

File I

Implementation

1 l3draw implementation

```

1 <*package>
2 <@@=draw>
3 \ProvidesExplPackage{l3draw}{2023-10-10}{ }
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: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:Nnnnn #1
17     { Opt } { -\box_dp:N #1 } { \box_wd:N #1 } { \box_ht:N #1 }
18 }
19 \cs_new_protected:Npn \__draw_box_use:Nnnnn #1#2#3#4#5
20 {
21     \bool_if:NT \l_draw_bb_update_bool
22     {
23         \__draw_point_process:nn
24         { \__draw_path_update_limits:nn }
25         { \draw_point_transform:n { #2 , #3 } }
26         \__draw_point_process:nn
27         { \__draw_path_update_limits:nn }
28         { \draw_point_transform:n { #4 , #3 } }
29         \__draw_point_process:nn
30         { \__draw_path_update_limits:nn }
31         { \draw_point_transform:n { #4 , #5 } }
32         \__draw_point_process:nn
33         { \__draw_path_update_limits:nn }
34         { \draw_point_transform:n { #2 , #5 } }
35     }
36     \group_begin:
37     \hbox_set:Nn \l__draw_tmp_box
38     {
39         \use:e
40         {
41             \__draw_backend_box_use:Nnnnn #1
42             { \fp_use:N \l__draw_matrix_a_fp }
43             { \fp_use:N \l__draw_matrix_b_fp }
44             { \fp_use:N \l__draw_matrix_c_fp }
45             { \fp_use:N \l__draw_matrix_d_fp }
46         }
47     }
48     \hbox_set:Nn \l__draw_tmp_box
49     {
50         \__kernel_kern:n { \l__draw_xshift_dim }
51         \box_move_up:nn { \l__draw_yshift_dim }
52         { \box_use_drop:N \l__draw_tmp_box }
53     }
54     \box_set_ht:Nn \l__draw_tmp_box { Opt }
55     \box_set_dp:Nn \l__draw_tmp_box { Opt }
56     \box_set_wd:Nn \l__draw_tmp_box { Opt }
57     \box_use_drop:N \l__draw_tmp_box
58     \group_end:
59 }

```

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

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

```

60 \cs_new_protected:Npn \draw_coffin_use:Nnn #1#2#3
61 {
62     \group_begin:
63     \hbox_set:Nn \l__draw_tmp_box

```

```

64         { \coffin_typeset:Nnnnn #1 {#2} {#3} { Opt } { Opt } }
65     \__draw_box_use:Nnnnn \l__draw_tmp_box
66     { \box_wd:N \l__draw_tmp_box - \coffin_wd:N #1 }
67     { -\box_dp:N \l__draw_tmp_box }
68     { \box_wd:N \l__draw_tmp_box }
69     { \box_ht:N \l__draw_tmp_box }
70     \group_end:
71 }

```

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

```

72 \</package>

```

3 l3draw-layers implementation

```

73 \*package>

```

```

74 \@@=draw>

```

3.1 User interface

\draw_layer_new:n

```

75 \cs_new_protected:Npn \draw_layer_new:n #1
76 {
77     \str_if_eq:nnTF {#1} { main }
78     { \msg_error:nnn { draw } { main-reserved } }
79     {
80         \box_new:c { g__draw_layer_ #1 _box }
81         \box_new:c { l__draw_layer_ #1 _box }
82     }
83 }

```

(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.

```

84 \tl_new:N \l__draw_layer_tl
85 \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.

```

86 \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
87 \clist_new:N \l_draw_layers_clist
88 \clist_set:Nn \l_draw_layers_clist { main }
89 \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.

```

90 \cs_new_protected:Npn \draw_layer_begin:n #1
91 {
92   \group_begin:
93   \box_if_exist:cTF { g__draw_layer_ #1 _box }
94   {
95     \str_if_eq:VnTF \l__draw_layer_tl {#1}
96     { \bool_set_false:N \l__draw_layer_close_bool }
97     {
98       \bool_set_true:N \l__draw_layer_close_bool
99       \tl_set:Nn \l__draw_layer_tl {#1}
100       \box_gset_wd:cn { g__draw_layer_ #1 _box } { Opt }
101       \hbox_gset_cw { g__draw_layer_ #1 _box }
102       \box_use_drop:c { g__draw_layer_ #1 _box }
103       \group_begin:
104     }
105     \draw_linewidth:n { \l_draw_default_linewidth_dim }
106   }
107   {
108     \str_if_eq:nnTF {#1} { main }
109     { \msg_error:nnn { draw } { unknown-layer } {#1} }
110     { \msg_error:nnn { draw } { main-layer } }
111   }
112 }
113 \cs_new_protected:Npn \draw_layer_end:
114 {
115   \bool_if:NT \l__draw_layer_close_bool
116   {
117     \group_end:
118     \hbox_gset_end:
119   }
120   \group_end:
121 }

```

(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.

```

122 \cs_new_protected:Npn \__draw_layers_insert:
123 {
124   \clist_map_inline:Nn \l_draw_layers_clist
125   {
126     \str_if_eq:nnTF {##1} { main }
127     {
128       \box_set_wd:Nn \l__draw_layer_main_box { Opt }
129       \box_use_drop:N \l__draw_layer_main_box
130     }
131     {
132       \__draw_backend_scope_begin:
133       \box_gset_wd:cn { g__draw_layer_ ##1 _box } { Opt }

```

```

134         \box_use_drop:c { g__draw_layer_ ##1 _box }
135         \__draw_backend_scope_end:
136     }
137 }
138 }

```

(End of definition for __draw_layers_insert:.)

__draw_layers_save: Simple save/restore functions.

```

\__draw_layers_restore:
139 \cs_new_protected:Npn \__draw_layers_save:
140 {
141     \clist_map_inline:Nn \l_draw_layers_clist
142     {
143         \str_if_eq:nnF {##1} { main }
144         {
145             \box_set_eq:cc { l__draw_layer_ ##1 _box }
146             { g__draw_layer_ ##1 _box }
147         }
148     }
149 }
150 \cs_new_protected:Npn \__draw_layers_restore:
151 {
152     \clist_map_inline:Nn \l_draw_layers_clist
153     {
154         \str_if_eq:nnF {##1} { main }
155         {
156             \box_gset_eq:cc { g__draw_layer_ ##1 _box }
157             { l__draw_layer_ ##1 _box }
158         }
159     }
160 }

```

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

```

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

```

4 l3draw-paths implementation

```

173 \*package
174 \@@=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 175 \tl_new:N \l__draw_path_tmp_tl
\l__draw_path_tmpb_fp 176 \fp_new:N \l__draw_path_tmpa_fp
177 \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 178 \dim_new:N \g__draw_path_lastx_dim
179 \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 180 \dim_new:N \g__draw_path_xmax_dim
\g__draw_path_ymax_dim 181 \dim_new:N \g__draw_path_xmin_dim
\g__draw_path_ymin_dim 182 \dim_new:N \g__draw_path_ymax_dim
183 \dim_new:N \g__draw_path_ymin_dim

```

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

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

```

\__draw_path_reset_limits: 184 \cs_new_protected:Npn \__draw_path_update_limits:nn #1#2
185 {
186   \dim_gset:Nn \g__draw_path_xmax_dim
187     { \dim_max:nn \g__draw_path_xmax_dim {#1} }
188   \dim_gset:Nn \g__draw_path_xmin_dim
189     { \dim_min:nn \g__draw_path_xmin_dim {#1} }
190   \dim_gset:Nn \g__draw_path_ymax_dim
191     { \dim_max:nn \g__draw_path_ymax_dim {#2} }
192   \dim_gset:Nn \g__draw_path_ymin_dim
193     { \dim_min:nn \g__draw_path_ymin_dim {#2} }
194   \bool_if:NT \l_draw_bb_update_bool
195   {
196     \dim_gset:Nn \g__draw_xmax_dim
197       { \dim_max:nn \g__draw_xmax_dim {#1} }
198     \dim_gset:Nn \g__draw_xmin_dim
199       { \dim_min:nn \g__draw_xmin_dim {#1} }

```

```

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

```

(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.

```

213 \cs_new_protected:Npn \__draw_path_update_last:nn #1#2
214 {
215     \dim_gset:Nn \g__draw_path_lastx_dim {#1}
216     \dim_gset:Nn \g__draw_path_lasty_dim {#2}
217 }

```

(End of definition for __draw_path_update_last:nn.)

4.2 Corner arcs

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

\l__draw_corner_xarc_dim The two arcs in use.

```

\l__draw_corner_yarc_dim
218 \dim_new:N \l__draw_corner_xarc_dim
219 \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.

```

220 \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.

```

221 \cs_new_protected:Npn \draw_path_corner_arc:nn #1#2
222 {
223     \dim_set:Nn \l__draw_corner_xarc_dim {#1}
224     \dim_set:Nn \l__draw_corner_yarc_dim {#2}
225     \bool_lazy_and:nnTF
226     { \dim_compare_p:nNn \l__draw_corner_xarc_dim = { 0pt } }
227     { \dim_compare_p:nNn \l__draw_corner_yarc_dim = { 0pt } }
228     { \bool_set_false:N \l__draw_corner_arc_bool }
229     { \bool_set_true:N \l__draw_corner_arc_bool }
230 }

```

(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.
231 \cs_new_protected:Npn \__draw_path_mark_corner:
232 {
233   \bool_if:NT \l__draw_corner_arc_bool
234   {
235     \__draw_softpath_roundpoint:VV
236     \l__draw_corner_xarc_dim
237     \l__draw_corner_yarc_dim
238   }
239 }

```

(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 240 \cs_new_protected:Npn \draw_path_moveto:n #1
\__draw_path_lineto:nn 241 {
\draw_path_curveto:nnn 242   \__draw_point_process:nn
\__draw_path_curveto:nnnnnn 243   { \__draw_path_moveto:nn }
244   { \draw_point_transform:n {#1} }
245 }
246 \cs_new_protected:Npn \__draw_path_moveto:nn #1#2
247 {
248   \__draw_path_update_limits:nn {#1} {#2}
249   \__draw_softpath_moveto:nn {#1} {#2}
250   \__draw_path_update_last:nn {#1} {#2}
251 }
252 \cs_new_protected:Npn \draw_path_lineto:n #1
253 {
254   \__draw_point_process:nn
255   { \__draw_path_lineto:nn }
256   { \draw_point_transform:n {#1} }
257 }
258 \cs_new_protected:Npn \__draw_path_lineto:nn #1#2
259 {
260   \__draw_path_mark_corner:
261   \__draw_path_update_limits:nn {#1} {#2}
262   \__draw_softpath_lineto:nn {#1} {#2}
263   \__draw_path_update_last:nn {#1} {#2}
264 }
265 \cs_new_protected:Npn \draw_path_curveto:nnn #1#2#3
266 {
267   \__draw_point_process:nnnn
268   {
269     \__draw_path_mark_corner:
270     \__draw_path_curveto:nnnnnn
271   }
272   { \draw_point_transform:n {#1} }
273   { \draw_point_transform:n {#2} }
274   { \draw_point_transform:n {#3} }

```

```

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

```

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

\draw_path_close: A simple wrapper.

```

284 \cs_new_protected:Npn \draw_path_close:
285 {
286   \__draw_path_mark_corner:
287   \__draw_softpath_closepath:
288 }

```

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

```

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

4.5 Computed curves

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

```

\draw_path_curveto:nn
\__draw_path_curveto:nnnn
\c__draw_path_curveto_a_fp
\c__draw_path_curveto_b_fp

```

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

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

and

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

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

```

302 \cs_new_protected:Npn \draw_path_curveto:nn #1#2
303 {
304   \__draw_point_process:nnn
305   { \__draw_path_curveto:nnnn }
306   { \draw_point_transform:n {#1} }
307   { \draw_point_transform:n {#2} }
308 }
309 \cs_new_protected:Npn \__draw_path_curveto:nnnn #1#2#3#4
310 {
311   \fp_set:Nn \l__draw_path_tmpa_fp { \c__draw_path_curveto_b_fp * #1 }
312   \fp_set:Nn \l__draw_path_tmpb_fp { \c__draw_path_curveto_b_fp * #2 }
313   \use:e
314   {
315     \__draw_path_mark_corner:
316     \__draw_path_curveto:nnnnnn
317     {
318       \fp_to_dim:n
319       {
320         \c__draw_path_curveto_a_fp * \g__draw_path_lastx_dim
321         + \l__draw_path_tmpa_fp
322       }
323     }
324     {
325       \fp_to_dim:n
326       {
327         \c__draw_path_curveto_a_fp * \g__draw_path_lasty_dim
328         + \l__draw_path_tmpb_fp
329       }
330     }
331     {
332       \fp_to_dim:n
333       { \c__draw_path_curveto_a_fp * #3 + \l__draw_path_tmpa_fp }
334     }
335     {
336       \fp_to_dim:n
337       { \c__draw_path_curveto_a_fp * #4 + \l__draw_path_tmpb_fp }
338     }
339     {#3}
340     {#4}
341   }
342 }
343 \fp_const:Nn \c__draw_path_curveto_a_fp { 1 / 3 }
344 \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 ??.)

