__draw_softpath_first_tl`
 1607 `{ \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }`
 1608 `\fp_set:Nn \l__draw_softpath_lastx_fp {#2}`
 1609 `\fp_set:Nn \l__draw_softpath_lasty_fp {#3}`
 1610 `\token_if_eq_meaning:NNTF #1 _draw_softpath_moveto_op:nn`
 1611 `{`
 1612 `\tl_put_right:No \l__draw_softpath_main_tl`
 1613 `\l__draw_softpath_move_tl`
 1614 `\tl_put_right:No \l__draw_softpath_main_tl`
 1615 `\l__draw_softpath_part_tl`
 1616 `\tl_set:Nn \l__draw_softpath_move_tl { #1 {#2} {#3} }`
 1617 `\tl_clear:N \l__draw_softpath_first_tl`
 1618 `\tl_clear:N \l__draw_softpath_part_tl`
 1619 `}`
`}`

```

1620         { \tl_put_right:Nn \l__draw_softpath_part_tl { #1 {#2} {#3} } }
1621         \__draw_softpath_round_loop:Nnn
1622     }
1623 }
1624 \cs_new_protected:Npn \__draw_softpath_round_action:nn #1#2
1625 {
1626     \dim_set:Nn \l__draw_softpath_corneri_dim {#1}
1627     \dim_set:Nn \l__draw_softpath_cornerii_dim {#2}
1628     \bool_lazy_and:nnTF
1629     { \dim_compare_p:nNn \l__draw_softpath_corneri_dim = { Opt } }
1630     { \dim_compare_p:nNn \l__draw_softpath_cornerii_dim = { Opt } }
1631     { \__draw_softpath_round_loop:Nnn }
1632     { \__draw_softpath_round_action:Nnn }
1633 }

```

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.

```

1634 \cs_new_protected:Npn \__draw_softpath_round_action:Nnn #1#2#3
1635 {
1636     \tl_if_empty:NT \l__draw_softpath_first_tl
1637     { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1638     \token_if_eq_meaning:NNTF #1 \__draw_softpath_curveto_opi:nn
1639     { \__draw_softpath_round_action_curveto:NnnNnn }
1640     {
1641         \token_if_eq_meaning:NNTF #1 \__draw_softpath_close_op:nn
1642         { \__draw_softpath_round_action_close: }
1643         {
1644             \token_if_eq_meaning:NNTF #1 \__draw_softpath_lineto_op:nn
1645             { \__draw_softpath_round_lookahead:NnnNnn }
1646             { \__draw_softpath_round_loop:Nnn }
1647         }
1648     }
1649     #1 {#2} {#3}
1650 }

```

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.

```

1651 \cs_new_protected:Npn \__draw_softpath_round_action_curveto:NnnNnn
1652 #1#2#3#4#5#6
1653 {
1654     \tl_put_right:Nn \l__draw_softpath_part_tl
1655     { #1 {#2} {#3} #4 {#5} {#6} }
1656     \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1657     \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1658     \__draw_softpath_round_lookahead:NnnNnn
1659 }
1660 \cs_new_protected:Npn \__draw_softpath_round_action_close:
1661 {
1662     \bool_lazy_and:nnTF
1663     { ! \tl_if_empty_p:N \l__draw_softpath_first_tl }
1664     { ! \tl_if_empty_p:N \l__draw_softpath_move_tl }
1665     {
1666         \exp_after:wN \__draw_softpath_round_close:nn
1667         \l__draw_softpath_first_tl

```

```

1668     }
1669     { \_draw_softpath_round_loop:Nnn }
1670 }

```

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.

```

1671 \cs_new_protected:Npn \_draw_softpath_round_lookahead:NnnNnn #1#2#3#4#5#6
1672 {
1673   \bool_lazy_any:nTF
1674   {
1675     { \token_if_eq_meaning_p:NN #4 \_draw_softpath_lineto_op:nn }
1676     { \token_if_eq_meaning_p:NN #4 \_draw_softpath_curveto_opi:nn }
1677     { \token_if_eq_meaning_p:NN #4 \_draw_softpath_close_op:nn }
1678   }
1679   {
1680     \_draw_softpath_round_calc:NnnNnn
1681     \_draw_softpath_round_loop:Nnn
1682     {#5} {#6}
1683   }
1684   {
1685     \token_if_eq_meaning:NNTF #4 \_draw_softpath_roundpoint_op:nn
1686     { \_draw_softpath_round_roundpoint:NnnNnnNnn }
1687     { \_draw_softpath_round_loop:Nnn }
1688   }
1689   #1 {#2} {#3}
1690   #4 {#5} {#6}
1691 }
1692 \cs_new_protected:Npn \_draw_softpath_round_roundpoint:NnnNnnNnn
1693 #1#2#3#4#5#6#7#8#9
1694 {
1695   \_draw_softpath_round_calc:NnnNnn
1696   \_draw_softpath_round_loop:Nnn
1697   {#8} {#9}
1698   #1 {#2} {#3}
1699   #4 {#5} {#6} #7 {#8} {#9}
1700 }

```

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.

```

1701 \cs_new_protected:Npn \_draw_softpath_round_calc:NnnNnn #1#2#3#4#5#6
1702 {
1703   \tl_set:Ne \l__draw_softpath_curve_end_tl
1704   {
1705     \draw_point_interpolate_distance:nnn
1706     \l__draw_softpath_cornerii_dim
1707     { #5 , #6 } { #2 , #3 }
1708   }

```

```

1709 \tl_put_right:Ne \l__draw_softpath_part_tl
1710 {
1711   \exp_not:N #4
1712   \__draw_softpath_round_calc:eVnnnn
1713   {
1714     \draw_point_interpolate_distance:nnn
1715     \l__draw_softpath_corneri_dim
1716     { #5 , #6 }
1717     {
1718       \l__draw_softpath_lastx_fp ,
1719       \l__draw_softpath_lasty_fp
1720     }
1721   }
1722   \l__draw_softpath_curve_end_tl
1723   {#5} {#6} {#2} {#3}
1724 }
1725 \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1726 \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1727 #1
1728 }

```

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

```

1729 \cs_new:Npn \__draw_softpath_round_calc:nnnnnn #1#2#3#4#5#6
1730 {
1731   \__draw_softpath_round_calc:nnnnw {#3} {#4} {#5} {#6}
1732   #1 \s__draw_mark #2 \s__draw_stop
1733 }
1734 \cs_generate_variant:Nn \__draw_softpath_round_calc:nnnnnn { eV }

```

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

```

1735 \cs_new:Npn \__draw_softpath_round_calc:nnnnw
1736 #1#2#3#4 #5 , #6 \s__draw_mark #7 , #8 \s__draw_stop
1737 {
1738   {#5} {#6}
1739   \exp_not:N \__draw_softpath_curveto_opi:nn
1740   {
1741     \fp_to_dim:n
1742     { #5 + \c__draw_softpath_arc_fp * ( #1 - #5 ) }
1743   }
1744   {
1745     \fp_to_dim:n
1746     { #6 + \c__draw_softpath_arc_fp * ( #2 - #6 ) }
1747   }
1748   \exp_not:N \__draw_softpath_curveto_opii:nn
1749   {
1750     \fp_to_dim:n
1751     { #7 + \c__draw_softpath_arc_fp * ( #1 - #7 ) }
1752   }
1753   {
1754     \fp_to_dim:n
1755     { #8 + \c__draw_softpath_arc_fp * ( #2 - #8 ) }
1756   }
1757   \exp_not:N \__draw_softpath_curveto_opiii:nn

```



```

1758     {#7} {#8}
1759   }

```

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.

```

1760 \cs_new_protected:Npn \__draw_softpath_round_close:nn #1#2
1761   {
1762     \use:e
1763     {
1764       \__draw_softpath_round_calc:NnnNnn
1765       {
1766         \tl_set:Nc \exp_not:N \l__draw_softpath_move_tl
1767           {
1768             \__draw_softpath_moveto_op:nn
1769             \exp_not:N \exp_after:wN
1770             \exp_not:N \__draw_softpath_round_close:w
1771             \exp_not:N \l__draw_softpath_curve_end_tl
1772             \s__draw_stop
1773           }
1774         \use:e
1775         {
1776           \exp_not:N \exp_not:N \exp_not:N \use_i:nnnn
1777           {
1778             \__draw_softpath_round_loop:Nnn
1779             \__draw_softpath_close_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         }
1786       }
1787     }
1788     {#1} {#2}
1789     \__draw_softpath_lineto_op:nn
1790     \exp_after:wN \use_none:n \l__draw_softpath_move_tl
1791   }
1792 \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.