<pre> \draw_path_arc:nnn \draw_path_arc:nnnn __draw_path_arc:nnnn __draw_path_arc:nnNnn __draw_path_arc_auxi:nnnnNnn __draw_path_arc_auxi:fnnnNnn __draw_path_arc_auxi:fnfnNnn __draw_path_arc_auxii:nnnNnnnn __draw_path_arc_auxiii:nn __draw_path_arc_auxiv:nnnn __draw_path_arc_auxv:nn __draw_path_arc_auxvi:nn __draw_path_arc_add:nnnn \l__draw_path_arc_delta_fp \l__draw_path_arc_start_fp \c__draw_path_arc_90_fp \c__draw_path_arc_60_fp </pre>	<p>Drawing an arc means dividing the total curve required into sections: using Bézier curves we can cover at most 90° at once. To allow for later manipulations, we aim to have roughly equal last segments to the line, with the split set at a final part of 115°.</p> <pre> 345 \cs_new_protected:Npn \draw_path_arc:nnn #1#2#3 346 { \draw_path_arc:nnnn {#1} {#2} {#3} {#3} } 347 \cs_new_protected:Npn \draw_path_arc:nnnn #1#2#3#4 348 { 349 \use:e </pre>
--	---

```

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

```

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.

```

394 \cs_new_protected:Npn \__draw_path_arc_auxi:nnnnNnn #1#2#3#4#5#6#7
395 {
396     \use:e
397     {
398         \__draw_path_arc_auxii:nnnNnnnn

```

```

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

```

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

```

421 \cs_new_protected:Npn \__draw_path_arc_auxii:nnnnNnnn #1#2#3#4#5#6#7#8
422 {
423     \tl_clear:N \l__draw_path_tmp_tl
424     \__draw_point_process:nn
425     { \__draw_path_arc_auxiii:nn }
426     {
427         \__draw_point_transform_noshift:n
428         { \draw_point_polar:nnn {#7} {#8} { #1 #4 90 } }
429     }
430     \__draw_point_process:nnn
431     { \__draw_path_arc_auxiv:nnnn }
432     {
433         \draw_point_transform:n
434         { \draw_point_polar:nnn {#5} {#6} {#1} }
435     }
436     {
437         \draw_point_transform:n
438         { \draw_point_polar:nnn {#5} {#6} {#2} }
439     }
440     \__draw_point_process:nn
441     { \__draw_path_arc_auxv:nn }
442     {
443         \__draw_point_transform_noshift:n
444         { \draw_point_polar:nnn {#7} {#8} { #2 #4 -90 } }
445     }
446     \exp_after:wN \__draw_path_curveto:nnnnnn \l__draw_path_tmp_tl

```

```

447 \fp_set:Nn \l__draw_path_arc_delta_fp { abs ( #2 - #3 ) }
448 \fp_set:Nn \l__draw_path_arc_start_fp {#2}
449 }

```

The first control point.

```

450 \cs_new_protected:Npn \__draw_path_arc_auxiii:nn #1#2
451 {
452   \__draw_path_arc_aux_add:nn
453   { \g__draw_path_lastx_dim + #1 }
454   { \g__draw_path_lasty_dim + #2 }
455 }

```

The end point: simple arithmetic.

```

456 \cs_new_protected:Npn \__draw_path_arc_auxiv:nnnn #1#2#3#4
457 {
458   \__draw_path_arc_aux_add:nn
459   { \g__draw_path_lastx_dim - #1 + #3 }
460   { \g__draw_path_lasty_dim - #2 + #4 }
461 }

```

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

```

462 \cs_new_protected:Npn \__draw_path_arc_auxv:nn #1#2
463 {
464   \exp_after:wN \__draw_path_arc_auxvi:nn
465   \l__draw_path_tmp_tl {#1} {#2}
466 }
467 \cs_new_protected:Npn \__draw_path_arc_auxvi:nn #1#2#3#4#5#6
468 {
469   \tl_set:Nn \l__draw_path_tmp_tl { {#1} {#2} }
470   \__draw_path_arc_aux_add:nn
471   { #5 + #3 }
472   { #6 + #4 }
473   \tl_put_right:Nn \l__draw_path_tmp_tl { {#3} {#4} }
474 }
475 \cs_new_protected:Npn \__draw_path_arc_aux_add:nn #1#2
476 {
477   \tl_put_right:Ne \l__draw_path_tmp_tl
478   { { \fp_to_dim:n {#1} } { \fp_to_dim:n {#2} } }
479 }
480 \fp_new:N \l__draw_path_arc_delta_fp
481 \fp_new:N \l__draw_path_arc_start_fp
482 \fp_const:cn { c__draw_path_arc_90_fp } { 4/3 * (sqrt(2) - 1) }
483 \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.

```

484 \cs_new_protected:Npn \draw_path_arc_axes:nnnn #1#2#3#4
485 {
486   \group_begin:
487   \draw_transform_triangle:nnn { 0cm , 0cm } {#3} {#4}
488   \draw_path_arc:nnn {#1} {#2} { 1pt }
489   \group_end:
490 }

```

(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
`__draw_path_ellipse:nnnnnn` same constant. We need to deal with the ellipse in four parts and also deal with moving
`__draw_path_ellipse_arci:nnnnnn` to the right place, closing it and ending up back at the center. That is handled on a
`__draw_path_ellipse_arcii:nnnnnn` per-arc basis, each in a separate auxiliary for readability.
`__draw_path_ellipse_arciiii:nnnnnn`
`__draw_path_ellipse_arciv:nnnnnn`
`\c__draw_path_ellipse_fp`

```

491 \cs_new_protected:Npn \draw_path_ellipse:nnn #1#2#3
492 {
493   \__draw_point_process:nnnn
494   { \__draw_path_ellipse:nnnnnn }
495   { \draw_point_transform:n {#1} }
496   { \__draw_point_transform_noshift:n {#2} }
497   { \__draw_point_transform_noshift:n {#3} }
498 }
499 \cs_new_protected:Npn \__draw_path_ellipse:nnnnnn #1#2#3#4#5#6
500 {
501   \use:e
502   {
503     \__draw_path_moveto:nn
504     { \fp_to_dim:n { #1 + #3 } } { \fp_to_dim:n { #2 + #4 } }
505     \__draw_path_ellipse_arci:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
506     \__draw_path_ellipse_arcii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
507     \__draw_path_ellipse_arciiii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
508     \__draw_path_ellipse_arciv:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
509   }
510   \__draw_softpath_closepath:
511   \__draw_path_moveto:nn {#1} {#2}
512 }
513 \cs_new:Npn \__draw_path_ellipse_arci:nnnnnn #1#2#3#4#5#6
514 {
515   \__draw_path_curveto:nnnnnn
516   { \fp_to_dim:n { #1 + #3 + #5 * \c__draw_path_ellipse_fp } }
517   { \fp_to_dim:n { #2 + #4 + #6 * \c__draw_path_ellipse_fp } }
518   { \fp_to_dim:n { #1 + #3 * \c__draw_path_ellipse_fp + #5 } }
519   { \fp_to_dim:n { #2 + #4 * \c__draw_path_ellipse_fp + #6 } }
520   { \fp_to_dim:n { #1 + #5 } }
521   { \fp_to_dim:n { #2 + #6 } }
522 }
523 \cs_new:Npn \__draw_path_ellipse_arcii:nnnnnn #1#2#3#4#5#6
524 {
525   \__draw_path_curveto:nnnnnn
526   { \fp_to_dim:n { #1 - #3 * \c__draw_path_ellipse_fp + #5 } }
527   { \fp_to_dim:n { #2 - #4 * \c__draw_path_ellipse_fp + #6 } }
528   { \fp_to_dim:n { #1 - #3 + #5 * \c__draw_path_ellipse_fp } }
529   { \fp_to_dim:n { #2 - #4 + #6 * \c__draw_path_ellipse_fp } }
530   { \fp_to_dim:n { #1 - #3 } }
531   { \fp_to_dim:n { #2 - #4 } }
532 }
533 \cs_new:Npn \__draw_path_ellipse_arciiii:nnnnnn #1#2#3#4#5#6
534 {
535   \__draw_path_curveto:nnnnnn
536   { \fp_to_dim:n { #1 - #3 - #5 * \c__draw_path_ellipse_fp } }
537   { \fp_to_dim:n { #2 - #4 - #6 * \c__draw_path_ellipse_fp } }
538   { \fp_to_dim:n { #1 - #3 * \c__draw_path_ellipse_fp - #5 } }
539   { \fp_to_dim:n { #2 - #4 * \c__draw_path_ellipse_fp - #6 } }
540   { \fp_to_dim:n { #1 - #5 } }

```

```

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

```

554   \cs_new_protected:Npn \draw_path_circle:nn #1#2
555   { \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

```

556   \cs_new_protected:Npn \draw_path_rectangle:nn #1#2
557   {
558     \__draw_point_process:nnn
559     {
560       \bool_lazy_or:nnTF
561       { \l__draw_corner_arc_bool }
562       { \l__draw_matrix_active_bool }
563       { \__draw_path_rectangle_rounded:nnnn }
564       { \__draw_path_rectangle:nnnn }
565     }
566     { \draw_point_transform:n {#1} }
567     {#2}
568   }
569   \cs_new_protected:Npn \__draw_path_rectangle:nnnn #1#2#3#4
570   {
571     \__draw_path_update_limits:nn {#1} {#2}
572     \__draw_path_update_limits:nn { #1 + #3 } { #2 + #4 }
573     \__draw_softpath_rectangle:nnnn {#1} {#2} {#3} {#4}
574     \__draw_path_update_last:nn {#1} {#2}
575   }
576   \cs_new_protected:Npn \__draw_path_rectangle_rounded:nnnn #1#2#3#4
577   {
578     \draw_path_moveto:n { #1 + #3 , #2 + #4 }
579     \draw_path_lineto:n { #1 , #2 + #4 }
580     \draw_path_lineto:n { #1 , #2 }
581     \draw_path_lineto:n { #1 + #3 , #2 }
582     \draw_path_close:
583     \draw_path_moveto:n { #1 , #2 }
584   }

```

(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
585 \cs_new_protected:Npn \draw_path_rectangle_corners:nn #1#2
586 {
587   \_draw_point_process:nnn
588   { \_draw_path_rectangle_corners:nnnnn {#1} }
589   {#1} {#2}
590 }
591 \cs_new_protected:Npn \_draw_path_rectangle_corners:nnnnn #1#2#3#4#5
592 { \draw_path_rectangle:nn {#1} { #4 - #2 , #5 - #3 } }

```

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

4.7 Grids

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

```

\_draw_path_grid_auxi:nnnnnn
\_draw_path_grid_auxi:ffnnnn
\_draw_path_grid_auxii:nnnnnn
\_draw_path_grid_auxiii:nnnnnn
\_draw_path_grid_auxiiii:ffnnnn
\_draw_path_grid_auxiv:nnnnnnnn
\_draw_path_grid_auxiv:ffnnnnnn
593 \cs_new_protected:Npn \draw_path_grid:nnnn #1#2#3#4
594 {
595   \_draw_point_process:nnn
596   {
597     \_draw_path_grid_auxi:ffnnnn
598     { \dim_eval:n { \dim_abs:n {#1} } }
599     { \dim_eval:n { \dim_abs:n {#2} } }
600   }
601   {#3} {#4}
602 }
603 \cs_new_protected:Npn \_draw_path_grid_auxi:nnnnnn #1#2#3#4#5#6
604 {
605   \dim_compare:nNnTF {#3} > {#5}
606   { \_draw_path_grid_auxii:nnnnnn {#1} {#2} {#5} {#4} {#3} {#6} }
607   { \_draw_path_grid_auxii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6} }
608 }
609 \cs_generate_variant:Nn \_draw_path_grid_auxi:nnnnnn { ff }
610 \cs_new_protected:Npn \_draw_path_grid_auxii:nnnnnn #1#2#3#4#5#6
611 {
612   \dim_compare:nNnTF {#4} > {#6}
613   { \_draw_path_grid_auxiii:nnnnnn {#1} {#2} {#3} {#6} {#5} {#4} }
614   { \_draw_path_grid_auxiii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6} }
615 }
616 \cs_new_protected:Npn \_draw_path_grid_auxiii:nnnnnn #1#2#3#4#5#6
617 {
618   \_draw_path_grid_auxiv:ffnnnnnn
619   { \fp_to_dim:n { #1 * trunc(#3/(#1)) } }
620   { \fp_to_dim:n { #2 * trunc(#4/(#2)) } }
621   {#1} {#2} {#3} {#4} {#5} {#6}
622 }
623 \cs_new_protected:Npn \_draw_path_grid_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
624 {
625   \dim_step_inline:nnnn
626   {#1}

```

```

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

```

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

4.8 Using paths

Actions to pass to the driver.

```

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

```

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

Actions handled at the macro layer.

```

\l__draw_path_use_bb_bool
\l__draw_path_use_clear_bool
646 \bool_new:N \l__draw_path_use_bb_bool
647 \bool_new:N \l__draw_path_use_clear_bool

```

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

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

```

\draw_path_use:n
\draw_path_use_clear:n
\__draw_path_use:n
\__draw_path_use_action_draw:
\__draw_path_use_action_fillstroke:
\__draw_path_use_stroke_bb:
\__draw_path_use_stroke_bb_aux:NnN
648 \cs_new_protected:Npn \draw_path_use:n #1
649 {
650     \tl_if_blank:nF {#1}
651     { \__draw_path_use:n {#1} }
652 }
653 \cs_new_protected:Npn \draw_path_use_clear:n #1
654 {
655     \bool_lazy_or:nnTF
656     { \tl_if_blank_p:n {#1} }
657     { \str_if_eq_p:nn {#1} { clear } }
658     {
659         \__draw_softpath_clear:
660         \__draw_path_reset_limits:
661     }
662     { \__draw_path_use:n { #1 , clear } }
663 }

```

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).

```

664 \cs_new_protected:Npn \__draw_path_use:n #1
665 {
666   \bool_set_false:N \l__draw_path_use_clip_bool
667   \bool_set_false:N \l__draw_path_use_fill_bool
668   \bool_set_false:N \l__draw_path_use_stroke_bool
669   \clist_map_inline:nn {#1}
670   {
671     \cs_if_exist:cTF { l__draw_path_use_ ##1 _ bool }
672     { \bool_set_true:c { l__draw_path_use_ ##1 _ bool } }
673     {
674       \cs_if_exist_use:cF { __draw_path_use_action_ ##1 : }
675       { \msg_error:nnn { draw } { invalid-path-action } {##1} }
676     }
677   }
678   \__draw_softpath_round_corners:
679   \bool_lazy_and:nnT
680   { \l_draw_bb_update_bool }
681   { \l__draw_path_use_stroke_bool }
682   { \__draw_path_use_stroke_bb: }
683   \__draw_softpath_use:
684   \bool_if:NT \l__draw_path_use_clip_bool
685   {
686     \__draw_backend_clip:
687     \bool_set_false:N \l_draw_bb_update_bool
688     \bool_lazy_or:nnF
689     { \l__draw_path_use_fill_bool }
690     { \l__draw_path_use_stroke_bool }
691     { \__draw_backend_discardpath: }
692   }
693   \bool_lazy_or:nnT
694   { \l__draw_path_use_fill_bool }
695   { \l__draw_path_use_stroke_bool }
696   {
697     \use:c
698     {
699       __draw_backend_
700       \bool_if:NT \l__draw_path_use_fill_bool { fill }
701       \bool_if:NT \l__draw_path_use_stroke_bool { stroke }
702       :
703     }
704   }
705   \bool_if:NT \l__draw_path_use_clear_bool
706   { \__draw_softpath_clear: }
707 }
708 \cs_new_protected:Npn \__draw_path_use_action_draw:
709 {
710   \bool_set_true:N \l__draw_path_use_stroke_bool
711 }
712 \cs_new_protected:Npn \__draw_path_use_action_fillstroke:
713 {
714   \bool_set_true:N \l__draw_path_use_fill_bool

```

```

715 \bool_set_true:N \l__draw_path_use_stroke_bool
716 }

```

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.

```

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

```

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

```

748 \cs_new_protected:Npn \draw_path_scope_begin:
749 {
750   \group_begin:
751   \dim_set_eq:NN \l__draw_path_lastx_dim \g__draw_path_lastx_dim
752   \dim_set_eq:NN \l__draw_path_lasty_dim \g__draw_path_lasty_dim

```

```

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

```

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

```

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

```

5 l3draw-points implementation

```

790 <*package>
791 <@@=draw>

```

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

- `\pgfpointorigin`: Can be given explicitly as `0pt,0pt`.
- `\pgfpointadd`, `\pgfpointdiff`, `\pgfpointscale`: Can be given explicitly.

- `\pgfextractx`, `\pgfextracty`: Available by applying `\use_i:nn/\use_ii:nn` or similar to the `x`-type expansion of a point expression.
- `\pgfgetlastxy`: Unused in the entire `pgf` core, may be emulated by `x`-type expansion of a point expression, then using the result.

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

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

5.1 Support functions

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

```

\__draw_point_process:nn
  \__draw_point_process_auxi:nn
  \__draw_point_process_auxii:nw
\__draw_point_process:nnn
  \__draw_point_process_auxiii:nnn
  \__draw_point_process_auxiv:nw
\__draw_point_process:nnnn
  \__draw_point_process_auxv:nnnn
  \__draw_point_process_auxvi:nw
\__draw_point_process:nnnnn
  \__draw_point_process_auxvii:nnnnn
  \__draw_point_process_auxviii:nw
792 \cs_new:Npn \__draw_point_process:nn #1#2
793 {
794   \exp_args:Nf \__draw_point_process_auxi:nn
795   { \draw_point:n {#2} }
796   {#1}
797 }
798 \cs_new:Npn \__draw_point_process_auxi:nn #1#2
799 { \__draw_point_process_auxii:nw {#2} #1 \s__draw_stop }
800 \cs_new:Npn \__draw_point_process_auxii:nw #1 #2 , #3 \s__draw_stop
801 { #1 {#2} {#3} }
802 \cs_new:Npn \__draw_point_process:nnn #1#2#3
803 {
804   \exp_args:Nff \__draw_point_process_auxiii:nnn
805   { \draw_point:n {#2} }
806   { \draw_point:n {#3} }
807   {#1}
808 }
809 \cs_new:Npn \__draw_point_process_auxiii:nnn #1#2#3
810 { \__draw_point_process_auxiv:nw {#3} #1 \s__draw_mark #2 \s__draw_stop }
811 \cs_new:Npn \__draw_point_process_auxiv:nw #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_stop
812 { #1 {#2} {#3} {#4} {#5} }
813 \cs_new:Npn \__draw_point_process:nnnn #1#2#3#4
814 {
815   \exp_args:Nfff \__draw_point_process_auxv:nnnn
816   { \draw_point:n {#2} }
817   { \draw_point:n {#3} }
818   { \draw_point:n {#4} }
819   {#1}
820 }

```

```

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

```

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

5.2 Basic points

<pre> \draw_point:n __draw_point_to_dim:n __draw_point_to_dim:f __draw_point_to_dim:w </pre>	<p>Co-ordinates are always returned as two dimensions.</p> <pre> 843 \cs_new:Npn \draw_point:n #1 844 { __draw_point_to_dim:f { \fp_eval:n {#1} } } 845 \cs_new:Npn __draw_point_to_dim:n #1 846 { __draw_point_to_dim:w #1 } 847 \cs_generate_variant:Nn __draw_point_to_dim:n { f } 848 \cs_new:Npn __draw_point_to_dim:w (#1 , ~ #2) { #1pt , #2pt } </pre>
---	---

5.3 Polar co-ordinates

<pre> \draw_point_polar:nn \draw_point_polar:nnn __draw_draw_polar:nnn __draw_draw_polar:fnn </pre>	<p>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.</p> <pre> 849 \cs_new:Npn \draw_point_polar:nn #1#2 850 { \draw_point_polar:nnn {#1} {#1} {#2} } 851 \cs_new:Npn \draw_point_polar:nnn #1#2#3 852 { __draw_draw_polar:fnn { \fp_eval:n {#3} } {#1} {#2} } 853 \cs_new:Npn __draw_draw_polar:nnn #1#2#3 854 { \draw_point:n { cosd(#1) * (#2) , sind(#1) * (#3) } } 855 \cs_generate_variant:Nn __draw_draw_polar:nnn { f } </pre>
---	---

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.

```

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

```

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

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

and

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

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

```

873 \cs_new:Npn \draw_point_intersect_lines:nnnn #1#2#3#4
874 {
875   \__draw_point_process:nnnnn
876   { \__draw_point_intersect_lines:nnnnnnnn {#1} {#2} {#3} {#4}
877     {#1} {#2} {#3} {#4}
878 }

```

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

```

#1 x1
#2 y1
#3 x2
#4 y2

```

#5 x_3

#6 y_3

#7 x_4

#8 y_4

so now just have to do all of the calculation.

```

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

```

```

\draw_point_intersect_circles:nnnnn
__draw_point_intersect_circles_auxi:nnnnnnnn
__draw_point_intersect_circles_auxii:nnnnnnnn
__draw_point_intersect_circles_auxiii:ffnnnnnn
draw_point_intersect_circles_auxiii:ffnnnnnn
draw_point_intersect_circles_auxiv:nnnnnnnnnn
draw_point_intersect_circles_auxiv:fnnnnnnnnn
draw_point_intersect_circles_auxv:nnnnnnnnnnnn
draw_point_intersect_circles_auxv:ffnnnnnnnnnn
draw_point_intersect_circles_auxvi:nnnnnnnnnnnn
draw_point_intersect_circles_auxvi:fnnnnnnnnnnn
draw_point_intersect_circles_auxvii:nnnnnnnnnn
draw_point_intersect_circles_auxvii:ffnnnnnnnn

```

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

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

in either

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

or

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

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

```

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

```

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.

```

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

```

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

```

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

```

```

930 }
931 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxiv:nnnnnnnn { f }
932 \cs_new:Npn \__draw_point_intersect_circles_auxv:nnnnnnnn #1#2#3#4#5#6#7#8#9
933 {
934   \__draw_point_intersect_circles_auxvi:fnnnnnnnn
935   { \fp_eval:n { 0.5 * #1 * ( #2 + #3 * #3 - #6 * #6 ) } }
936   {#1} {#2} {#4} {#5} {#7} {#8} {#9}
937 }
938 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxv:nnnnnnnn { ff }

```

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

```

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

```

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

```

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

```

The intersection points P_1 and P_2 between a line joining points (x_1, y_1) and (x_2, y_2) and

```

\draw_point_intersect_line_circle:nnnnn
w_point_intersect_line_circle_auxi:nnnnnnnn
_point_intersect_line_circle_auxii:nnnnnnnn
_point_intersect_line_circle_auxiii:fnnnnnnnn
_point_intersect_line_circle_auxiiii:nnnnnnnn
_point_intersect_line_circle_auxiii:ffnnnnnnnn
_point_intersect_line_circle_auxiv:nnnnnnnn
_point_intersect_line_circle_auxiv:ffnnnnnnnn
draw_point_intersect_line_circle_auxv:nnnnnn
draw_point_intersect_line_circle_auxv:fnnnnn

```

a circle with center (x_3, y_3) and radius r . We use the intermediate values

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

in either

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

or

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

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

```

954 \cs_new:Npn \draw_point_intersect_line_circle:nnnnn #1#2#3#4#5
955 {
956   \__draw_point_process:nnnn
957   { \__draw_point_intersect_line_circle_auxi:nnnnnnnn {#4} {#5} }
958   {#1} {#2} {#3}
959 }
960 \cs_new:Npn \__draw_point_intersect_line_circle_auxi:nnnnnnnn #1#2#3#4#5#6#7#8
961 {
962   \__draw_point_intersect_line_circle_auxii:fnnnnnnnn
963   { \fp_eval:n {#1} } {#3} {#4} {#5} {#6} {#7} {#8} {#2}
964 }

```

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

```

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

```

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

```

965 \cs_new:Npn \__draw_point_intersect_line_circle_auxii:nnnnnnnn #1#2#3#4#5#6#7#8
966 {
967   \__draw_point_intersect_line_circle_auxiii:ffnnnnnn
968   { \fp_eval:n { (#4-#2)*(#4-#2)+(#5-#3)*(#5-#3) } }
969   { \fp_eval:n { 2*((#4-#2)*(#2-#6)+(#5-#3)*(#3-#7)) } }
970   { \fp_eval:n { (#6*#6+#7*#7)+(#2*#2+#3*#3)-(2*(#6*#2+#7*#3))-(#1*#1) } }
971   {#2} {#3} {#4} {#5} {#6} {#7} {#8}
972 }
973 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxii:nnnnnnnn { f }

```

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

```

974 \cs_new:Npn \__draw_point_intersect_line_circle_auxiii:nnnnnnnn #1#2#3#4#5#6#7#8
975 {
976   \__draw_point_intersect_line_circle_auxiv:ffnnnnnn
977   { \fp_eval:n { #2 * #2 - 4 * #1 * #3 } }
978   { \int_if_odd:nTF {#8} { 1 } { -1 } }
979   {#1} {#2} {#4} {#5} {#6} {#7}
980 }
981 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxiii:nnnnnnnn { fff }

```

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.

```

982 \cs_new:Npn \__draw_point_intersect_line_circle_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
983 {
984   \__draw_point_intersect_line_circle_auxv:fnnnn
985   { \fp_eval:n { (-1 * #4 + #2 * sqrt(#1)) / (2 * #3) } }
986   {#5} {#6} {#7} {#8}
987 }
988 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxiv:nnnnnnnn { ff }
989 \cs_new:Npn \__draw_point_intersect_line_circle_auxv:nnnnn #1#2#3#4#5
990 {
991   \draw_point:n
992   { #2 + #1 * (#4 - #2), #3 + #1 * (#5 - #3) }
993 }
994 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxv:nnnnn { f }

```

5.6 Interpolation on a line (vector) or arc

Simple maths after expansion.

```

\draw_point_interpolate_line:nnn 995 \cs_new:Npn \draw_point_interpolate_line:nnn #1#2#3
\_draw_point_interpolate_line_aux:nnnnn 996 {
\_draw_point_interpolate_line_aux:fnnnn 997 \_draw_point_process:nnn
\_draw_point_interpolate_line_aux:nnnnnn 998 { \_draw_point_interpolate_line_aux:fnnnn { \fp_eval:n {#1} } }
\_draw_point_interpolate_line_aux:fnnnnnn 999 {#2} {#3}
1000 }
1001 \cs_new:Npn \_draw_point_interpolate_line_aux:nnnnn #1#2#3#4#5
1002 {
1003 \_draw_point_interpolate_line_aux:fnnnnn { \fp_eval:n { 1 - #1 } }
1004 {#1} {#2} {#3} {#4} {#5}
1005 }
1006 \cs_generate_variant:Nn \_draw_point_interpolate_line_aux:nnnnn { f }
1007 \cs_new:Npn \_draw_point_interpolate_line_aux:nnnnnn #1#2#3#4#5#6
1008 { \draw_point:n { #2 * #3 + #1 * #5 , #2 * #4 + #1 * #6 } }
1009 \cs_generate_variant:Nn \_draw_point_interpolate_line_aux:nnnnnn { f }

```

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

```

\draw_point_interpolate_distance:nnn 1010 \cs_new:Npn \draw_point_interpolate_distance:nnn #1#2#3
\_draw_point_interpolate_distance:nnnnn 1011 {
\_draw_point_interpolate_distance:nnnnnn 1012 \_draw_point_process:nn
\_draw_point_interpolate_distance:fnnnnnn 1013 { \_draw_point_interpolate_distance:nnnn {#1} {#3} }
1014 {#2}
1015 }
1016 \cs_new:Npn \_draw_point_interpolate_distance:nnnn #1#2#3#4
1017 {
1018 \_draw_point_process:nn
1019 {
1020 \_draw_point_interpolate_distance:fnnnnn
1021 { \fp_eval:n {#1} } {#3} {#4}
1022 }
1023 { \draw_point_unit_vector:n { ( #2 ) - ( #3 , #4 ) } }
1024 }
1025 \cs_new:Npn \_draw_point_interpolate_distance:nnnnn #1#2#3#4#5
1026 { \draw_point:n { #2 + #1 * #4 , #3 + #1 * #5 } }
1027 \cs_generate_variant:Nn \_draw_point_interpolate_distance:nnnnn { f }

```

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

```

\draw_point_interpolate_arcaxes:nnnnnn 2028 \cs_new:Npn \draw_point_interpolate_arcaxes:nnnnnn #1#2#3#4#5#6
draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn 2029 {
draw_point_interpolate_arcaxes_auxii:nnnnnnnnnn 2030 \_draw_point_process:nnnn
draw_point_interpolate_arcaxes_auxiii:nnnnnnnnnn 2031 { \_draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn {#1} {#5} {#6} }
draw_point_interpolate_arcaxes_auxiiii:nnnnnnnnnn 2032 {#2} {#3} {#4}
draw_point_interpolate_arcaxes_auxiv:nnnnnnnnnn 2033 }
draw_point_interpolate_arcaxes_auxiv:fnnnnnnnnnn 2034 \cs_new:Npn \_draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
2035 {
2036 \_draw_point_interpolate_arcaxes_auxii:fnnnnnnnnnn

```

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.

```

1037     { \fp_eval:n {#1} } {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1038   }

```

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

```

#1 p
#2  $\theta_1$ 
#3  $\theta_2$ 
#4  $x_c$ 
#5  $y_c$ 
#6  $x_{a1}$ 
#7  $y_{a1}$ 
#8  $x_{a2}$ 
#9  $y_{a2}$ 

```

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

```

1039 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnn #1#2#3#4#5#6#7#8#9
1040   {
1041     \__draw_point_interpolate_arcaxes_auxiii:fnnnnnnn
1042     { \fp_eval:n { #1 * (#3) + ( 1 - #1 ) * (#2) } }
1043     {#4} {#5} {#6} {#7} {#8} {#9}
1044   }
1045 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnn { f }
1046 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnn #1#2#3#4#5#6#7
1047   {
1048     \__draw_point_interpolate_arcaxes_auxiv:ffnnnnnnn
1049     { \fp_eval:n { cosd (#1) } }
1050     { \fp_eval:n { sind (#1) } }
1051     {#2} {#3} {#4} {#5} {#6} {#7}
1052   }
1053 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnn { f }
1054 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnnn #1#2#3#4#5#6#7#8
1055   {
1056     \draw_point:n
1057     { #3 + #1 * #5 + #2 * #7 , #4 + #1 * #6 + #2 * #8 }
1058   }
1059 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnnn { ff }

```

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

```

\draw_point_interpolate_curve:nnnnn
\draw_point_interpolate_curve_auxi:nnnnnnnnn
\draw_point_interpolate_curve_auxii:nnnnnnnnn
\draw_point_interpolate_curve_auxiii:fnnnnnnnnn
\draw_point_interpolate_curve_auxiiii:nnnnnnn
\draw_point_interpolate_curve_auxv:ffnnnnnnn
\draw_point_interpolate_curve_auxvi:ffnnnnnnn
\draw_point_interpolate_curve_auxvii:ffnnnnnnn
\draw_point_interpolate_curve_auxviii:ffnnnnnnn
\draw_point_interpolate_curve_auxviiii:ffnnnnnnn