```

1793 \cs_new_protected:Npn \__draw_softpath_round_end:
1794   {
1795     \tl_put_right:No \l__draw_softpath_main_tl
1796     \l__draw_softpath_move_tl
1797     \tl_put_right:No \l__draw_softpath_main_tl
1798     \l__draw_softpath_part_tl
1799     \tl_build_gbegin:N \g__draw_softpath_main_tl
1800     \__draw_softpath_add:o \l__draw_softpath_main_tl
1801   }

```

(End of definition for __draw_softpath_round_corners: and others.)

```

1802 \</package>

```

8 l3draw-state implementation

1803 `<*package>`

1804 `<@@=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.

1805 `\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.

1806 `\dim_new:N \l_draw_default_linewidth_dim`

1807 `\dim_set:Nn \l_draw_default_linewidth_dim { 0.4pt }`

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

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

1808 `\cs_new_protected:Npn \draw_linewidth:n #1`

1809 `{`

1810 `\dim_gset:Nn \g__draw_linewidth_dim { \fp_to_dim:n {#1} }`

1811 `__draw_backend_linewidth:n \g__draw_linewidth_dim`

1812 `}`

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

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

`\l__draw_tmp_seq` 1813 `\cs_new_protected:Npn \draw_dash_pattern:nn #1#2`

1814 `{`

1815 `\group_begin:`

1816 `\seq_set_from_clist:Nn \l__draw_tmp_seq {#1}`

1817 `\seq_set_map:NNn \l__draw_tmp_seq \l__draw_tmp_seq`

1818 `{ \fp_to_dim:n {##1} }`

1819 `\use:e`

1820 `{`

1821 `__draw_backend_dash_pattern:nn`

1822 `{ \seq_use:Nn \l__draw_tmp_seq { , } }`

1823 `{ \fp_to_dim:n {#2} }`

1824 `}`

1825 `\group_end:`

1826 `}`

1827 `\seq_new:N \l__draw_tmp_seq`

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

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

1828 `\cs_new_protected:Npn \draw_miterlimit:n #1`

1829 `{ \exp_args:Ne __draw_backend_miterlimit:n { \fp_eval:n {#1} } }`

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

```
\draw_cap_but: All straight wrappers.
\draw_cap_rectangle: 1830 \cs_new_protected:Npn \draw_cap_but: { \__draw_backend_cap_but: }
\draw_cap_round: 1831 \cs_new_protected:Npn \draw_cap_rectangle: { \__draw_backend_cap_rectangle: }
\draw_evenodd_rule: 1832 \cs_new_protected:Npn \draw_cap_round: { \__draw_backend_cap_round: }
\draw_nonzero_rule: 1833 \cs_new_protected:Npn \draw_evenodd_rule: { \__draw_backend_evenodd_rule: }
\draw_join_bevel: 1834 \cs_new_protected:Npn \draw_nonzero_rule: { \__draw_backend_nonzero_rule: }
\draw_join_miter: 1835 \cs_new_protected:Npn \draw_join_bevel: { \__draw_backend_join_bevel: }
\draw_join_round: 1836 \cs_new_protected:Npn \draw_join_miter: { \__draw_backend_join_miter: }
1837 \cs_new_protected:Npn \draw_join_round: { \__draw_backend_join_round: }
```

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

```
1838 </package>
```

9 I3draw-transforms implementation

```
1839 <*package>
```

```
1840 <@@=draw>
```

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

- `\pgfgettransform`, `\pgfgettransformentries`: Awaiting use cases.
- `\pgftransformlineattime`, `\pgftransformarcaxesattime`, `\pgftransformcurveattime`: Need to look at the use cases for these to fully understand them.
- `\pgftransformarrow`: Likely to be done when other arrow functions are added.
- `\pgftransformationadjustments`: Used mainly by `CircuiTikZ` although also for shapes, likely needs more use cases before addressing.
- `\pgfflowlevelsynccm`, `\pgfflowlevel`: Likely to be added when use cases are encountered in other parts of the code.
- `\pgfviewboxscope`: Seems very speicalied, need to understand the requirements here.

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

```
1841 \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 1842 \fp_new:N \l__draw_matrix_a_fp
\l__draw_matrix_c_fp 1843 \fp_new:N \l__draw_matrix_b_fp
\l__draw_xshift_dim 1844 \fp_new:N \l__draw_matrix_c_fp
\l__draw_yshift_dim 1845 \fp_new:N \l__draw_matrix_d_fp
1846 \dim_new:N \l__draw_xshift_dim
1847 \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:
1848 \cs_new_protected:Npn \draw_transform_matrix_reset:
1849 {
1850   \fp_set:Nn \l__draw_matrix_a_fp { 1 }
1851   \fp_zero:N \l__draw_matrix_b_fp
1852   \fp_zero:N \l__draw_matrix_c_fp
1853   \fp_set:Nn \l__draw_matrix_d_fp { 1 }
1854   \bool_set_false:N \l__draw_matrix_active_bool
1855 }
1856 \cs_new_protected:Npn \draw_transform_shift_reset:
1857 {
1858   \dim_zero:N \l__draw_xshift_dim
1859   \dim_zero:N \l__draw_yshift_dim
1860 }
1861 \draw_transform_matrix_reset:
1862 \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
1863 \cs_new_protected:Npn \draw_transform_matrix_absolute:nmmn #1#2#3#4
1864 {
1865   \fp_set:Nn \l__draw_matrix_a_fp {#1}
1866   \fp_set:Nn \l__draw_matrix_b_fp {#2}
1867   \fp_set:Nn \l__draw_matrix_c_fp {#3}
1868   \fp_set:Nn \l__draw_matrix_d_fp {#4}
1869   \bool_lazy_all:nTF
1870     {
1871       { \fp_compare_p:nNn \l__draw_matrix_a_fp = \c_one_fp }
1872       { \fp_compare_p:nNn \l__draw_matrix_b_fp = \c_zero_fp }
1873       { \fp_compare_p:nNn \l__draw_matrix_c_fp = \c_zero_fp }
1874       { \fp_compare_p:nNn \l__draw_matrix_d_fp = \c_one_fp }
1875     }
1876     { \bool_set_false:N \l__draw_matrix_active_bool }
1877     { \bool_set_true:N \l__draw_matrix_active_bool }
1878 }
1879 \cs_new_protected:Npn \draw_transform_shift_absolute:n #1
1880 {
1881   \__draw_point_process:nn
1882   { \__draw_transform_shift_absolute:nn } {#1}
1883 }
1884 \cs_new_protected:Npn \__draw_transform_shift_absolute:nn #1#2
1885 { \__draw_transform_shift:nnnn { Opt } { Opt } {#1} {#2} }

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

\draw_transform_matrix:nnnn Much the same story for adding to an existing matrix, with a bit of pre-expansion so
\__draw_transform:nnnn that the calculation uses “frozen” values.
\draw_transform_shift:n 1886 \cs_new_protected:Npn \draw_transform_matrix:nnnn #1#2#3#4
\_draw_transform_shift:nn 1887 {
1888   \use:e

```

```

1889     {
1890         \__draw_transform:nnnn
1891         { \fp_eval:n {#1} }
1892         { \fp_eval:n {#2} }
1893         { \fp_eval:n {#3} }
1894         { \fp_eval:n {#4} }
1895     }
1896 }
1897 \cs_new_protected:Npn \__draw_transform:nnnn #1#2#3#4
1898 {
1899     \use:e
1900     {
1901         \draw_transform_matrix_absolute:nnnn
1902         { #1 * \l__draw_matrix_a_fp + #2 * \l__draw_matrix_c_fp }
1903         { #1 * \l__draw_matrix_b_fp + #2 * \l__draw_matrix_d_fp }
1904         { #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_c_fp }
1905         { #3 * \l__draw_matrix_b_fp + #4 * \l__draw_matrix_d_fp }
1906     }
1907 }
1908 \cs_new_protected:Npn \draw_transform_shift:n #1
1909 {
1910     \__draw_point_process:nn
1911     { \__draw_transform_shift:nn } {#1}
1912 }
1913 \cs_new_protected:Npn \__draw_transform_shift:nn #1#2
1914 {
1915     \__draw_transform_shift:nnnn
1916     \l__draw_xshift_dim
1917     \l__draw_yshift_dim
1918     {#1} {#2}
1919 }

```

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

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

```

1920 \cs_new_protected:Npn \__draw_transform_shift:nnnn #1#2#3#4
1921 {
1922     \dim_set:Nn \l__draw_xshift_dim
1923     {
1924         \fp_to_dim:n
1925         {
1926             #1 +
1927             ( #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_c_fp )
1928         }
1929     }
1930     \dim_set:Nn \l__draw_yshift_dim
1931     {
1932         \fp_to_dim:n
1933         {
1934             #2 +
1935             ( #3 * \l__draw_matrix_b_fp + #4 * \l__draw_matrix_d_fp )
1936         }

```

```

1937     }
1938 }

```

(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 1939 \cs_new_protected:Npn \draw_transform_matrix_invert:
\__draw_transform_invert:e 1940 {
\draw_transform_shift_invert: 1941   \bool_if:NT \l__draw_matrix_active_bool
1942   {
1943     \__draw_transform_invert:e
1944     {
1945       \fp_eval:n
1946       {
1947         1 /
1948         (
1949           \l__draw_matrix_a_fp * \l__draw_matrix_d_fp
1950           - \l__draw_matrix_b_fp * \l__draw_matrix_c_fp
1951         )
1952       }
1953     }
1954   }
1955 }
1956 \cs_new_protected:Npn \__draw_transform_invert:n #1
1957 {
1958   \fp_set:Nn \l__draw_matrix_a_fp
1959   { \l__draw_matrix_d_fp * #1 }
1960   \fp_set:Nn \l__draw_matrix_b_fp
1961   { -\l__draw_matrix_b_fp * #1 }
1962   \fp_set:Nn \l__draw_matrix_c_fp
1963   { -\l__draw_matrix_c_fp * #1 }
1964   \fp_set:Nn \l__draw_matrix_d_fp
1965   { \l__draw_matrix_a_fp * #1 }
1966 }
1967 \cs_generate_variant:Nn \__draw_transform_invert:n { e }
1968 \cs_new_protected:Npn \draw_transform_shift_invert:
1969 {
1970   \dim_set:Nn \l__draw_xshift_dim { -\l__draw_xshift_dim }
1971   \dim_set:Nn \l__draw_yshift_dim { -\l__draw_yshift_dim }
1972 }

```

(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.

```

1973 \cs_new_protected:Npn \draw_transform_triangle:nnn #1#2#3
1974 {
1975   \__draw_point_process:nnn
1976   {
1977     \__draw_point_process:nn
1978     { \__draw_transform_triangle:nnnnnn }
1979     {#1}
1980   }
1981   {#2} {#3}
1982 }

```

```

1983 \cs_new_protected:Npn \__draw_transform_triangle:nnnnnn #1#2#3#4#5#6
1984 {
1985   \use:e
1986   {
1987     \draw_transform_matrix_absolute:nnnn
1988     { #3 - #1 }
1989     { #4 - #2 }
1990     { #5 - #1 }
1991     { #6 - #2 }
1992     \draw_transform_shift_absolute:n { #1 , #2 }
1993   }
1994 }

```

(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 1995 \cs_new_protected:Npn \draw_transform_scale:n #1
\draw_transform_yscale:n 1996 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { #1 } }
\draw_transform_xshift:n 1997 \cs_new_protected:Npn \draw_transform_xscale:n #1
\draw_transform_yshift:n 1998 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { 1 } }
\draw_transform_xslant:n 1999 \cs_new_protected:Npn \draw_transform_yscale:n #1
\draw_transform_yslant:n 2000 { \draw_transform_matrix:nnnn { 1 } { 0 } { 0 } { #1 } }
2001 \cs_new_protected:Npn \draw_transform_xshift:n #1
2002 { \draw_transform_shift:n { #1 , Opt } }
2003 \cs_new_protected:Npn \draw_transform_yshift:n #1
2004 { \draw_transform_shift:n { Opt , #1 } }
2005 \cs_new_protected:Npn \draw_transform_xslant:n #1
2006 { \draw_transform_matrix:nnnn { 1 } { 0 } { #1 } { 1 } }
2007 \cs_new_protected:Npn \draw_transform_yslant:n #1
2008 { \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 2009 \cs_new_protected:Npn \draw_transform_rotate:n #1
\__draw_transform_rotate:e 2010 { \__draw_transform_rotate:e { \fp_eval:n {#1} } }
\__draw_transform_rotate:nn 2011 \cs_new_protected:Npn \__draw_transform_rotate:n #1
\__draw_transform_rotate:ee 2012 {
2013   \__draw_transform_rotate:ee
2014   { \fp_eval:n { cosd(#1) } }
2015   { \fp_eval:n { sind(#1) } }
2016 }
2017 \cs_generate_variant:Nn \__draw_transform_rotate:n { e }
2018 \cs_new_protected:Npn \__draw_transform_rotate:nn #1#2
2019 { \draw_transform_matrix:nnnn {#1} {#2} { -#2 } { #1 } }
2020 \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 ??.)

```

2021 </package>