```

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.

```

1060 \cs_new:Npn \draw_point_interpolate_curve:nnnnnn #1#2#3#4#5
1061 {
1062   \__draw_point_process:nnnnn
1063   { \__draw_point_interpolate_curve_auxi:nnnnnnnnn {#1} }
1064   {#2} {#3} {#4} {#5}
1065 }
1066 \cs_new:Npn \__draw_point_interpolate_curve_auxi:nnnnnnnnn #1#2#3#4#5#6#7#8#9
1067 {
1068   \__draw_point_interpolate_curve_auxii:fnnnnnnnn
1069   { \fp_eval:n {#1} }
1070   {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1071 }

```

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!

```

1072 \cs_new:Npn \__draw_point_interpolate_curve_auxii:nnnnnnnnn
1073   #1#2#3#4#5#6#7#8#9
1074 {
1075   \__draw_point_interpolate_curve_auxiii:fnnnnnn
1076   { \fp_eval:n { 1 - #1 } }
1077   {#1}
1078   { {#2} {#3} } { {#4} {#5} } { {#6} {#7} } { {#8} {#9} }
1079 }
1080 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxii:nnnnnnnnn { f }
1081 % \begin{macrocode}
1082 % We need to do the first cycle, but haven't got enough arguments to keep

```

```

1083 % everything in play at once. So her ewe use a but of argument re-ordering
1084 % and a single auxiliary to get the job done.
1085 % \begin{macrocode}
1086 \cs_new:Npn \__draw_point_interpolate_curve_auxiii:nnnnnn #1#2#3#4#5#6
1087 {
1088   \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #3 #4
1089   \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #4 #5
1090   \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #5 #6
1091   \prg_do_nothing:
1092   \__draw_point_interpolate_curve_auxvi:n { {#1} {#2} }
1093 }
1094 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxiii:nnnnnn { f }
1095 \cs_new:Npn \__draw_point_interpolate_curve_auxiv:nnnnnn #1#2#3#4#5#6
1096 {
1097   \__draw_point_interpolate_curve_auxv:ffw
1098   { \fp_eval:n { #1 * #3 + #2 * #5 } }
1099   { \fp_eval:n { #1 * #4 + #2 * #6 } }
1100 }
1101 \cs_new:Npn \__draw_point_interpolate_curve_auxv:nnw
1102 #1#2#3 \prg_do_nothing: #4#5
1103 {
1104   #3
1105   \prg_do_nothing:
1106   #4 { #5 {#1} {#2} }
1107 }
1108 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxv:nnw { ff }
1109 % \begin{macrocode}
1110 % Get the arguments back into the right places and to the second and
1111 % third cycles directly.
1112 % \begin{macrocode}
1113 \cs_new:Npn \__draw_point_interpolate_curve_auxvi:n #1
1114 { \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1 }
1115 \cs_new:Npn \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1#2#3#4#5#6#7#8
1116 {
1117   \__draw_point_interpolate_curve_auxviii:ffffnn
1118   { \fp_eval:n { #1 * #5 + #2 * #3 } }
1119   { \fp_eval:n { #1 * #6 + #2 * #4 } }
1120   { \fp_eval:n { #1 * #7 + #2 * #5 } }
1121   { \fp_eval:n { #1 * #8 + #2 * #6 } }
1122   {#1} {#2}
1123 }
1124 \cs_new:Npn \__draw_point_interpolate_curve_auxviii:nnnnnn #1#2#3#4#5#6
1125 {
1126   \draw_point:n
1127   { #5 * #3 + #6 * #1 , #5 * #4 + #6 * #2 }
1128 }
1129 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxviii:nnnnnn { ffff }

```

(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 1130 \dim_new:N \l__draw_xvec_x_dim
\l__draw_yvec_x_dim 1131 \dim_new:N \l__draw_xvec_y_dim
\l__draw_yvec_y_dim 1132 \dim_new:N \l__draw_yvec_x_dim
\l__draw_zvec_x_dim 1133 \dim_new:N \l__draw_yvec_y_dim
\l__draw_zvec_y_dim 1134 \dim_new:N \l__draw_zvec_x_dim
1135 \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 1136 \cs_new_protected:Npn \draw_xvec:n #1
\draw_zvec:n 1137 { \__draw_vec:nn { x } {#1} }
\__draw_vec:nn 1138 \cs_new_protected:Npn \draw_yvec:n #1
\__draw_vec:nnn 1139 { \__draw_vec:nn { y } {#1} }
1140 \cs_new_protected:Npn \draw_zvec:n #1
1141 { \__draw_vec:nn { z } {#1} }
1142 \cs_new_protected:Npn \__draw_vec:nn #1#2
1143 {
1144   \__draw_point_process:nn { \__draw_vec:nnn {#1} } {#2}
1145 }
1146 \cs_new_protected:Npn \__draw_vec:nnn #1#2#3
1147 {
1148   \dim_set:cn { l__draw_ #1 vec_x_dim } {#2}
1149   \dim_set:cn { l__draw_ #1 vec_y_dim } {#3}
1150 }

```

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

Initialise the vectors.

```

1151 \draw_xvec:n { 1cm , 0cm }
1152 \draw_yvec:n { 0cm , 1cm }
1153 \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 1154 \cs_new:Npn \draw_point_vec:nn #1#2
\__draw_point_vec:ff 1155 { \__draw_point_vec:ff { \fp_eval:n {#1} } { \fp_eval:n {#2} } }
\draw_point_vec:nnn 1156 \cs_new:Npn \__draw_point_vec:nn #1#2
\__draw_point_vec:nnn 1157 {
\__draw_point_vec:fff 1158   \draw_point:n
1159   {
1160     #1 * \l__draw_xvec_x_dim + #2 * \l__draw_yvec_x_dim ,
1161     #1 * \l__draw_xvec_y_dim + #2 * \l__draw_yvec_y_dim
1162   }
1163 }
1164 \cs_generate_variant:Nn \__draw_point_vec:nn { ff }
1165 \cs_new:Npn \draw_point_vec:nnn #1#2#3
1166 {
1167   \__draw_point_vec:fff
1168   { \fp_eval:n {#1} } { \fp_eval:n {#2} } { \fp_eval:n {#3} }
1169 }
1170 \cs_new:Npn \__draw_point_vec:nnn #1#2#3
1171 {
1172   \draw_point:n
1173   {

```

```

1174         #1 * \l__draw_xvec_x_dim
1175     + #2 * \l__draw_yvec_x_dim
1176     + #3 * \l__draw_zvec_x_dim
1177     ,
1178     #1 * \l__draw_xvec_y_dim
1179     + #2 * \l__draw_yvec_y_dim
1180     + #3 * \l__draw_zvec_y_dim
1181 }
1182 }
1183 \cs_generate_variant:Nn \__draw_point_vec:nnn { fff }

```

(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
\__draw_point_vec_polar:nnn
\__draw_point_vec_polar:fnn
1184 \cs_new:Npn \draw_point_vec_polar:nn #1#2
1185 { \draw_point_vec_polar:nnn {#1} {#1} {#2} }
1186 \cs_new:Npn \draw_point_vec_polar:nnn #1#2#3
1187 { \__draw_draw_vec_polar:fnn { \fp_eval:n {#3} } {#1} {#2} }
1188 \cs_new:Npn \__draw_draw_vec_polar:nnn #1#2#3
1189 {
1190     \draw_point:n
1191     {
1192         cosd(#1) * (#2) * \l__draw_xvec_x_dim ,
1193         sind(#1) * (#3) * \l__draw_yvec_y_dim
1194     }
1195 }
1196 \cs_generate_variant:Nn \__draw_draw_vec_polar:nnn { f }

```

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

```

1197 \cs_new:Npn \draw_point_transform:n #1
1198 {
1199     \__draw_point_process:nn
1200     { \__draw_point_transform:nn } {#1}
1201 }
1202 \cs_new:Npn \__draw_point_transform:nn #1#2
1203 {
1204     \bool_if:NTF \l__draw_matrix_active_bool
1205     {
1206         \draw_point:n
1207         {
1208             (
1209                 \l__draw_matrix_a_fp * #1
1210                 + \l__draw_matrix_c_fp * #2
1211                 + \l__draw_xshift_dim
1212             )
1213             ,
1214             (
1215                 \l__draw_matrix_b_fp * #1

```

```

1216         + \l__draw_matrix_d_fp * #2
1217         + \l__draw_yshift_dim
1218     )
1219 }
1220 }
1221 {
1222     \draw_point:n
1223     {
1224         (#1, #2)
1225         + ( \l__draw_xshift_dim , \l__draw_yshift_dim )
1226     }
1227 }
1228 }

```

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

`__draw_point_transform_noshift:n`
`__draw_point_transform_noshift:nn`

A version with no shift: used for internal purposes.

```

1229 \cs_new:Npn \__draw_point_transform_noshift:n #1
1230 {
1231     \__draw_point_process:nn
1232     { \__draw_point_transform_noshift:nn } {#1}
1233 }
1234 \cs_new:Npn \__draw_point_transform_noshift:nn #1#2
1235 {
1236     \bool_if:NTF \l__draw_matrix_active_bool
1237     {
1238         \draw_point:n
1239         {
1240             (
1241                 \l__draw_matrix_a_fp * #1
1242                 + \l__draw_matrix_c_fp * #2
1243             )
1244             ,
1245             (
1246                 \l__draw_matrix_b_fp * #1
1247                 + \l__draw_matrix_d_fp * #2
1248             )
1249         }
1250     }
1251     { \draw_point:n { (#1, #2) } }
1252 }

```

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

```

1253 \end{package}

```

6 l3draw-scopes implementation

```

1254 \*package
1255 \@@=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

<code>\g__draw_xmax_dim</code>	Used to track the overall (official) size of the image created: may not actually be the
<code>\g__draw_xmin_dim</code>	natural size of the content.
<code>\g__draw_ymax_dim</code>	1256 <code>\dim_new:N \g__draw_xmax_dim</code>
<code>\g__draw_ymin_dim</code>	1257 <code>\dim_new:N \g__draw_xmin_dim</code>
	1258 <code>\dim_new:N \g__draw_ymax_dim</code>
	1259 <code>\dim_new:N \g__draw_ymin_dim</code>
	<i>(End of definition for <code>\g__draw_xmax_dim</code> and others.)</i>
<code>\l_draw_bb_update_bool</code>	Flag to indicate that a path (or similar) should update the bounding box of the drawing.
	1260 <code>\bool_new:N \l_draw_bb_update_bool</code>
	<i>(End of definition for <code>\l_draw_bb_update_bool</code>. This variable is documented on page ??.)</i>
<code>\l__draw_layer_main_box</code>	Box for setting the drawing itself and the top-level layer.
	1261 <code>\box_new:N \l__draw_main_box</code>
	1262 <code>\box_new:N \l__draw_layer_main_box</code>
	<i>(End of definition for <code>\l__draw_layer_main_box</code>.)</i>
<code>\g__draw_id_int</code>	The drawing number.
	1263 <code>\int_new:N \g__draw_id_int</code>
	<i>(End of definition for <code>\g__draw_id_int</code>.)</i>
<code>__draw_reset_bb:</code>	A simple auxiliary.
	1264 <code>\cs_new_protected:Npn __draw_reset_bb:</code>
	1265 <code>{</code>
	1266 <code>\dim_gset:Nn \g__draw_xmax_dim { -\c_max_dim }</code>
	1267 <code>\dim_gset:Nn \g__draw_xmin_dim { \c_max_dim }</code>
	1268 <code>\dim_gset:Nn \g__draw_ymax_dim { -\c_max_dim }</code>
	1269 <code>\dim_gset:Nn \g__draw_ymin_dim { \c_max_dim }</code>
	1270 <code>}</code>
	<i>(End of definition for <code>__draw_reset_bb:</code>.)</i>
<code>\draw_begin:</code>	Drawings are created by setting them into a box, then adjusting the box before inserting
<code>\draw_end:</code>	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 <code>hbox</code> . To allow for layers,
	there is some box nesting: notice that we
	1271 <code>\cs_new_protected:Npn \draw_begin:</code>
	1272 <code>{</code>
	1273 <code>\group_begin:</code>
	1274 <code>\int_gincr:N \g__draw_id_int</code>
	1275 <code>\hbox_set:Nw \l__draw_main_box</code>
	1276 <code>__draw_backend_begin:</code>
	1277 <code>__draw_reset_bb:</code>

```

1278     \__draw_path_reset_limits:
1279     \bool_set_true:N \l_draw_bb_update_bool
1280     \draw_transform_matrix_reset:
1281     \draw_transform_shift_reset:
1282     \__draw_softpath_clear:
1283     \draw_linewidth:n { \l_draw_default_linewidth_dim }
1284     \color_select:n { . }
1285     \draw_nonzero_rule:
1286     \draw_cap_but:
1287     \draw_join_miter:
1288     \draw_miterlimit:n { 10 }
1289     \draw_dash_pattern:nn { } { 0cm }
1290     \hbox_set:Nw \l__draw_layer_main_box
1291   }
1292 \cs_new_protected:Npn \draw_end:
1293 {
1294     \__draw_baseline_finalise:w
1295     \exp_args:NNNV \hbox_set_end:
1296     \clist_set:Nn \l_draw_layers_clist \l_draw_layers_clist
1297     \__draw_layers_insert:
1298     \__draw_backend_end:
1299     \hbox_set_end:
1300     \dim_compare:nNnT \g__draw_xmin_dim = \c_max_dim
1301     {
1302         \dim_gzero:N \g__draw_xmax_dim
1303         \dim_gzero:N \g__draw_xmin_dim
1304         \dim_gzero:N \g__draw_ymax_dim
1305         \dim_gzero:N \g__draw_ymin_dim
1306     }
1307     \__draw_finalise:
1308     \box_set_wd:Nn \l__draw_main_box
1309     { \g__draw_xmax_dim - \g__draw_xmin_dim }
1310     \mode_leave_vertical:
1311     \box_use_drop:N \l__draw_main_box
1312 \group_end:
1313 }

```

(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 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.