```

Index

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

B	
<code>\begin</code> . . .	193, 823, 1121, 1125, 1149, 1152
bool commands:	
<code>\bool_gset_eq:NN</code>	1492
<code>\bool_gset_false:N</code>	1475, 1598
<code>\bool_gset_true:N</code>	1499, 1545
<code>\bool_if:NTF</code>	
.	33, 137, 216, 255, 724, 740, 741,
.	745, 1244, 1276, 1394, 1527, 1582, 1941
<code>\bool_lazy_all:nTF</code>	1869
<code>\bool_lazy_and:nnTF</code>	
.	247, 719, 1628, 1662
<code>\bool_lazy_any:nTF</code>	1673
<code>\bool_lazy_or:nnTF</code>	580, 678, 728, 733
<code>\bool_new:N</code>	108, 242, 667, 668, 669,
.	670, 790, 1300, 1385, 1461, 1498, 1841
<code>\bool_set_eq:NN</code>	1483
<code>\bool_set_false:N</code>	118,
.	250, 706, 707, 708, 727, 1854, 1876
<code>\bool_set_true:N</code>	120, 251,
.	712, 753, 757, 758, 1319, 1389, 1877
box commands:	
<code>\box_dp:N</code>	17, 22, 89
<code>\box_gset_eq:NN</code>	178
<code>\box_gset_wd:Nn</code>	122, 155
<code>\box_ht:N</code>	17, 22, 91
<code>\box_if_exist:NTF</code>	115
<code>\box_move_down:nn</code>	1359, 1372
<code>\box_move_up:nn</code>	63
<code>\box_new:N</code>	13, 102, 103, 1301, 1302
<code>\box_set_dp:Nn</code>	67, 1363, 1376
<code>\box_set_eq:NN</code>	167
<code>\box_set_ht:Nn</code>	66, 1364, 1382
<code>\box_set_wd:Nn</code>	68, 150, 1348
<code>\box_use_drop:N</code>	64,
.	69, 124, 151, 156, 1351, 1361, 1374
<code>\box_wd:N</code>	17, 22, 88, 90
C	
clist commands:	
<code>\clist_map_inline:Nn</code>	146, 163, 174
<code>\clist_map_inline:nn</code>	709
<code>\clist_new:N</code>	109, 111
<code>\clist_set:Nn</code>	110, 1336
coffin commands:	
<code>\coffin_typeset:Nnnnn</code>	86
<code>\coffin_wd:N</code>	88
color commands:	
<code>\color_select:n</code>	1324
cs commands:	
<code>\cs_generate_variant:Nn</code>	442, 633,
.	666, 835, 847, 860, 878, 886, 894,
.	912, 937, 956, 963, 971, 978, 987,
.	993, 1013, 1021, 1028, 1034, 1046,
.	1049, 1067, 1085, 1093, 1099, 1120,
.	1134, 1148, 1169, 1204, 1223, 1236,
.	1464, 1547, 1734, 1967, 2017, 2020
<code>\cs_if_exist:NTF</code>	711
<code>\cs_if_exist_use:NTF</code>	425, 434, 714
<code>\cs_new:Npn</code>	
.	535, 545, 555, 565, 827, 833,
.	836, 838, 845, 848, 850, 858, 861,
.	864, 873, 879, 882, 884, 887, 888,
.	890, 892, 895, 897, 903, 913, 919,
.	929, 938, 944, 949, 957, 964, 972,
.	979, 988, 994, 1000, 1005, 1014,
.	1022, 1029, 1035, 1041, 1047, 1050,
.	1056, 1065, 1068, 1074, 1079, 1086,
.	1094, 1100, 1106, 1112, 1126, 1135,
.	1141, 1153, 1155, 1164, 1194, 1196,
.	1205, 1210, 1224, 1226, 1228, 1237,
.	1242, 1269, 1274, 1729, 1735, 1792
<code>\cs_new_protected:Npn</code>	
.	14, 19, 24, 31, 72, 77,
.	82, 97, 112, 135, 144, 161, 172, 206,
.	228, 235, 243, 253, 262, 268, 274,
.	280, 287, 298, 306, 311, 313, 315,
.	324, 331, 367, 369, 380, 386, 416,
.	443, 472, 478, 484, 489, 497, 506,
.	513, 521, 576, 578, 593, 600, 609,
.	615, 617, 627, 634, 640, 647, 671,
.	676, 687, 696, 704, 751, 755, 760,
.	767, 791, 805, 1176, 1178, 1180,
.	1182, 1186, 1304, 1311, 1332, 1354,
.	1367, 1387, 1392, 1407, 1414, 1425,
.	1434, 1442, 1450, 1462, 1465, 1472,
.	1477, 1488, 1500, 1509, 1518, 1523,
.	1533, 1541, 1548, 1550, 1552, 1554,
.	1556, 1558, 1560, 1562, 1564, 1566,
.	1568, 1580, 1600, 1624, 1634, 1651,
.	1660, 1671, 1692, 1701, 1760, 1793,
.	1808, 1813, 1828, 1830, 1831, 1832,
.	1833, 1834, 1835, 1836, 1837, 1848,
.	1856, 1863, 1879, 1884, 1886, 1897,
.	1908, 1913, 1920, 1939, 1956, 1968,

1973, 1983, 1995, 1997, 1999, 2001,
2003, 2005, 2007, 2009, 2011, 2018

D

dim commands:

\dim_abs:n 622, 623
 \dim_compare:nNnTF 629, 636, 769, 1340
 \dim_compare_p:nNn 248, 249, 1629, 1630
 \dim_gset:Nn 208, 210, 212,
 214, 218, 220, 222, 224, 230, 231,
 232, 233, 237, 238, 698, 771, 1306,
 1307, 1308, 1309, 1529, 1530, 1810
 \dim_gset_eq:NN
 808, 809, 810, 811, 812, 813,
 814, 815, 1417, 1436, 1437, 1438, 1439
 \dim_gzero:N .. 1342, 1343, 1344, 1345
 \dim_max:nn .. 209, 213, 219, 223, 1378
 \dim_min:nn 211, 215, 221, 225
 \dim_new:N 200, 201,
 202, 203, 204, 205, 240, 241, 782,
 783, 784, 785, 786, 787, 788, 789,
 1170, 1171, 1172, 1173, 1174, 1175,
 1296, 1297, 1298, 1299, 1386, 1404,
 1421, 1422, 1423, 1424, 1496, 1497,
 1575, 1576, 1805, 1806, 1846, 1847
 \dim_set:Nn
 . 245, 246, 1188, 1189, 1390, 1626,
 1627, 1807, 1922, 1930, 1970, 1971
 \dim_set_eq:NN
 794, 795, 796, 797, 798, 799,
 800, 801, 1411, 1428, 1429, 1430, 1431
 \dim_step_inline:nnnn 649, 657
 \dim_use:N
 . 700, 769, 774, 776, 1399, 1505, 1506
 \dim_zero:N 1858, 1859
 \c_max_dim 230, 231, 232,
 233, 769, 1306, 1307, 1308, 1309, 1340

draw commands:

\draw_baseline:n 1387, 1387
 \l_draw_bb_update_bool
 33, 216, 720, 727, 1300, 1319
 \draw_begin: 1311, 1311
 \draw_box_use:N 14, 14
 \draw_box_use:Nn 14, 19
 \draw_cap_but: 1326, 1830, 1830
 \draw_cap_rectangle: 1830, 1831
 \draw_cap_round: 1830, 1832
 \draw_coffin_use:Nnn 36, 72, 72
 \draw_coffin_use:Nnnn 72, 77
 \draw_dash_pattern:nn 1329, 1813, 1813
 \l_draw_default_linewidth_dim ...
 127, 1323, 1806
 \draw_end: 1311, 1332
 \draw_evenodd_rule: 1830, 1833

\draw_join_bevel: 1830, 1835
 \draw_join_miter: ... 1327, 1830, 1836
 \draw_join_round: 1830, 1837
 \draw_layer_begin:n 112, 112
 \draw_layer_end: 112, 135
 \draw_layer_new:n 97, 97
 \l_draw_layers_clist
 109, 146, 163, 174, 1336
 \draw_linewidth:n 127, 1323, 1808, 1808
 \draw_miterlimit:n .. 1328, 1828, 1828
 \draw_nonzero_rule: . 1325, 1830, 1834
 \draw_path_arc:nnn 367, 367, 510
 \draw_path_arc:nnnn ... 367, 368, 369
 \draw_path_arc_axes:nnnn ... 506, 506
 \draw_path_canvas_curveto:nnn ...
 311, 315
 \draw_path_canvas_lineto:n . 311, 313
 \draw_path_canvas_moveto:n . 311, 311
 \draw_path_circle:nn 576, 576
 \draw_path_close: 306, 306, 606
 \draw_path_corner_arc:nn ... 243, 243
 \draw_path_curveto:nn 324, 324
 \draw_path_curveto:nnn 262, 287
 \draw_path_ellipse:nnn . 513, 513, 577
 \draw_path_grid:nnnn 617, 617
 \draw_path_lineto:n
 262, 274, 603, 604, 605, 655, 663
 \draw_path_moveto:n
 262, 262, 602, 607, 654, 662
 \draw_path_rectangle:nn 578, 578, 616
 \draw_path_rectangle_corners:nn .
 609, 609
 \draw_path_replace_bb: 671, 687
 \draw_path_scope_begin:
 791, 791, 1412, 1445
 \draw_path_scope_end:
 791, 805, 1416, 1453
 \draw_path_use:n 671, 671
 \draw_path_use_clear:n 671, 676
 \draw_point:n 830, 841,
 842, 853, 854, 855, 867, 868, 869,
 870, 882, 882, 893, 908, 931, 990,
 1031, 1048, 1066, 1096, 1166, 1198,
 1212, 1230, 1246, 1262, 1278, 1291
 \draw_point_interpolate_arcaxes:nnnnnn
 1068, 1068
 \draw_point_interpolate_curve:nnnnn
 1100
 \draw_point_interpolate_curve:nnnnnn
 1100
 \draw_point_interpolate_curve_
 auxi:nnnnnnnn 1100
 \draw_point_interpolate_curve_
 auxii:nnnnnnnn 1100

<code>\draw_point_interpolate_curve_</code>		<code>\draw_transform_shift_invert:</code>	...
<code>auxiii:nnnnnn</code>	1100	1939, 1968
<code>\draw_point_interpolate_curve_</code>		<code>\draw_transform_shift_reset:</code>	...
<code>auxiv:nnnnnn</code>	1100	1321, 1447, 1848, 1856, 1862
<code>\draw_point_interpolate_curve_</code>		<code>\draw_transform_triangle:nnn</code>	...
<code>auxv:nnw</code>	1100	509, 1973, 1973
<code>\draw_point_interpolate_curve_</code>		<code>\draw_transform_xscale:n</code>	1995, 1997
<code>auxvi:n</code>	1100	<code>\draw_transform_xshift:n</code>	1995, 2001
<code>\draw_point_interpolate_curve_</code>		<code>\draw_transform_xslant:n</code>	1995, 2005
<code>auxvii:nnnnnnnn</code>	1100	<code>\draw_transform_yscale:n</code>	1995, 1999
<code>\draw_point_interpolate_curve_</code>		<code>\draw_transform_yshift:n</code>	1995, 2003
<code>auxviii:nnnnnnn</code>	1100	<code>\draw_transform_yslant:n</code>	1995, 2007
<code>\draw_point_interpolate_distance:nnn</code>	1050, 1050, 1705, 1714	<code>\draw_xvec:n</code>	1176, 1176, 1191
<code>\draw_point_interpolate_line:nnn</code>	1035, 1035	<code>\draw_yvec:n</code>	1176, 1178, 1192
<code>\draw_point_intersect_circles:nnnnn</code>	938, 938	<code>\draw_zvec:n</code>	1176, 1180, 1193
<code>\draw_point_intersect_line_</code>		draw internal commands:	
<code>circle:nnnnn</code>	994, 994	<code>_draw_backend_begin:</code>	1316
<code>\draw_point_intersect_lines:nnnn</code>	913, 913	<code>_draw_backend_box_use:Nnnnn</code>	53
<code>\draw_point_polar:nn</code>	888, 888	<code>_draw_backend_cap_but:</code>	1830
<code>\draw_point_polar:nnn</code>	450, 456, 460, 466, 888, 889, 890	<code>_draw_backend_cap_rectangle:</code>	1831
<code>\draw_point_transform:n</code>	37, 40, 43, 46, 266, 278, 294, 295, 296, 328, 329, 455, 459, 517, 1237, 1237	<code>_draw_backend_cap_round:</code>	1832
<code>\draw_point_unit_vector:n</code>	895, 895, 1063	<code>_draw_backend_clip:</code>	726
<code>\draw_point_vec:nn</code>	1194, 1194	<code>_draw_backend_closepath:</code>	1549
<code>\draw_point_vec:nnn</code>	1194, 1205	<code>_draw_backend_curveto:nnnnnn</code>	1553
<code>\draw_point_vec_polar:nn</code>	1224, 1224	<code>_draw_backend_dash_pattern:nn</code>	1821
<code>\draw_point_vec_polar:nnn</code>	1224, 1225, 1226	<code>_draw_backend_discardpath:</code>	731
<code>\draw_scope_begin:</code>	26, 1407, 1407	<code>_draw_backend_end:</code>	1338
<code>\draw_scope_end:</code>	29, 1414	<code>_draw_backend_evenodd_rule:</code>	1833
<code>\draw_suspend_begin:</code>	1442, 1442	<code>_draw_backend_join_bevel:</code>	1835
<code>\draw_suspend_end:</code>	1442, 1450	<code>_draw_backend_join_miter:</code>	1836
<code>\draw_transform_matrix:nnnn</code>	1886, 1886, 1996, 1998, 2000, 2006, 2008, 2019	<code>_draw_backend_join_round:</code>	1837
<code>\draw_transform_matrix_absolute:nnnn</code>	1863, 1863, 1901, 1987	<code>_draw_backend_lineto:nn</code>	1559
<code>\draw_transform_matrix_invert:</code>	1939, 1939	<code>_draw_backend_linewidth:n</code>	1811
<code>\draw_transform_matrix_reset:</code>	1320, 1446, 1848, 1848, 1861	<code>_draw_backend_miterlimit:n</code>	1829
<code>\draw_transform_rotate:n</code>	2009, 2009	<code>_draw_backend_moveto:nn</code>	1561
<code>\draw_transform_scale:n</code>	1995, 1995	<code>_draw_backend_nonzero_rule:</code>	1834
<code>\draw_transform_shift:n</code>	27, 1886, 1908, 2002, 2004	<code>_draw_backend_rectangle:nnnn</code>	1567
<code>\draw_transform_shift_absolute:n</code>	1863, 1879, 1992	<code>_draw_backend_scope_begin:</code>	154, 1409
		157, 1419
		<code>_draw_backend_scope_end:</code>	157, 1419
		<code>\l_draw_baseline_bool</code>
		1385, 1389, 1394
		<code>\l__draw_baseline_dim</code>	1385, 1390, 1399
		<code>_draw_baseline_finalise:w</code>
		1334, 1392, 1392
		<code>_draw_box_use:Nnnnn</code>	14
		<code>_draw_box_use:nNnnnn</code>	21, 24, 79
		<code>_draw_box_use:Nnnnnnn</code>	16, 28, 31, 74
		<code>_draw_box_use:nNnnnnn</code>	14
		<code>_draw_coffin_use:nNnn</code>	72, 74, 79, 82
		<code>\l__draw_corner_arc_bool</code>
		242, 250, 251, 255, 581
		<code>\l__draw_corner_xarc_dim</code>
		240, 245, 248, 258