```

1314 \cs_new_protected:Npn \__draw_finalise:
1315 {
1316     \hbox_set:Nn \l__draw_main_box
1317     {
1318         \skip_horizontal:n { -\g__draw_xmin_dim }
1319         \box_move_down:nn
1320         { \g__draw_ymin_dim }
1321         { \box_use_drop:N \l__draw_main_box }
1322     }
1323     \box_set_dp:Nn \l__draw_main_box { 0pt }

```

```

1324     \box_set_ht:Nn \l__draw_main_box
1325     { \g__draw_ymax_dim - \g__draw_ymin_dim }
1326   }
1327   \cs_new_protected:Npn \__draw_finalise_baseline:n #1
1328   {
1329     \hbox_set:Nn \l__draw_main_box
1330     {
1331       \skip_horizontal:n { -\g__draw_xmin_dim }
1332       \box_move_down:nn
1333       { #1 }
1334       { \box_use_drop:N \l__draw_main_box }
1335     }
1336     \box_set_dp:Nn \l__draw_main_box
1337     {
1338       \dim_max:nn
1339       { #1 - \g__draw_ymin_dim }
1340       { 0pt }
1341     }
1342     \box_set_ht:Nn \l__draw_main_box
1343     { \g__draw_ymax_dim + #1 }
1344   }

```

(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

```

1345 \bool_new:N \l__draw_baseline_bool
1346 \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.

```

1347 \cs_new_protected:Npn \draw_baseline:n #1
1348   {
1349     \bool_set_true:N \l__draw_baseline_bool
1350     \dim_set:Nn \l__draw_baseline_dim { \fp_to_dim:n { #1 } }
1351   }

```

(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.

```

1352 \cs_new_protected:Npn \__draw_baseline_finalise:w #1 \__draw_finalise:
1353   {
1354     \bool_if:NTF \l__draw_baseline_bool
1355     {
1356       \use:e
1357       {
1358         \exp_not:n { #1 }
1359         \__draw_finalise_baseline:n { \dim_use:N \l__draw_baseline_dim }
1360       }
1361     }
1362     { #1 \__draw_finalise: }
1363   }

```

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

6.3 Scopes

`\l__draw_linewidth_dim` Storage for local variables.

```
\l__draw_fill_color_tl 1364 \dim_new:N \l__draw_linewidth_dim
\l__draw_stroke_color_tl 1365 \tl_new:N \l__draw_fill_color_tl
1366 \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: 1367 \cs_new_protected:Npn \draw_scope_begin:
1368 {
1369   \__draw_backend_scope_begin:
1370   \group_begin:
1371     \dim_set_eq:NN \l__draw_linewidth_dim \g__draw_linewidth_dim
1372     \draw_path_scope_begin:
1373   }
1374   \cs_new_protected:Npn \draw_scope_end:
1375   {
1376     \draw_path_scope_end:
1377     \dim_gset_eq:NN \g__draw_linewidth_dim \l__draw_linewidth_dim
1378     \group_end:
1379     \__draw_backend_scope_end:
1380   }
```

(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 1381 \dim_new:N \l__draw_xmax_dim
\l__draw_ymax_dim 1382 \dim_new:N \l__draw_xmin_dim
\l__draw_ymin_dim 1383 \dim_new:N \l__draw_ymax_dim
1384 \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: 1385 \cs_new_protected:Npn \__draw_scope_bb_begin:
1386 {
1387   \group_begin:
1388     \dim_set_eq:NN \l__draw_xmax_dim \g__draw_xmax_dim
1389     \dim_set_eq:NN \l__draw_xmin_dim \g__draw_xmin_dim
1390     \dim_set_eq:NN \l__draw_ymax_dim \g__draw_ymax_dim
1391     \dim_set_eq:NN \l__draw_ymin_dim \g__draw_ymin_dim
1392     \__draw_reset_bb:
1393   }
1394   \cs_new_protected:Npn \__draw_scope_bb_end:
1395   {
1396     \dim_gset_eq:NN \g__draw_xmax_dim \l__draw_xmax_dim
1397     \dim_gset_eq:NN \g__draw_xmin_dim \l__draw_xmin_dim
1398     \dim_gset_eq:NN \g__draw_ymax_dim \l__draw_ymax_dim
1399     \dim_gset_eq:NN \g__draw_ymin_dim \l__draw_ymin_dim
1400     \group_end:
1401   }
```

(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: 1402 \cs_new_protected:Npn \draw_suspend_begin:
1403 {
1404   \__draw_scope_bb_begin:
1405   \draw_path_scope_begin:
1406   \draw_transform_matrix_reset:
1407   \draw_transform_shift_reset:
1408   \__draw_layers_save:
1409 }
1410 \cs_new_protected:Npn \draw_suspend_end:
1411 {
1412   \__draw_layers_restore:
1413   \draw_path_scope_end:
1414   \__draw_scope_bb_end:
1415 }
```

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

```
1416 \</package>
```

7 l3draw-softpath implementation

```
1417 \*package>
```

```
1418 \@@=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.

```
1419 \tl_new:N \g__draw_softpath_main_tl
```

(End of definition for `\g__draw_softpath_main_tl.`)

`\l__draw_softpath_internal_tl` The soft path itself.

```
1420 \tl_new:N \l__draw_softpath_internal_tl
```

(End of definition for `\l__draw_softpath_internal_tl.`)

`\g__draw_softpath_corners_bool` Allow for optimised path use.

```
1421 \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 1422 \cs_new_protected:Npn \__draw_softpath_add:n
\__draw_softpath_add:e 1423 { \tl_build_gput_right:Nn \g__draw_softpath_main_tl }
1424 \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: 1425 \cs_new_protected:Npn \__draw_softpath_clear:
1426 {
1427   \tl_build_get:NN \g__draw_softpath_main_tl \l__draw_softpath_internal_tl
1428   \l__draw_softpath_internal_tl
1429 }
1430 \cs_new_protected:Npn \__draw_softpath_clear:
1431 {
1432   \tl_build_gclear:N \g__draw_softpath_main_tl
1433   \bool_gset_false:N \g__draw_softpath_corners_bool
1434 }

```

(End of definition for `__draw_softpath_use:` and `__draw_softpath_clear:`.)

```

\g__draw_softpath_lastx_dim For tracking the end of the path (to close it).
\g__draw_softpath_lasty_dim 1435 \dim_new:N \g__draw_softpath_lastx_dim
1436 \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.
1437 \bool_new:N \g__draw_softpath_move_bool
1438 \bool_gset_true:N \g__draw_softpath_move_bool

```

(End of definition for `\g__draw_softpath_move_bool`.)

```

\__draw_softpath_curveto:nnnnnn The various parts of a path expressed as the appropriate soft path functions.
\__draw_softpath_lineto:nn 1439 \cs_new_protected:Npn \__draw_softpath_closepath:
\__draw_softpath_moveto:nn 1440 {
1441   \__draw_softpath_add:e
1442   {
1443     \__draw_softpath_close_op:nn
1444     { \dim_use:N \g__draw_softpath_lastx_dim }
1445     { \dim_use:N \g__draw_softpath_lasty_dim }
1446   }
1447 }
1448 \cs_new_protected:Npn \__draw_softpath_curveto:nnnnnn #1#2#3#4#5#6
1449 {
1450   \__draw_softpath_add:n
1451   {
1452     \__draw_softpath_curveto_opi:nn {#1} {#2}
1453     \__draw_softpath_curveto_opii:nn {#3} {#4}
1454     \__draw_softpath_curveto_opiii:nn {#5} {#6}
1455   }
1456 }
1457 \cs_new_protected:Npn \__draw_softpath_lineto:nn #1#2

```

```

1458 {
1459   \__draw_softpath_add:n
1460   { \__draw_softpath_lineto_op:nn {#1} {#2} }
1461 }
1462 \cs_new_protected:Npn \__draw_softpath_moveto:nn #1#2
1463 {
1464   \__draw_softpath_add:n
1465   { \__draw_softpath_moveto_op:nn {#1} {#2} }
1466   \bool_if:NT \g__draw_softpath_move_bool
1467   {
1468     \dim_gset:Nn \g__draw_softpath_lastx_dim {#1}
1469     \dim_gset:Nn \g__draw_softpath_lasty_dim {#2}
1470   }
1471 }
1472 \cs_new_protected:Npn \__draw_softpath_rectangle:nnnn #1#2#3#4
1473 {
1474   \__draw_softpath_add:n
1475   {
1476     \__draw_softpath_rectangle_opi:nn {#1} {#2}
1477     \__draw_softpath_rectangle_opii:nn {#3} {#4}
1478   }
1479 }
1480 \cs_new_protected:Npn \__draw_softpath_roundpoint:nn #1#2
1481 {
1482   \__draw_softpath_add:n
1483   { \__draw_softpath_roundpoint_op:nn {#1} {#2} }
1484   \bool_gset_true:N \g__draw_softpath_corners_bool
1485 }
1486 \cs_generate_variant:Nn \__draw_softpath_roundpoint:nn { VV }

```

(End of definition for __draw_softpath_curveto:nnnnnn and others.)

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

```

1487 \cs_new_protected:Npn \__draw_softpath_close_op:nn #1#2
1488 { \__draw_backend_closepath: }
1489 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nn #1#2
1490 { \__draw_softpath_curveto_opi:nnNnnNnn {#1} {#2} }
1491 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nnNnnNnn #1#2#3#4#5#6#7#8
1492 { \__draw_backend_curveto:nnnnnn {#1} {#2} {#4} {#5} {#7} {#8} }
1493 \cs_new_protected:Npn \__draw_softpath_curveto_opii:nn #1#2
1494 { \__draw_softpath_curveto_opii:nn }
1495 \cs_new_protected:Npn \__draw_softpath_curveto_opiii:nn #1#2
1496 { \__draw_softpath_curveto_opiii:nn }
1497 \cs_new_protected:Npn \__draw_softpath_lineto_op:nn #1#2
1498 { \__draw_backend_lineto:nn {#1} {#2} }
1499 \cs_new_protected:Npn \__draw_softpath_moveto_op:nn #1#2
1500 { \__draw_backend_moveto:nn {#1} {#2} }
1501 \cs_new_protected:Npn \__draw_softpath_roundpoint_op:nn #1#2 { }
1502 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nn #1#2
1503 { \__draw_softpath_rectangle_opi:nnNnn {#1} {#2} }
1504 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nnNnn #1#2#3#4#5
1505 { \__draw_backend_rectangle:nnnn {#1} {#2} {#4} {#5} }

```

```
1506 \cs_new_protected:Npn \__draw_softpath_rectangle_opii:nn #1#2 { }
```

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

7.2 Rounding soft path corners

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

\l__draw_softpath_main_tl For constructing the updated path.

```
1507 \tl_new:N \l__draw_softpath_main_tl
```

(End of definition for \l__draw_softpath_main_tl.)

\l__draw_softpath_part_tl Data structures.

```
1508 \tl_new:N \l__draw_softpath_part_tl
```

```
1509 \tl_new:N \l__draw_softpath_curve_end_tl
```

(End of definition for \l__draw_softpath_part_tl.)

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

```
1510 \fp_new:N \l__draw_softpath_lastx_fp
```

```
1511 \fp_new:N \l__draw_softpath_lasty_fp
```

```
1512 \dim_new:N \l__draw_softpath_corneri_dim
```

```
1513 \dim_new:N \l__draw_softpath_cornerii_dim
```

```
1514 \tl_new:N \l__draw_softpath_first_tl
```

```
1515 \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.

```
1516 \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:

__draw_softpath_round_loop:Nnn

__draw_softpath_round_action:nn

__draw_softpath_round_action:Nnn

__draw_softpath_round_action_curveto:NnnNnn

__draw_softpath_round_action_close:

__draw_softpath_round_lookahead:NnnNnn

__draw_softpath_round_roundpoint:NnnNnnNnn

__draw_softpath_round_calc:NnnNnn

__draw_softpath_round_calc:nnnnnn

__draw_softpath_round_calc:fVnnnn

__draw_softpath_round_calc:nnnnw

__draw_softpath_round_close:nn

__draw_softpath_round_close:w

__draw_softpath_round_end:

Rounding corners on a path means going through the entire path and adjusting it. As such, we avoid this entirely if we know there are no corners to deal with. Assuming there is work to do, we recover the existing path and start a loop.

```
1517 \cs_new_protected:Npn \__draw_softpath_round_corners:
```

```
1518 {
```

```
1519 \bool_if:NT \g__draw_softpath_corners_bool
```

```
1520 {
```

```
1521 \group_begin:
```

```
1522 \tl_clear:N \l__draw_softpath_main_tl
```

```
1523 \tl_clear:N \l__draw_softpath_part_tl
```

```
1524 \fp_zero:N \l__draw_softpath_lastx_fp
```

```
1525 \fp_zero:N \l__draw_softpath_lasty_fp
```

```
1526 \tl_clear:N \l__draw_softpath_first_tl
```

```
1527 \tl_clear:N \l__draw_softpath_move_tl
```

```
1528 \tl_build_get:NN \g__draw_softpath_main_tl \l__draw_softpath_internal_tl
```

```
1529 \exp_after:wN \__draw_softpath_round_loop:Nnn
```

```
1530 \l__draw_softpath_internal_tl
```

```
1531 \q__draw_recursion_tail ? ?
```

```
1532 \q__draw_recursion_stop
```

```

1533     \group_end:
1534   }
1535   \bool_gset_false:N \g__draw_softpath_corners_bool
1536 }

```

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.

```

1537 \cs_new_protected:Npn \__draw_softpath_round_loop:Nnn #1#2#3
1538 {
1539   \__draw_if_recursion_tail_stop_do:Nn #1 { \__draw_softpath_round_end: }
1540   \token_if_eq_meaning:NNTF #1 \__draw_softpath_roundpoint_op:nn
1541   { \__draw_softpath_round_action:nn {#2} {#3} }
1542   {
1543     \tl_if_empty:NT \l__draw_softpath_first_tl
1544     { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1545     \fp_set:Nn \l__draw_softpath_lastx_fp {#2}
1546     \fp_set:Nn \l__draw_softpath_lasty_fp {#3}
1547     \token_if_eq_meaning:NNTF #1 \__draw_softpath_moveto_op:nn
1548     {
1549       \tl_put_right:No \l__draw_softpath_main_tl
1550       \l__draw_softpath_move_tl
1551       \tl_put_right:No \l__draw_softpath_main_tl
1552       \l__draw_softpath_part_tl
1553       \tl_set:Nn \l__draw_softpath_move_tl { #1 {#2} {#3} }
1554       \tl_clear:N \l__draw_softpath_first_tl
1555       \tl_clear:N \l__draw_softpath_part_tl
1556     }
1557     { \tl_put_right:Nn \l__draw_softpath_part_tl { #1 {#2} {#3} } }
1558     \__draw_softpath_round_loop:Nnn
1559   }
1560 }
1561 \cs_new_protected:Npn \__draw_softpath_round_action:nn #1#2
1562 {
1563   \dim_set:Nn \l__draw_softpath_corneri_dim {#1}
1564   \dim_set:Nn \l__draw_softpath_cornerii_dim {#2}
1565   \bool_lazy_and:nnTF
1566   { \dim_compare_p:nNn \l__draw_softpath_corneri_dim = { Opt } }
1567   { \dim_compare_p:nNn \l__draw_softpath_cornerii_dim = { Opt } }
1568   { \__draw_softpath_round_loop:Nnn }
1569   { \__draw_softpath_round_action:Nnn }
1570 }

```

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.