<code>\l__draw_corner_yarc_dim</code>	<code>__draw_path_arc_auxi:nnnnNnn</code> ...
..... 240 , 246 , 249 , 259 367 , 394 , 401 , 409 , 416 , 442
<code>__draw_draw_polar:nnn</code>	<code>__draw_path_arc_auxii:nnnNnnnn</code> .
..... 888 , 891 , 892 , 894 367 , 420 , 443
<code>_draw_draw_vec_polar:nnm</code>	<code>_draw_path_arc_auxiii:nn</code>
..... 1227 , 1228 , 1236 367 , 447 , 472
<code>\l__draw_fill_color_tl</code>	<code>__draw_path_arc_auxiv:nnnn</code>
..... 1404 367 , 453 , 478
<code>__draw_finalise:</code>	<code>__draw_path_arc_auxv:nn</code> 367 , 463 , 484
..... 1347 , 1354 , 1354 , 1392 , 1402	<code>_draw_path_arc_auxvi:nn</code>
<code>__draw_finalise_baseline:n</code> 367 , 486 , 489
..... 1354 , 1367 , 1399	<code>\l__draw_path_arc_delta_fp</code> 367
<code>\g__draw_id_int</code>	<code>\l__draw_path_arc_start_fp</code> 367
..... 1303 , 1314	<code>__draw_path_curveto:nnnn</code>
<code>__draw_if_recursion_tail_stop_</code> 324 , 327 , 331
<code>do:Nn</code>	<code>__draw_path_curveto:nnnnnn</code>
..... 9 , 9 , 1602 262 , 292 ,
<code>\l__draw_layer_close_bool</code>	298 , 320 , 338 , 468 , 537 , 547 , 557 , 567
..... 108 , 118 , 120 , 137	<code>\c__draw_path_curveto_a_fp</code> 324
<code>\l__draw_layer_main_box</code>	<code>\c__draw_path_curveto_b_fp</code> 324
..... 150 , 151 , 1301 , 1330	<code>__draw_path_ellipse:nnnnnn</code>
<code>\l__draw_layer_tl</code> 513 , 516 , 521
..... 106 , 117 , 121	<code>_draw_path_ellipse_arci:nnnnnn</code>
<code>\g__draw_layers_clist</code> 513 , 527 , 535
..... 109	<code>__draw_path_ellipse_arcii:nnnnnn</code>
<code>_draw_layers_insert:</code> 144 , 144 , 1337 513 , 528 , 545
<code>_draw_layers_restore:</code> 161 , 172 , 1452	<code>_draw_path_ellipse_arciiii:nnnnnn</code>
<code>_draw_layers_save:</code> . 161 , 161 , 1448 513 , 529 , 555
<code>\g__draw_linewidth_dim</code>	<code>_draw_path_ellipse_arciiv:nnnnnn</code>
..... 701 , 777 , 1411 , 1417 , 1805 , 1810 , 1811 513 , 530 , 565
<code>\l__draw_linewidth_dim</code>	<code>\c__draw_path_ellipse_fp</code>
..... 1404 , 1411 , 1417 513
<code>\l__draw_main_box</code>	<code>__draw_path_grid_auxi:nnnnnn</code> ...
..... 1301 , 1315 , 1348 , 1351 , 1356 , 1361 , 617 , 621 , 627 , 633
1363 , 1364 , 1369 , 1374 , 1376 , 1382	<code>_draw_path_grid_auxii:nnnnnn</code> .
<code>\l__draw_matrix_a_fp</code> 617 , 630 , 631 , 634
..... 54 ,	<code>_draw_path_grid_auxiii:nnnnnn</code> .
1249 , 1281 , 1842 , 1850 , 1865 , 1871 , 617 , 637 , 638 , 640
1902 , 1904 , 1927 , 1949 , 1958 , 1965	<code>_draw_path_grid_auxiiii:nnnnnn</code> 617
<code>\l__draw_matrix_active_bool</code>	<code>__draw_path_grid_auxiv:nnnnnnnn</code>
..... 582 , 1244 , 617 , 642 , 647 , 666
1276 , 1841 , 1854 , 1876 , 1877 , 1941	<code>\g__draw_path_lastx_dim</code>
<code>\l__draw_matrix_b_fp</code> 200 , 237 , 342 , 475 , 481 , 794 , 814
..... 55 ,	<code>\l__draw_path_lastx_dim</code> 782 , 794 , 814
1255 , 1286 , 1842 , 1851 , 1866 , 1872 ,	<code>\g__draw_path_lasty_dim</code>
1903 , 1905 , 1935 , 1950 , 1960 , 1961 200 , 238 , 349 , 476 , 482 , 795 , 815
<code>\l__draw_matrix_c_fp</code>	<code>\l__draw_path_lasty_dim</code> 782 , 795 , 815
..... 56 ,	<code>_draw_path_lineto:nn</code>
1250 , 1282 , 1842 , 1852 , 1867 , 1873 , 262 , 277 , 280 , 314
1902 , 1904 , 1927 , 1950 , 1962 , 1963	<code>_draw_path_mark_corner:</code>
<code>\l__draw_matrix_d_fp</code> 253 , 253 , 282 , 291 , 308 , 319 , 337 , 408
..... 57 ,	<code>__draw_path_moveto:nn</code>
1256 , 1287 , 1845 , 1853 , 1868 , 1874 , 262 , 265 , 268 , 312 , 525 , 533
1903 , 1905 , 1935 , 1949 , 1959 , 1964	<code>__draw_path_rectangle:nnnn</code>
<code>_draw_path_arc:nnnn</code> . 367 , 373 , 380 578 , 588 , 593
<code>_draw_path_arc:nnNnn</code>	
..... 367 , 383 , 384 , 386	
<code>\c__draw_path_arc_60_fp</code>	
..... 367	
<code>\c__draw_path_arc_90_fp</code>	
..... 367	
<code>_draw_path_arc_add:nnnn</code>	
..... 367	
<code>_draw_path_arc_aux_add:nn</code>	
..... 474 , 480 , 492 , 497	

<code>__draw_path_rectangle_corners:nmnn</code>	<code>__draw_point_interpolate_-</code>
..... 609	arcaxes_auxii:nmnnnnnnnn
<code>__draw_path_rectangle_corners:nmnnn</code> 1068 , 1076 , 1079 , 1085
..... 612 , 615	<code>__draw_point_interpolate_-</code>
<code>__draw_path_rectangle_rounded:nmnn</code>	arcaxes_auxiii:nmnnnnnn
..... 578 , 584 , 600 1068 , 1081 , 1086 , 1093
<code>__draw_path_replace_bb:NnN</code>	<code>__draw_point_interpolate_-</code>
..... 671 , 689 , 690 , 691 , 692 , 696	arcaxes_auxiv:nmnnnnnnnn
<code>__draw_path_reset_limits:</code> 1068 , 1088 , 1094 , 1099
... 206 , 228 , 683 , 694 , 748 , 802 , 1318	<code>__draw_point_interpolate_curve_-</code>
<code>\l__draw_path_tmp_tl</code>	auxi:nmnnnnnnnn
... 197 , 445 , 468 , 487 , 491 , 495 , 499 1103 , 1106
<code>\l__draw_path_tmpa_fp</code>	<code>__draw_point_interpolate_curve_-</code>
..... 197 , 333 , 343 , 355	auxii:nmnnnnnnnn
<code>\l__draw_path_tmpb_fp</code> 1108 , 1112 , 1120
..... 197 , 334 , 350 , 359	<code>__draw_point_interpolate_curve_-</code>
<code>__draw_path_update_last:n</code>	auxiii:nmnnnnnn
..... 235 , 235 , 272 , 285 , 304 , 598	auxiv:nmnnnnnn
<code>__draw_path_update_limits:n</code> 1128 , 1129 , 1130 , 1135
..... 36 , 39 , 42 , 45 , 206 ,	<code>__draw_point_interpolate_curve_-</code>
206 , 270 , 283 , 300 , 301 , 302 , 595 , 596	auxv:nw
<code>__draw_path_use:n</code> 1137 , 1141 , 1148
..... 671 , 674 , 685 , 704	<code>__draw_point_interpolate_curve_-</code>
<code>__draw_path_use_action_draw:</code>	auxvi:n
..... 671 , 751 1132 , 1153
<code>__draw_path_use_action_fillstroke:</code>	<code>__draw_point_interpolate_curve_-</code>
..... 671 , 755	auxvii:nmnnnnnnnn
<code>__draw_path_use_bb:NnN</code> 1154 , 1155
..... 671 , 762 , 763 , 764 , 765 , 767	<code>__draw_point_interpolate_curve_-</code>
<code>\l__draw_path_use_clear_bool</code> 670 , 745	auxviii:nmnnnnnn
<code>\l__draw_path_use_clip_bool</code> 1157 , 1164 , 1169
..... 667 , 706 , 724	<code>__draw_point_interpolate_-</code>
<code>\l__draw_path_use_fill_bool</code>	distance:nmnn
..... 667 , 707 , 729 , 734 , 740 , 757 1053 , 1056
<code>__draw_path_use_stroke_bb:</code>	<code>__draw_point_interpolate_-</code>
..... 671 , 722 , 760	distance:nmnnnn
<code>\l__draw_path_use_stroke_bool</code> 1050 , 1060 , 1065 , 1067
..... 667 , 708 , 721 , 730 , 735 , 741 , 753 , 758	<code>__draw_point_interpolate_-</code>
<code>\g__draw_path_xmax_dim</code>	distance:nmnnnnnn
..... 202 , 208 , 209 , 230 , 796 , 810 1050
<code>\l__draw_path_xmax_dim</code>	<code>__draw_point_interpolate_line_-</code>
..... 782 , 796 , 810	aux:nmnnnn
<code>\g__draw_path_xmin_dim</code> 1035 , 1038 , 1041 , 1046
..... 202 , 210 , 211 , 231 , 797 , 811	<code>__draw_point_interpolate_line_-</code>
<code>\l__draw_path_xmin_dim</code>	aux:nmnnnnnn
..... 782 , 797 , 811 1035 , 1043 , 1047 , 1049
<code>\g__draw_path_ymax_dim</code>	<code>__draw_point_intersect_circles_-</code>
..... 202 , 212 , 213 , 232 , 798 , 812	auxi:nmnnnnnn
<code>\l__draw_path_ymax_dim</code> 938 , 941 , 944
..... 202 , 214 , 215 , 233 , 799 , 813	<code>__draw_point_intersect_circles_-</code>
<code>\l__draw_path_ymin_dim</code>	auxii:nmnnnnnn
..... 782 , 799 , 813 938 , 946 , 949 , 956
<code>__draw_point_interpolate_-</code>	<code>__draw_point_intersect_circles_-</code>
arcaxes_auxi:nmnnnnnnnn	auxiii:nmnnnnnn
..... 1068 , 1071 , 1074 938 , 951 , 957 , 963
	<code>__draw_point_intersect_circles_-</code>
	auxiv:nmnnnnnnnn
 938 , 959 , 964 , 971
	<code>__draw_point_intersect_circles_-</code>
	auxv:nmnnnnnnnn
 938 , 966 , 972 , 978
	<code>__draw_point_intersect_circles_-</code>
	auxvi:nmnnnnnnnn
 938 , 974 , 979 , 987
	<code>__draw_point_intersect_circles_-</code>
	auxvii:nmnnnnnn
 938 , 981 , 988 , 993
	<code>__draw_point_intersect_line_-</code>
	circle_auxi:nmnnnnnnnn
 994 , 997 , 1000
	<code>__draw_point_intersect_line_-</code>
	circle_auxii:nmnnnnnnnn

..... 994, 1002, 1005, 1013
 _draw_point_intersect_line_-
 circle_auxiii:nnnnnnnn
 994, 1007, 1014, 1021
 _draw_point_intersect_line_-
 circle_auxiv:nnnnnnnn
 994, 1016, 1022, 1028
 _draw_point_intersect_line_-
 circle_auxv:nnnnn
 994, 1024, 1029, 1034
 _draw_point_intersect_lines:nnnnnn
 913
 _draw_point_intersect_lines:nnnnnnnn
 913, 916, 919
 _draw_point_intersect_lines_-
 aux:nnnnnn 913, 921, 929, 937
 _draw_point_process:nn
 35, 38, 41, 44, 264, 276, 312, 314,
 446, 462, 827, 827, 896, 1052, 1058,
 1184, 1239, 1271, 1881, 1910, 1977
 _draw_point_process:nnn
 326, 452, 584,
 588, 611, 619, 827, 838, 940, 1037, 1975
 _draw_point_process:nnnn
 ... 289, 317, 515, 827, 850, 996, 1070
 _draw_point_process:nnnnn
 827, 864, 915, 1102
 _draw_point_process_auxi:nn
 827, 829, 833, 835
 _draw_point_process_auxii:nw ..
 827, 834, 836
 _draw_point_process_auxiii:nnn
 827, 840, 845, 847
 _draw_point_process_auxiv:nw ..
 827, 846, 848
 _draw_point_process_auxv:nnnn ..
 827, 852, 858, 860
 _draw_point_process_auxvi:nw ..
 827, 859, 861
 _draw_point_process_auxvii:nnnnn
 827, 866, 873, 878
 _draw_point_process_auxviii:nw
 827, 875, 879
 _draw_point_to_dim:n
 882, 883, 884, 886
 _draw_point_to_dim:w . 882, 885, 887
 _draw_point_transform:nn
 1237, 1240, 1242
 _draw_point_transform_noshift:n
 449, 465, 518, 519, 1269, 1269
 _draw_point_transform_noshift:nn
 1269, 1272, 1274
 _draw_point_unit_vector:nn
 895, 896, 897
 _draw_point_unit_vector:nnn ...
 895, 899, 903, 912
 _draw_point_vec:nn
 1194, 1195, 1196, 1204
 _draw_point_vec:nnn
 1194, 1207, 1210, 1223
 _draw_point_vec_polar:nnn .. 1224
 _draw_reset_bb:
 1304, 1304, 1317, 1432
 _draw_scope_bb_begin:
 1425, 1425, 1444
 _draw_scope_bb_end: 1425, 1434, 1454
 _draw_softpath_add:n
 ... 1462, 1462, 1464, 1491, 1502,
 1511, 1520, 1525, 1535, 1543, 1800
 \c_draw_softpath_arc_fp
 1579, 1742, 1746, 1751, 1755
 _draw_softpath_clear: 682,
 693, 747, 1322, 1465, 1472, 1486, 1490
 _draw_softpath_close_op:nn ...
 .. 1504, 1548, 1548, 1641, 1677, 1779
 _draw_softpath_closepath:
 309, 532, 1500, 1500
 \l_draw_softpath_corneri_dim ...
 1573, 1626, 1629, 1715
 \l_draw_softpath_cornerii_dim ..
 1573, 1627, 1630, 1706
 \g_draw_softpath_corners_bool ..
 1461,
 1475, 1485, 1493, 1545, 1582, 1598
 \l_draw_softpath_corners_bool ..
 782, 1484, 1494
 \l_draw_softpath_curve_end_tl ..
 1572, 1703, 1722, 1771, 1782
 _draw_softpath_curveto:nnnnnn ..
 303, 1500, 1509
 _draw_softpath_curveto_opi:nn ..
 .. 1513, 1548, 1550, 1638, 1676, 1739
 _draw_softpath_curveto_-
 opi:nnNnnNnn 1548, 1551, 1552
 _draw_softpath_curveto_opiii:nn
 1514, 1548, 1554, 1555, 1748
 _draw_softpath_curveto_-
 opiii:nn 1515, 1548, 1556, 1557, 1757
 \l_draw_softpath_first_tl
 1573, 1589, 1606,
 1607, 1617, 1636, 1637, 1663, 1667
 \g_draw_softpath_lastx_dim
 800, 808, 1496, 1505, 1529
 \l_draw_softpath_lastx_dim
 788, 800, 808
 \l_draw_softpath_lastx_fp
 .. 1573, 1587, 1608, 1656, 1718, 1725

\g__draw_softpath_lasty_dim	__draw_softpath_round_close:w
. 801, 809, 1496 , 1506 , 1530 1580 , 1770 , 1781 , 1792
\l__draw_softpath_lasty_dim	__draw_softpath_round_corners:
. 789, 801, 809 718, 1580 , 1580
\l__draw_softpath_lasty_fp	__draw_softpath_round_end:
. 1573 , 1588 , 1609 , 1657 , 1719 , 1726 1580 , 1602 , 1793
__draw_softpath_lineto:nn	__draw_softpath_round_lookahead:NnnNnn
. 284, 1500 , 1518 1580 , 1645 , 1658 , 1671
__draw_softpath_lineto_op:nn	__draw_softpath_round_loop:Nnn
. 1521 , 1548 , 1558 , 1644 , 1675 , 1788 1580 , 1592 , 1600 , 1621 , 1631 ,
\g__draw_softpath_main_tl	1646 , 1669 , 1681 , 1687 , 1696 , 1778
. 1459 , 1463 , 1468 ,	__draw_softpath_round_roundpoint:NnnNnnNnn
1474 , 1479 , 1482 , 1591 , 1593 , 1799 1580 , 1686 , 1692
\l__draw_softpath_main_tl	__draw_softpath_roundpoint:nn
. 20, 1481 , 1491 , 1570 , 257, 1500 , 1541 , 1547
1585 , 1612 , 1614 , 1795 , 1797 , 1800	__draw_softpath_roundpoint_-
\g__draw_softpath_move_bool	op:nn
. 1498 , 1527 1544 , 1548 , 1562 , 1563 , 1603 , 1685
\l__draw_softpath_move_tl	__draw_softpath_save: 803, 1477 , 1477
. 1573 , 1590 ,	\l__draw_softpath_tmp_tl
1613 , 1616 , 1664 , 1766 , 1789 , 1796 1460 , 1469 , 1470
__draw_softpath_moveto:nn	__draw_softpath_use: 723, 1465 , 1465
. 271, 1500 , 1523	\l__draw_stroke_color_tl 1404
__draw_softpath_moveto_op:nn	\l__draw_tmp_box 13 , 49 , 60 ,
. 1526 , 1548 , 1560 , 1610 , 1768	64 , 66 , 67 , 68 , 69 , 85 , 87 , 88 , 89 , 90 , 91
\l__draw_softpath_part_tl	\l__draw_tmp_seq 1813
. 1571 , 1586 ,	__draw_transform:nnnn
1615 , 1618 , 1620 , 1654 , 1709 , 1798 1886 , 1890 , 1897
__draw_softpath_rectangle:nnnn	__draw_transform_invert:n
. 597, 1500 , 1533 1939 , 1943 , 1956 , 1967
__draw_softpath_rectangle_-	__draw_transform_rotate:n
opi:nn 1537 , 1548 , 1564 2009, 2010, 2011, 2017
__draw_softpath_rectangle_-	__draw_transform_rotate:nn
opi:nnNnn 1548 , 1565 , 1566 2009, 2013, 2018, 2020
__draw_softpath_rectangle_-	__draw_transform_shift:nn
opii:nn 1538 , 1548 , 1568 , 1569 1886 , 1911 , 1913
__draw_softpath_restore:	__draw_transform_shift:nnnn
. 807, 1477 , 1488 1885 , 1915 , 1920 , 1920
__draw_softpath_round_action:nn	__draw_transform_shift_absolute:nn
. 1580 , 1604 , 1624 1863 , 1882 , 1884
__draw_softpath_round_action:Nnn	__draw_transform_triangle:nnnnnn
. 1580 , 1632 , 1634
__draw_softpath_round_action_-	__draw_vec:nn
close: 1580 , 1642 , 1660 1176 , 1177 , 1179 , 1181 , 1182
__draw_softpath_round_action_-	__draw_vec:nnn 1176 , 1184 , 1186
curveto:NnnNnn 1580 , 1639 , 1651	\g__draw_xmax_dim 218,
__draw_softpath_round_calc:NnnNnn	219 , 1296 , 1306 , 1342 , 1349 , 1428 , 1436
. 1580 , 1680 , 1695 , 1701 , 1764	\l__draw_xmax_dim 1421 , 1428 , 1436
__draw_softpath_round_calc:nnnnnn	\g__draw_xmin_dim
. 1580 , 1712 , 1729 , 1734 220, 221, 1296 , 1307 , 1340 ,
__draw_softpath_round_calc:nnnnw	1343 , 1349 , 1358 , 1371 , 1429 , 1437
. 1580 , 1731 , 1735	\l__draw_xmin_dim 1421 , 1429 , 1437
__draw_softpath_round_close:nn	\l__draw_xshift_dim 62, 589, 1251 ,
. 1580 , 1666 , 1760	1265 , 1842 , 1858 , 1916 , 1922 , 1970