```

1571 \cs_new_protected:Npn \__draw_softpath_round_action:Nnn #1#2#3
1572 {
1573   \tl_if_empty:NT \l__draw_softpath_first_tl
1574   { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1575   \token_if_eq_meaning:NNTF #1 \__draw_softpath_curveto_opi:nn
1576   { \__draw_softpath_round_action_curveto:NnnNnn }

```

```

1577     {
1578         \token_if_eq_meaning:NNTF #1 \__draw_softpath_close_op:nn
1579         { \__draw_softpath_round_action_close: }
1580         {
1581             \token_if_eq_meaning:NNTF #1 \__draw_softpath_lineto_op:nn
1582             { \__draw_softpath_round_lookahead:NnnNnn }
1583             { \__draw_softpath_round_loop:Nnn }
1584         }
1585     }
1586     #1 {#2} {#3}
1587 }

```

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.

```

1588 \cs_new_protected:Npn \__draw_softpath_round_action_curveto:NnnNnn
1589     #1#2#3#4#5#6
1590     {
1591         \tl_put_right:Nn \l__draw_softpath_part_tl
1592         { #1 {#2} {#3} #4 {#5} {#6} }
1593         \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1594         \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1595         \__draw_softpath_round_lookahead:NnnNnn
1596     }
1597 \cs_new_protected:Npn \__draw_softpath_round_action_close:
1598     {
1599         \bool_lazy_and:nnTF
1600         { ! \tl_if_empty_p:N \l__draw_softpath_first_tl }
1601         { ! \tl_if_empty_p:N \l__draw_softpath_move_tl }
1602         {
1603             \exp_after:wN \__draw_softpath_round_close:nn
1604             \l__draw_softpath_first_tl
1605         }
1606         { \__draw_softpath_round_loop:Nnn }
1607     }

```

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.

```

1608 \cs_new_protected:Npn \__draw_softpath_round_lookahead:NnnNnn #1#2#3#4#5#6
1609     {
1610         \bool_lazy_any:nTF
1611         {
1612             { \token_if_eq_meaning_p:NN #4 \__draw_softpath_lineto_op:nn }
1613             { \token_if_eq_meaning_p:NN #4 \__draw_softpath_curveto_opi:nn }
1614             { \token_if_eq_meaning_p:NN #4 \__draw_softpath_close_op:nn }
1615         }
1616         {
1617             \__draw_softpath_round_calc:NnnNnn
1618             \__draw_softpath_round_loop:Nnn
1619             {#5} {#6}
1620         }
1621         {
1622             \token_if_eq_meaning:NNTF #4 \__draw_softpath_roundpoint_op:nn
1623             { \__draw_softpath_round_roundpoint:NnnNnnNnn }
1624             { \__draw_softpath_round_loop:Nnn }

```

```

1625     }
1626     #1 {#2} {#3}
1627     #4 {#5} {#6}
1628 }
1629 \cs_new_protected:Npn \__draw_softpath_round_roundpoint:NnnNnnNnn
1630 #1#2#3#4#5#6#7#8#9
1631 {
1632     \__draw_softpath_round_calc:NnnNnn
1633     \__draw_softpath_round_loop:Nnn
1634     {#8} {#9}
1635     #1 {#2} {#3}
1636     #4 {#5} {#6} #7 {#8} {#9}
1637 }

```

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.

```

1638 \cs_new_protected:Npn \__draw_softpath_round_calc:NnnNnn #1#2#3#4#5#6
1639 {
1640     \tl_set:Nc \l__draw_softpath_curve_end_tl
1641     {
1642         \draw_point_interpolate_distance:nnn
1643         \l__draw_softpath_cornerii_dim
1644         { #5 , #6 } { #2 , #3 }
1645     }
1646     \tl_put_right:Nc \l__draw_softpath_part_tl
1647     {
1648         \exp_not:N #4
1649         \__draw_softpath_round_calc:fVnnnn
1650         {
1651             \draw_point_interpolate_distance:nnn
1652             \l__draw_softpath_corneri_dim
1653             { #5 , #6 }
1654             {
1655                 \l__draw_softpath_lastx_fp ,
1656                 \l__draw_softpath_lasty_fp
1657             }
1658         }
1659         \l__draw_softpath_curve_end_tl
1660         {#5} {#6} {#2} {#3}
1661     }
1662     \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1663     \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1664     #1
1665 }

```

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).

```

1666 \cs_new:Npn \__draw_softpath_round_calc:nnnnnn #1#2#3#4#5#6

```

```

1667 {
1668   \__draw_softpath_round_calc:nnnw {#3} {#4} {#5} {#6}
1669   #1 \s__draw_mark #2 \s__draw_stop
1670 }
1671 \cs_generate_variant:Nn \__draw_softpath_round_calc:nnnnnn { fV }

```

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

```

1672 \cs_new:Npn \__draw_softpath_round_calc:nnnw
1673 #1#2#3#4 #5 , #6 \s__draw_mark #7 , #8 \s__draw_stop
1674 {
1675   {#5} {#6}
1676   \exp_not:N \__draw_softpath_curveto_opi:nn
1677   {
1678     \fp_to_dim:n
1679     { #5 + \c__draw_softpath_arc_fp * ( #1 - #5 ) }
1680   }
1681   {
1682     \fp_to_dim:n
1683     { #6 + \c__draw_softpath_arc_fp * ( #2 - #6 ) }
1684   }
1685   \exp_not:N \__draw_softpath_curveto_opii:nn
1686   {
1687     \fp_to_dim:n
1688     { #7 + \c__draw_softpath_arc_fp * ( #1 - #7 ) }
1689   }
1690   {
1691     \fp_to_dim:n
1692     { #8 + \c__draw_softpath_arc_fp * ( #2 - #8 ) }
1693   }
1694   \exp_not:N \__draw_softpath_curveto_opiii:nn
1695   {#7} {#8}
1696 }

```

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.

```

1697 \cs_new_protected:Npn \__draw_softpath_round_close:nn #1#2
1698 {
1699   \use:e
1700   {
1701     \__draw_softpath_round_calc:NnnNnn
1702     {
1703       \tl_set:Nc \exp_not:N \l__draw_softpath_move_tl
1704       {
1705         \__draw_softpath_moveto_op:nn
1706         \exp_not:N \exp_after:wN
1707         \exp_not:N \__draw_softpath_round_close:w
1708         \exp_not:N \l__draw_softpath_curve_end_tl
1709         \s__draw_stop
1710       }
1711     }
1712     \use:e
1713     {
1714       \exp_not:N \exp_not:N \exp_not:N \use_i:nnnn

```

```

1714         {
1715             \__draw_softpath_round_loop:Nnn
1716             \__draw_softpath_close_op:nn
1717             \exp_not:N \exp_after:wN
1718             \exp_not:N \__draw_softpath_round_close:w
1719             \exp_not:N \l__draw_softpath_curve_end_tl
1720             \s__draw_stop
1721         }
1722     }
1723 }
1724 {#1} {#2}
1725 \__draw_softpath_lineto_op:nn
1726 \exp_after:wN \use_none:n \l__draw_softpath_move_tl
1727 }
1728 }
1729 \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.

```

1730 \cs_new_protected:Npn \__draw_softpath_round_end:
1731 {
1732     \tl_put_right:No \l__draw_softpath_main_tl
1733     \l__draw_softpath_move_tl
1734     \tl_put_right:No \l__draw_softpath_main_tl
1735     \l__draw_softpath_part_tl
1736     \tl_build_gclear:N \g__draw_softpath_main_tl
1737     \__draw_softpath_add:o \l__draw_softpath_main_tl
1738 }

```

(End of definition for __draw_softpath_round_corners: and others.)

```

1739 </package>

```

8 l3draw-state implementation

```

1740 <*package>

```

```

1741 <@@=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.

```

1742 \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.

```

1743 \dim_new:N \l_draw_default_linewidth_dim
1744 \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.

```

1745 \cs_new_protected:Npn \draw_linewidth:n #1
1746 {
1747   \dim_gset:Nn \g__draw_linewidth_dim { \fp_to_dim:n {#1} }
1748   \__draw_backend_linewidth:n \g__draw_linewidth_dim
1749 }

```

(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
1750 \cs_new_protected:Npn \draw_dash_pattern:nn #1#2
1751 {
1752   \group_begin:
1753     \seq_set_from_clist:Nn \l__draw_tmp_seq {#1}
1754     \seq_set_map:NnNn \l__draw_tmp_seq \l__draw_tmp_seq
1755       { \fp_to_dim:n {##1} }
1756     \use:e
1757     {
1758       \__draw_backend_dash_pattern:nn
1759       { \seq_use:Nn \l__draw_tmp_seq { , } }
1760       { \fp_to_dim:n {#2} }
1761     }
1762   \group_end:
1763 }
1764 \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.

```

1765 \cs_new_protected:Npn \draw_miterlimit:n #1
1766 { \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: 1767 \cs_new_protected:Npn \draw_cap_but: { \__draw_backend_cap_but: }
\draw_cap_round:      1768 \cs_new_protected:Npn \draw_cap_rectangle: { \__draw_backend_cap_rectangle: }
\draw_evenodd_rule:   1769 \cs_new_protected:Npn \draw_cap_round: { \__draw_backend_cap_round: }
\draw_nonzero_rule:   1770 \cs_new_protected:Npn \draw_evenodd_rule: { \__draw_backend_evenodd_rule: }
\draw_join_bevel:     1771 \cs_new_protected:Npn \draw_nonzero_rule: { \__draw_backend_nonzero_rule: }
\draw_join_miter:     1772 \cs_new_protected:Npn \draw_join_bevel: { \__draw_backend_join_bevel: }
\draw_join_round:     1773 \cs_new_protected:Npn \draw_join_miter: { \__draw_backend_join_miter: }
1774 \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 ??.)

```

1775 </package>

```

9 l3draw-transforms implementation

1776 `<*package>`

1777 `<@@=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.
- `\pgflowlevelsynccm`, `\pgflowlevel`: 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.

1778 `\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` 1779 `\fp_new:N \l__draw_matrix_a_fp`

`\l__draw_matrix_c_fp` 1780 `\fp_new:N \l__draw_matrix_b_fp`

`\l__draw_xshift_dim` 1781 `\fp_new:N \l__draw_matrix_c_fp`

`\l__draw_yshift_dim` 1782 `\fp_new:N \l__draw_matrix_d_fp`

1783 `\dim_new:N \l__draw_xshift_dim`

1784 `\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:` 1785 `\cs_new_protected:Npn \draw_transform_matrix_reset:`

1786 `{`

1787 `\fp_set:Nn \l__draw_matrix_a_fp { 1 }`

1788 `\fp_zero:N \l__draw_matrix_b_fp`

1789 `\fp_zero:N \l__draw_matrix_c_fp`

1790 `\fp_set:Nn \l__draw_matrix_d_fp { 1 }`

1791 `}`

1792 `\cs_new_protected:Npn \draw_transform_shift_reset:`

1793 `{`

1794 `\dim_zero:N \l__draw_xshift_dim`

1795 `\dim_zero:N \l__draw_yshift_dim`

1796 `}`

1797 `\draw_transform_matrix_reset:`

1798 `\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:nnnn` Setting the transform matrix is straight-forward, with just a bit of expansion to sort out.
`\draw_transform_shift_absolute:n` With the mechanism active, the identity matrix is set.

```

1799 \cs_new_protected:Npn \draw_transform_matrix_absolute:nnnn #1#2#3#4
1800 {
1801   \fp_set:Nn \l__draw_matrix_a_fp {#1}
1802   \fp_set:Nn \l__draw_matrix_b_fp {#2}
1803   \fp_set:Nn \l__draw_matrix_c_fp {#3}
1804   \fp_set:Nn \l__draw_matrix_d_fp {#4}
1805   \bool_lazy_all:nTF
1806   {
1807     { \fp_compare_p:nNn \l__draw_matrix_a_fp = \c_one_fp }
1808     { \fp_compare_p:nNn \l__draw_matrix_b_fp = \c_zero_fp }
1809     { \fp_compare_p:nNn \l__draw_matrix_c_fp = \c_zero_fp }
1810     { \fp_compare_p:nNn \l__draw_matrix_d_fp = \c_one_fp }
1811   }
1812   { \bool_set_false:N \l__draw_matrix_active_bool }
1813   { \bool_set_true:N \l__draw_matrix_active_bool }
1814 }
1815 \cs_new_protected:Npn \draw_transform_shift_absolute:n #1
1816 {
1817   \__draw_point_process:nn
1818   { \__draw_transform_shift_absolute:nn } {#1}
1819 }
1820 \cs_new_protected:Npn \__draw_transform_shift_absolute:nn #1#2
1821 {
1822   \dim_set:Nn \l__draw_xshift_dim {#1}
1823   \dim_set:Nn \l__draw_yshift_dim {#2}
1824 }

```

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

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

```

1825 \cs_new_protected:Npn \draw_transform_matrix:nnnn #1#2#3#4
1826 {
1827   \use:e
1828   {
1829     \__draw_transform:nnnn
1830     { \fp_eval:n {#1} }
1831     { \fp_eval:n {#2} }
1832     { \fp_eval:n {#3} }
1833     { \fp_eval:n {#4} }
1834   }
1835 }
1836 \cs_new_protected:Npn \__draw_transform:nnnn #1#2#3#4
1837 {
1838   \use:e
1839   {
1840     \draw_transform_matrix_absolute:nnnn
1841     { #1 * \l__draw_matrix_a_fp + #2 * \l__draw_matrix_c_fp }
1842     { #1 * \l__draw_matrix_b_fp + #2 * \l__draw_matrix_d_fp }
1843     { #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_c_fp }
1844     { #3 * \l__draw_matrix_b_fp + #4 * \l__draw_matrix_d_fp }

```

```

1845     }
1846   }
1847   \cs_new_protected:Npn \draw_transform_shift:n #1
1848   {
1849     \__draw_point_process:nn
1850     { \__draw_transform_shift:nn } {#1}
1851   }
1852   \cs_new_protected:Npn \__draw_transform_shift:nn #1#2
1853   {
1854     \dim_set:Nn \l__draw_xshift_dim { \l__draw_xshift_dim + #1 }
1855     \dim_set:Nn \l__draw_yshift_dim { \l__draw_yshift_dim + #2 }
1856   }