<code>\l__draw_xvec_x_dim</code>	<code>\fp_to_dim:n</code>
..... 1170, 1200, 1214, 1232	245, 246, 340, 347, 354, 358, 376, 377,
<code>\l__draw_xvec_y_dim</code> . 1170, 1201, 1218	423, 432, 500, 526, 538, 539, 540,
<code>\g__draw_yvec_x_dim</code> . 222, 223, 1296,	541, 542, 543, 548, 549, 550, 551,
1308, 1344, 1365, 1383, 1430, 1438	552, 553, 558, 559, 560, 561, 562,
<code>\l__draw_yvec_y_dim</code> ... 1421, 1430, 1438	563, 568, 569, 570, 571, 572, 573,
<code>\g__draw_ymin_dim</code>	643, 644, 1390, 1741, 1745, 1750,
..... 224, 225, 1296, 1309,	1754, 1810, 1818, 1823, 1924, 1932
1345, 1360, 1365, 1379, 1431, 1439	<code>\fp_use:N</code>
<code>\l__draw_ymin_dim</code> ... 1421, 1431, 1439	54, 55, 56, 57, 575
<code>\l__draw_yshift_dim</code> 63, 589, 1257,	<code>\fp_while_do:nNnn</code>
1265, 1842, 1859, 1917, 1930, 1971	390
<code>\l__draw_yvec_x_dim</code> . 1170, 1200, 1215	<code>\fp_zero:N</code> 1587, 1588, 1851, 1852
<code>\l__draw_yvec_y_dim</code>	<code>\c_one_fp</code>
..... 1170, 1201, 1219, 1233	1871, 1874
<code>\l__draw_zvec_x_dim</code> 1170, 1216	<code>\c_zero_fp</code>
<code>\l__draw_zvec_y_dim</code> 1170, 1220	905, 1872, 1873
E	
<code>\end</code>	
191, 821	
exp commands:	
<code>\exp_after:wN</code>	
468, 486, 1592, 1666, 1769, 1780, 1789	
<code>\exp_args:Ne</code>	
1829	
<code>\exp_args:NNNV</code>	
1335	
<code>\exp_not:N</code>	
1711, 1739, 1748, 1757, 1766, 1769,	
1770, 1771, 1776, 1780, 1781, 1782	
<code>\exp_not:n</code>	
1398	
F	
fp commands:	
<code>\fp_compare:nNnTF</code> 382, 392, 905	
<code>\fp_compare_p:nNn</code>	
..... 1871, 1872, 1873, 1874	
<code>\fp_const:Nn</code>	
..... 365, 366, 504, 505, 575, 1579	
<code>\fp_eval:n</code>	
..... 374, 375, 396, 403, 412, 883,	
891, 900, 922, 923, 924, 925, 926,	
927, 947, 952, 953, 960, 967, 968,	
975, 982, 984, 1003, 1008, 1009,	
1010, 1017, 1025, 1038, 1043, 1061,	
1077, 1082, 1089, 1090, 1109, 1116,	
1138, 1139, 1158, 1159, 1160, 1161,	
1195, 1208, 1227, 1829, 1891, 1892,	
1893, 1894, 1945, 2010, 2014, 2015	
<code>\fp_new:N</code>	
198, 199, 502,	
503, 1573, 1574, 1842, 1843, 1844, 1845	
<code>\fp_set:Nn</code> 333, 334, 388, 389,	
469, 470, 1608, 1609, 1656, 1657,	
1725, 1726, 1850, 1853, 1865, 1866,	
1867, 1868, 1958, 1960, 1962, 1964	
<code>\fp_to_decimal:N</code> 395, 402, 410	
	<code>\fp_to_dim:n</code>
	245,
	246, 340, 347, 354, 358, 376, 377,
	423, 432, 500, 526, 538, 539, 540,
	541, 542, 543, 548, 549, 550, 551,
	552, 553, 558, 559, 560, 561, 562,
	563, 568, 569, 570, 571, 572, 573,
	643, 644, 1390, 1741, 1745, 1750,
	1754, 1810, 1818, 1823, 1924, 1932
	<code>\fp_use:N</code>
	54, 55, 56, 57, 575
	<code>\fp_while_do:nNnn</code>
	390
	<code>\fp_zero:N</code> 1587, 1588, 1851, 1852
	<code>\c_one_fp</code>
	1871, 1874
	<code>\c_zero_fp</code>
	905, 1872, 1873
G	
group commands:	
<code>\group_begin:</code> ... 48, 84, 114, 125,	
508, 793, 1313, 1410, 1427, 1584, 1815	
<code>\group_end:</code> 70, 92, 139, 142,	
511, 816, 1352, 1418, 1440, 1596, 1825	
H	
hbox commands:	
<code>\hbox_gset:Nw</code>	123
<code>\hbox_gset_end:</code>	140
<code>\hbox_set:Nn</code> ... 49, 60, 85, 1356, 1369	
<code>\hbox_set:Nw</code>	1315, 1330
<code>\hbox_set_end:</code>	1335, 1339
I	
int commands:	
<code>\int_gincr:N</code>	1314
<code>\int_if_odd:nTF</code>	983, 1018
<code>\int_new:N</code>	1303
K	
kernel internal commands:	
<code>__kernel_kern:n</code>	62
<code>__kernel_quark_new_test:N</code>	9
M	
mode commands:	
<code>\mode_leave_vertical:</code>	1350
msg commands:	
<code>\msg_error:nnn</code> 100, 131, 132, 715	
<code>\msg_new:nnn</code>	186
<code>\msg_new:nnnn</code>	183, 188, 818
P	
<code>\pgfextractx</code>	21
<code>\pgfextracty</code>	21
<code>\pgfgetlastxy</code>	21
<code>\pgfgettransform</code>	51
<code>\pgfgettransformentries</code>	51
<code>\pgfinnerlinewidth</code>	50