```

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

```

\draw_transform_matrix_invert: Standard mathematics: calculate the inverse matrix and use that, then undo the shifts.
\__draw_transform_invert:n 1857 \cs_new_protected:Npn \draw_transform_matrix_invert:
\__draw_transform_invert:f 1858 {
\draw_transform_shift_invert: 1859   \bool_if:NT \l__draw_matrix_active_bool
1860   {
1861     \__draw_transform_invert:f
1862     {
1863       \fp_eval:n
1864       {
1865         1 /
1866         (
1867           \l__draw_matrix_a_fp * \l__draw_matrix_d_fp
1868           - \l__draw_matrix_b_fp * \l__draw_matrix_c_fp
1869         )
1870       }
1871     }
1872   }
1873 }
1874 \cs_new_protected:Npn \__draw_transform_invert:n #1
1875 {
1876   \fp_set:Nn \l__draw_matrix_a_fp
1877   { \l__draw_matrix_d_fp * #1 }
1878   \fp_set:Nn \l__draw_matrix_b_fp
1879   { -\l__draw_matrix_b_fp * #1 }
1880   \fp_set:Nn \l__draw_matrix_c_fp
1881   { -\l__draw_matrix_c_fp * #1 }
1882   \fp_set:Nn \l__draw_matrix_d_fp
1883   { \l__draw_matrix_a_fp * #1 }
1884 }
1885 \cs_generate_variant:Nn \__draw_transform_invert:n { f }
1886 \cs_new_protected:Npn \draw_transform_shift_invert:
1887 {
1888   \dim_set:Nn \l__draw_xshift_dim { -\l__draw_xshift_dim }
1889   \dim_set:Nn \l__draw_yshift_dim { -\l__draw_yshift_dim }
1890 }

```

(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.

```

1891 \cs_new_protected:Npn \draw_transform_triangle:nnn #1#2#3
1892 {
1893   \__draw_point_process:nnn
1894   {
1895     \__draw_point_process:nn
1896     { \__draw_transform_triangle:nnnnnn }
1897     {#1}
1898   }
1899   {#2} {#3}
1900 }
1901 \cs_new_protected:Npn \__draw_transform_triangle:nnnnnn #1#2#3#4#5#6
1902 {
1903   \use:e
1904   {
1905     \draw_transform_matrix_absolute:nnnn
1906     { #3 - #1 }
1907     { #4 - #2 }
1908     { #5 - #1 }
1909     { #6 - #2 }
1910     \draw_transform_shift_absolute:n { #1 , #2 }
1911   }
1912 }

```

(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 1913 \cs_new_protected:Npn \draw_transform_scale:n #1
\draw_transform_yscale:n 1914 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { #1 } }
\draw_transform_xshift:n 1915 \cs_new_protected:Npn \draw_transform_xscale:n #1
\draw_transform_yshift:n 1916 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { #1 } }
\draw_transform_xslant:n 1917 \cs_new_protected:Npn \draw_transform_yscale:n #1
\draw_transform_yslant:n 1918 { \draw_transform_matrix:nnnn { 1 } { 0 } { 0 } { #1 } }
1919 \cs_new_protected:Npn \draw_transform_xshift:n #1
1920 { \draw_transform_shift:n { #1 , Opt } }
1921 \cs_new_protected:Npn \draw_transform_yshift:n #1
1922 { \draw_transform_shift:n { Opt , #1 } }
1923 \cs_new_protected:Npn \draw_transform_xslant:n #1
1924 { \draw_transform_matrix:nnnn { 1 } { 0 } { #1 } { 1 } }
1925 \cs_new_protected:Npn \draw_transform_yslant:n #1
1926 { \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 1927 \cs_new_protected:Npn \draw_transform_rotate:n #1
\__draw_transform_rotate:f 1928 { \__draw_transform_rotate:f { \fp_eval:n {#1} } }
\__draw_transform_rotate:nn 1929 \cs_new_protected:Npn \__draw_transform_rotate:n #1
\__draw_transform_rotate:ff 1930 {
1931   \__draw_transform_rotate:ff
1932   { \fp_eval:n { cosd(#1) } }
1933   { \fp_eval:n { sind(#1) } }
1934 }

```

```

1935 \cs_generate_variant:Nn \_draw_transform_rotate:n { f }
1936 \cs_new_protected:Npn \_draw_transform_rotate:nn #1#2
1937   { \draw_transform_matrix:nnnn {#1} {#2} { -#2 } { #1 } }
1938 \cs_generate_variant:Nn \_draw_transform_rotate:nn { ff }

```

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

```

1939 \endpackage

```

Index

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

B

`\begin` ... 171, 788, 1081, 1085, 1109, 1112

bool commands:

`\bool_gset_eq:NN` 769

`\bool_gset_false:N` 1433, 1535

`\bool_gset_true:N` 1438, 1484

`\bool_if:NTF`
 . 21, 115, 194, 233, 684, 700, 701,
 705, 1204, 1236, 1354, 1466, 1519, 1859

`\bool_lazy_all:nTF` 1805

`\bool_lazy_and:nnTF`
 225, 679, 1565, 1599

`\bool_lazy_any:nTF` 1610

`\bool_lazy_or:nnTF` . 560, 655, 688, 693

`\bool_new:N`
 86, 220, 643, 644, 645, 646,
 647, 747, 1260, 1345, 1421, 1437, 1778

`\bool_set_eq:NN` 761

`\bool_set_false:N`
 96, 228, 666, 667, 668, 687, 1812

`\bool_set_true:N` 98, 229,
 672, 710, 714, 715, 1279, 1349, 1813

box commands:

`\box_dp:N` 17, 67

`\box_gset_eq:NN` 156

`\box_gset_wd:Nn` 100, 133

`\box_ht:N` 17, 69

`\box_if_exist:NTF` 93

`\box_move_down:nn` 1319, 1332

`\box_move_up:nn` 51

`\box_new:N` 13, 80, 81, 1261, 1262

`\box_set_dp:Nn` 55, 1323, 1336

`\box_set_eq:NN` 145

`\box_set_ht:Nn` 54, 1324, 1342

`\box_set_wd:Nn` 56, 128, 1308

`\box_use_drop:N` 52,
 57, 102, 129, 134, 1311, 1321, 1334

`\box_wd:N` 17, 66, 68

C

clist commands:

`\clist_map_inline:Nn` .. 124, 141, 152

`\clist_map_inline:nn` 669

`\clist_new:N` 87, 89

`\clist_set:Nn` 88, 1296

coffin commands:

`\coffin_typeset:Nnnnn` 64

`\coffin_wd:N` 66

color commands:

`\color_select:n` 1284

cs commands:

`\cs_generate_variant:Nn`
 420, 609, 642,
 847, 855, 897, 916, 923, 931, 938,
 947, 953, 973, 981, 988, 994, 1006,
 1009, 1027, 1045, 1053, 1059, 1080,
 1094, 1108, 1129, 1164, 1183, 1196,
 1424, 1486, 1671, 1885, 1935, 1938

`\cs_if_exist:NTF` 671

`\cs_if_exist_use:NTF` .. 403, 412, 674

`\cs_new:Npn` 513, 523, 533, 543, 792,
 798, 800, 802, 809, 811, 813, 821,
 823, 826, 835, 840, 843, 845, 848,
 849, 851, 853, 856, 858, 864, 873,
 879, 889, 898, 904, 909, 917, 924,
 932, 939, 948, 954, 960, 965, 974,
 982, 989, 995, 1001, 1007, 1010,
 1016, 1025, 1028, 1034, 1039, 1046,
 1054, 1060, 1066, 1072, 1086, 1095,
 1101, 1113, 1115, 1124, 1154, 1156,
 1165, 1170, 1184, 1186, 1188, 1197,
 1202, 1229, 1234, 1666, 1672, 1729

`\cs_new_protected:Npn` . 14, 19, 60,
 75, 90, 113, 122, 139, 150, 184, 206,
 213, 221, 231, 240, 246, 252, 258,
 265, 276, 284, 289, 291, 293, 302,
 309, 345, 347, 358, 364, 394, 421,
 450, 456, 462, 467, 475, 484, 491,
 499, 554, 556, 569, 576, 585, 591,
 593, 603, 610, 616, 623, 648, 653,
 664, 708, 712, 717, 724, 748, 766,
 1136, 1138, 1140, 1142, 1146, 1264,
 1271, 1292, 1314, 1327, 1347, 1352,
 1367, 1374, 1385, 1394, 1402, 1410,
 1422, 1425, 1430, 1439, 1448, 1457,
 1462, 1472, 1480, 1487, 1489, 1491,
 1493, 1495, 1497, 1499, 1501, 1502,
 1504, 1506, 1517, 1537, 1561, 1571,
 1588, 1597, 1608, 1629, 1638, 1697,
 1730, 1745, 1750, 1765, 1767, 1768,
 1769, 1770, 1771, 1772, 1773, 1774,
 1785, 1792, 1799, 1815, 1820, 1825,
 1836, 1847, 1852, 1857, 1874, 1886,
 1891, 1901, 1913, 1915, 1917, 1919,
 1921, 1923, 1925, 1927, 1929, 1936

D

dim commands:

`\dim_abs:n` 598, 599
`\dim_compare:nNnTF` 605, 612, 726, 1300
`\dim_compare_p:nNn` 226, 227, 1566, 1567
`\dim_eval:n` 598, 599
`\dim_gset:Nn` 186, 188,
190, 192, 196, 198, 200, 202, 208,
209, 210, 211, 215, 216, 728, 1266,
1267, 1268, 1269, 1468, 1469, 1747
`\dim_gset_eq:NN`
.... 773, 774, 775, 776, 777, 778,
779, 780, 1377, 1396, 1397, 1398, 1399
`\dim_gzero:N` .. 1302, 1303, 1304, 1305
`\dim_max:nn` .. 187, 191, 197, 201, 1338
`\dim_min:nn` 189, 193, 199, 203
`\dim_new:N` 178, 179,
180, 181, 182, 183, 218, 219, 739,
740, 741, 742, 743, 744, 745, 746,
1130, 1131, 1132, 1133, 1134, 1135,
1256, 1257, 1258, 1259, 1346, 1364,
1381, 1382, 1383, 1384, 1435, 1436,
1512, 1513, 1742, 1743, 1783, 1784
`\dim_set:Nn` 223, 224,
1148, 1149, 1350, 1563, 1564, 1744,
1822, 1823, 1854, 1855, 1888, 1889
`\dim_set_eq:NN`
.... 751, 752, 753, 754, 755, 756,
757, 758, 1371, 1388, 1389, 1390, 1391
`\dim_step_inline:nnnn` 625, 633
`\dim_use:N`
.... 726, 731, 733, 1359, 1444, 1445
`\dim_zero:N` 1794, 1795
`\c_max_dim` 208, 209, 210,
211, 726, 1266, 1267, 1268, 1269, 1300

draw commands:

`\draw_baseline:n` 1347, 1347
`\l_draw_bb_update_bool`
.... 21, 194, 680, 687, 1260, 1279
`\draw_begin:` 1271, 1271
`\draw_box_use:N` 14, 14
`\draw_cap_but:` 1286, 1767, 1767
`\draw_cap_rectangle:` 1767, 1768
`\draw_cap_round:` 1767, 1769
`\draw_coffin_use:Nnn` 36, 60, 60
`\draw_dash_pattern:nn` 1289, 1750, 1750
`\l_draw_default_linewidth_dim` ...
.... 105, 1283, 1743
`\draw_end:` 1271, 1292
`\draw_evenodd_rule:` 1767, 1770
`\draw_join_bevel:` 1767, 1772
`\draw_join_miter:` ... 1287, 1767, 1773
`\draw_join_round:` 1767, 1774
`\draw_layer_begin:n` 90, 90

`\draw_layer_end:` 90, 113
`\draw_layer_new:n` 75, 75
`\l_draw_layers_clist`
.... 87, 124, 141, 152, 1296
`\draw_linewidth:n` 105, 1283, 1745, 1745
`\draw_miterlimit:n` .. 1288, 1765, 1765
`\draw_nonzero_rule:` . 1285, 1767, 1771
`\draw_path_arc:nnn` 345, 345, 488
`\draw_path_arc:nnnn` ... 345, 346, 347
`\draw_path_arc_axes:nnnn` ... 484, 484
`\draw_path_canvas_curveto:nnn` ...
.... 289, 293
`\draw_path_canvas_lineto:n` . 289, 291
`\draw_path_canvas_moveto:n` . 289, 289
`\draw_path_circle:nn` 554, 554
`\draw_path_close:` 284, 284, 582
`\draw_path_corner_arc:nn` ... 221, 221
`\draw_path_curveto:nn` 302, 302
`\draw_path_curveto:nnn` 240, 265
`\draw_path_ellipse:nnn` . 491, 491, 555
`\draw_path_grid:nnnn` 593, 593
`\draw_path_lineto:n`
.... 240, 252, 579, 580, 581, 631, 639
`\draw_path_moveto:n`
.... 240, 240, 578, 583, 630, 638
`\draw_path_rectangle:nn` 556, 556, 592
`\draw_path_rectangle_corners:nn` .
.... 585, 585
`\draw_path_scope_begin:`
.... 748, 748, 1372, 1405
`\draw_path_scope_end:`
.... 748, 766, 1376, 1413
`\draw_path_use:n` 648, 648
`\draw_path_use_clear:n` 648, 653
`\draw_point:n` 795, 805,
806, 816, 817, 818, 829, 830, 831,
832, 843, 843, 854, 869, 891, 950,
991, 1008, 1026, 1056, 1126, 1158,
1172, 1190, 1206, 1222, 1238, 1251
`\draw_point_interpolate_arcaxes:nnnnnn`
.... 1028, 1028
`\draw_point_interpolate_curve:nnnnnn`
.... 1060
`\draw_point_interpolate_curve:nnnnnnn`
.... 1060
`\draw_point_interpolate_curve_-`
auxi:nnnnnnnnn 1060
`\draw_point_interpolate_curve_-`
auxii:nnnnnnnnn 1060
`\draw_point_interpolate_curve_-`
auxiii:nnnnnnn 1060
`\draw_point_interpolate_curve_-`
auxiv:nnnnnn 1060

\draw_point_interpolate_curve- auxv:nnw	1060	\draw_transform_triangle:nnn ...	487, 1891, 1891
\draw_point_interpolate_curve- auxvi:n	1060	\draw_transform_xscale:n .	1913, 1915
\draw_point_interpolate_curve- auxvii:nnnnnnnn	1060	\draw_transform_xshift:n .	1913, 1919
\draw_point_interpolate_curve- auxviii:nnnnnn	1060	\draw_transform_xslant:n .	1913, 1923
\draw_point_interpolate_distance:nnn	1010, 1010, 1642, 1651	\draw_transform_yscale:n .	1913, 1917
\draw_point_interpolate_line:nnn	995, 995	\draw_transform_yshift:n .	1913, 1921
\draw_point_intersect_circles:nnnnn	898, 898	\draw_transform_yslant:n .	1913, 1925
\draw_point_intersect_line- circle:nnnnn	954, 954	\draw_xvec:n	1136, 1136, 1151
\draw_point_intersect_lines:nnnn	873, 873	\draw_yvec:n	1136, 1138, 1152
\draw_point_polar:nn	849, 849	\draw_zvec:n	1136, 1140, 1153
\draw_point_polar:nnn	428, 434, 438, 444, 849, 850, 851	draw internal commands:	
\draw_point_transform:n	25, 28, 31, 34, 244, 256, 272, 273, 274, 306, 307, 433, 437, 495, 566, 1197, 1197	_draw_backend_begin:	1276
\draw_point_unit_vector:n	856, 856, 1023	_draw_backend_box_use:Nnnnn ..	41
\draw_point_vec:nn	1154, 1154	_draw_backend_cap_but:	1767
\draw_point_vec:nnn	1154, 1165	_draw_backend_cap_rectangle: ..	1768
\draw_point_vec_polar:nn .	1184, 1184	_draw_backend_cap_round: ..	1769
\draw_point_vec_polar:nnn	1184, 1185, 1186	_draw_backend_clip:	686
\draw_scope_begin:	1367, 1367	_draw_backend_closepath: ..	1488
\draw_scope_end:	1374	_draw_backend_curveto:nnnnnn	1492
\draw_suspend_begin:	1402, 1402	_draw_backend_dash_pattern:nn	1758
\draw_suspend_end:	1402, 1410	_draw_backend_discardpath: ..	691
\draw_transform_matrix:nnnn	1825, 1825, 1914, 1916, 1918, 1924, 1926, 1937	_draw_backend_end:	1298
\draw_transform_matrix_absolute:nnnn	1799, 1799, 1840, 1905	_draw_backend_evenodd_rule: ..	1770
\draw_transform_matrix_invert: ..	1857, 1857	_draw_backend_join_bevel: ..	1772
\draw_transform_matrix_reset: ..	1280, 1406, 1785, 1785, 1797	_draw_backend_join_miter: ..	1773
\draw_transform_rotate:n .	1927, 1927	_draw_backend_join_round: ..	1774
\draw_transform_scale:n .	1913, 1913	_draw_backend_lineto:nn	1498
\draw_transform_shift:n	1825, 1847, 1920, 1922	_draw_backend_linewidth:n ..	1748
\draw_transform_shift_absolute:n	1799, 1815, 1910	_draw_backend_miterlimit:n .	1766
\draw_transform_shift_invert: ..	1857, 1886	_draw_backend_moveto:nn	1500
\draw_transform_shift_reset: ..	1281, 1407, 1785, 1792, 1798	_draw_backend_nonzero_rule: ..	1771
		_draw_backend_rectangle:nnnn	1505
		_draw_backend_scope_begin: ..	132, 1369
		_draw_backend_scope_end: ..	135, 1379
		_l_draw_baseline_bool	
		1345, 1349, 1354
		_l_draw_baseline_dim	1345, 1350, 1359
		_draw_baseline_finalise:w	1294, 1352, 1352
		_draw_box_use:Nnnnn .	14, 16, 19, 65
		_l_draw_corner_arc_bool	220, 228, 229, 233, 561
		_l_draw_corner_xarc_dim	218, 223, 226, 236
		_l_draw_corner_yarc_dim	218, 224, 227, 237
		_draw_draw_polar:nnn	849, 852, 853, 855
		_draw_draw_vec_polar:nnn	1187, 1188, 1196
		_l_draw_fill_color_tl	1364

```

\__draw_finalise: .....
..... 1307, 1314, 1314, 1352, 1362
\__draw_finalise_baseline:n ....
..... 1314, 1327, 1359
\g__draw_id_int ..... 1263, 1274
\__draw_if_recursion_tail_stop_-
do:Nn ..... 9, 9, 1539
\l__draw_layer_close_bool .....
..... 86, 96, 98, 115
\l__draw_layer_main_box .....
..... 128, 129, 1261, 1290
\l__draw_layer_tl ..... 84, 95, 99
\g__draw_layers_clist ..... 87
\__draw_layers_insert: 122, 122, 1297
\__draw_layers_restore: 139, 150, 1412
\__draw_layers_save: . 139, 139, 1408
\g__draw_linewidth_dim .....
... 734, 1371, 1377, 1742, 1747, 1748
\l__draw_linewidth_dim .....
..... 1364, 1371, 1377
\l__draw_main_box .....
1261, 1275, 1308, 1311, 1316, 1321,
1323, 1324, 1329, 1334, 1336, 1342
\l__draw_matrix_a_fp .....
. 42, 1209, 1241, 1779, 1787, 1801,
1807, 1841, 1843, 1867, 1876, 1883
\l__draw_matrix_active_bool ....
562, 1204, 1236, 1778, 1812, 1813, 1859
\l__draw_matrix_b_fp .....
. 43, 1215, 1246, 1779, 1788, 1802,
1808, 1842, 1844, 1868, 1878, 1879
\l__draw_matrix_c_fp .....
. 44, 1210, 1242, 1779, 1789, 1803,
1809, 1841, 1843, 1868, 1880, 1881
\l__draw_matrix_d_fp .....
. 45, 1216, 1247, 1782, 1790, 1804,
1810, 1842, 1844, 1867, 1877, 1882
\__draw_path_arc:nnnn . 345, 351, 358
\__draw_path_arc:nnNnn .....
..... 345, 361, 362, 364
\c__draw_path_arc_60_fp ..... 345
\c__draw_path_arc_90_fp ..... 345
\__draw_path_arc_add:nnnn ..... 345
\__draw_path_arc_aux_add:nn ....
..... 452, 458, 470, 475
\__draw_path_arc_auxi:nnnnNnn ...
..... 345, 372, 379, 387, 394, 420
\__draw_path_arc_auxii:nnnnnnn .
..... 345, 398, 421
\__draw_path_arc_auxiii:nn .....
..... 345, 425, 450
\__draw_path_arc_auxiv:nnnn .....
..... 345, 431, 456
\__draw_path_arc_auxv:nn 345, 441, 462
\__draw_path_arc_auxvi:nn .....
..... 345, 464, 467
\l__draw_path_arc_delta_fp .... 345
\l__draw_path_arc_start_fp .... 345
\__draw_path_curveto:nnnn .....
..... 302, 305, 309
\__draw_path_curveto:nnnnnn .....
..... 240, 270,
276, 298, 316, 446, 515, 525, 535, 545
\c__draw_path_curveto_a_fp .... 302
\c__draw_path_curveto_b_fp .... 302
\__draw_path_ellipse:nnnnnn .....
..... 491, 494, 499
\__draw_path_ellipse_arci:nnnnnn
..... 491, 505, 513
\__draw_path_ellipse_arci:nnnnnn
..... 491, 506, 523
\__draw_path_ellipse_arci:nnnnnn
..... 491, 507, 533
\__draw_path_ellipse_arci:nnnnnn
..... 491, 508, 543
\c__draw_path_ellipse_fp ..... 491
\__draw_path_grid_auxi:nnnnnn ...
..... 593, 597, 603, 609
\__draw_path_grid_auxii:nnnnnn .
..... 593, 606, 607, 610
\__draw_path_grid_auxiii:nnnnnn .
..... 593, 613, 614, 616
\__draw_path_grid_auxiiii:nnnnnn 593
\__draw_path_grid_auxiv:nnnnnnnn
..... 593, 618, 623, 642
\g__draw_path_lastx_dim .....
..... 178, 215, 320, 453, 459, 751, 779
\l__draw_path_lastx_dim 739, 751, 779
\g__draw_path_lasty_dim .....
..... 178, 216, 327, 454, 460, 752, 780
\l__draw_path_lasty_dim 739, 752, 780
\__draw_path_lineto:nn .....
..... 240, 255, 258, 292
\__draw_path_mark_corner: .....
231, 231, 260, 269, 286, 297, 315, 386
\__draw_path_moveto:nn .....
..... 240, 243, 246, 290, 503, 511
\__draw_path_rectangle:nnnn .....
..... 556, 564, 569
\__draw_path_rectangle_corners:nnnn
..... 585
\__draw_path_rectangle_corners:nnnnn
..... 588, 591
\__draw_path_rectangle_rounded:nnnn
..... 556, 563, 576
\__draw_path_reset_limits: .....
..... 184, 206, 660, 759, 1278

```

\l__draw_path_tmp_tl	__draw_point_interpolate_curve_-
.... 175 , 423 , 446 , 465 , 469 , 473 , 477	auxii:nnnnnnnnn . 1068 , 1072 , 1080
\l__draw_path_tmpa_fp	__draw_point_interpolate_curve_-
..... 175 , 311 , 321 , 333	auxiii:nnnnnn . 1075 , 1086 , 1094
\l__draw_path_tmpb_fp	__draw_point_interpolate_curve_-
..... 175 , 312 , 328 , 337	auxiv:nnnnnn 1088 , 1089 , 1090 , 1095
__draw_path_update_last:nn	__draw_point_interpolate_curve_-
..... 213 , 213 , 250 , 263 , 282 , 574	auxv:nnw 1097 , 1101 , 1108
__draw_path_update_limits:nn ...	__draw_point_interpolate_curve_-
..... 24 , 27 , 30 , 33 , 184 ,	auxvi:n 1092 , 1113
184 , 248 , 261 , 278 , 279 , 280 , 571 , 572	__draw_point_interpolate_curve_-
__draw_path_use:n . 648 , 651 , 662 , 664	auxvii:nnnnnnnn 1114 , 1115
__draw_path_use_action_draw: ...	__draw_point_interpolate_curve_-
..... 648 , 708	auxviii:nnnnnn . 1117 , 1124 , 1129
__draw_path_use_action_fillstroke:	__draw_point_interpolate_-
..... 648 , 712	distance:nnnn 1013 , 1016
\l__draw_path_use_bb_bool 646	__draw_point_interpolate_-
\l__draw_path_use_clear_bool 646 , 705	distance:nnnnn
\l__draw_path_use_clip_bool 1010 , 1020 , 1025 , 1027
..... 643 , 666 , 684	__draw_point_interpolate_-
\l__draw_path_use_fill_bool	distance:nnnnnn 1010
..... 643 , 667 , 689 , 694 , 700 , 714	__draw_point_interpolate_line_-
__draw_path_use_stroke_bb:	aux:nnnnnn . 995 , 998 , 1001 , 1006
..... 648 , 682 , 717	__draw_point_interpolate_line_-
__draw_path_use_stroke_bb_-	aux:nnnnnn . 995 , 1003 , 1007 , 1009
aux:NnN 648 , 719 , 720 , 721 , 722 , 724	__draw_point_intersect_circles_-
\l__draw_path_use_stroke_bool ...	auxi:nnnnnnnn 898 , 901 , 904
643 , 668 , 681 , 690 , 695 , 701 , 710 , 715	__draw_point_intersect_circles_-
\g__draw_path_xmax_dim	auxii:nnnnnnnn . 898 , 906 , 909 , 916
..... 180 , 186 , 187 , 208 , 753 , 775	__draw_point_intersect_circles_-
\l__draw_path_xmax_dim . 739 , 753 , 775	auxiii:nnnnnnnn . 898 , 911 , 917 , 923
\g__draw_path_xmin_dim	__draw_point_intersect_circles_-
..... 180 , 188 , 189 , 209 , 754 , 776	auxiv:nnnnnnnn . 898 , 919 , 924 , 931
\l__draw_path_xmin_dim . 739 , 754 , 776	__draw_point_intersect_circles_-
\g__draw_path_ymax_dim	auxv:nnnnnnnnnn . 898 , 926 , 932 , 938
..... 180 , 190 , 191 , 210 , 755 , 777	__draw_point_intersect_circles_-
\l__draw_path_ymax_dim . 739 , 755 , 777	auxvi:nnnnnnnnnn . 898 , 934 , 939 , 947
\g__draw_path_ymin_dim	__draw_point_intersect_circles_-
..... 180 , 192 , 193 , 211 , 756 , 778	auxvii:nnnnnnnn . 898 , 941 , 948 , 953
\l__draw_path_ymin_dim . 739 , 756 , 778	__draw_point_intersect_line_-
__draw_point_interpolate_-	circle_auxi:nnnnnnnn 954 , 957 , 960
arcaxes_auxi:nnnnnnnnnn	__draw_point_intersect_line_-
..... 1028 , 1031 , 1034	circle_auxii:nnnnnnnnnn
__draw_point_interpolate_- 954 , 962 , 965 , 973
arcaxes_auxii:nnnnnnnnnn	__draw_point_intersect_line_-
..... 1028 , 1036 , 1039 , 1045	circle_auxiii:nnnnnnnnnn
__draw_point_interpolate_- 954 , 967 , 974 , 981
arcaxes_auxiii:nnnnnnnn	__draw_point_intersect_line_-
..... 1028 , 1041 , 1046 , 1053	circle_auxiv:nnnnnnnnnn
__draw_point_interpolate_- 954 , 976 , 982 , 988
arcaxes_auxiv:nnnnnnnnnn	__draw_point_intersect_line_-
..... 1028 , 1048 , 1054 , 1059	circle_auxv:nnnnnn 954 , 984 , 989 , 994
__draw_point_interpolate_curve_-	__draw_point_intersect_lines:nnnnnn
auxi:nnnnnnnnnn 1063 , 1066 873

```

\__draw_point_intersect_lines:nnnnnnnn
..... 873, 876, 879
\__draw_point_intersect_lines_-
aux:nnnnnn ..... 873, 881, 889, 897
\__draw_point_process:nn .....
23, 26, 29, 32, 242, 254, 290, 292,
424, 440, 792, 792, 857, 1012, 1018,
1144, 1199, 1231, 1817, 1849, 1895
\__draw_point_process:nnn 304, 430,
558, 587, 595, 792, 802, 900, 997, 1893
\__draw_point_process:nnnn .....
... 267, 295, 493, 792, 813, 956, 1030
\__draw_point_process:nnnn .....
..... 792, 826, 875, 1062
\__draw_point_process_auxi:nn ...
..... 792, 794, 798
\__draw_point_process_auxii:nw ..
..... 792, 799, 800
\__draw_point_process_auxiii:nnn
..... 792, 804, 809
\__draw_point_process_auxiv:nw ..
..... 792, 810, 811
\__draw_point_process_auxv:nnnn .
..... 792, 815, 821
\__draw_point_process_auxvi:nw ..
..... 792, 822, 823
\__draw_point_process_auxvii:nnnnn
..... 792, 828, 835
\__draw_point_process_auxviii:nw
..... 792, 837, 840
\__draw_point_to_dim:n .....
..... 843, 844, 845, 847
\__draw_point_to_dim:w . 843, 846, 848
\__draw_point_transform:nn .....
..... 1197, 1200, 1202
\__draw_point_transform_noshift:n
..... 427, 443, 496, 497, 1229, 1229
\__draw_point_transform_noshift:nn
..... 1229, 1232, 1234
\__draw_point_unit_vector:nn ...
..... 856, 857, 858
\__draw_point_unit_vector:nnn ...
..... 856, 860, 864
\__draw_point_vec:nn .....
..... 1154, 1155, 1156, 1164
\__draw_point_vec:nnn .....
..... 1154, 1167, 1170, 1183
\__draw_point_vec_polar:nnn .. 1184
\__draw_reset_bb: .....
..... 1264, 1264, 1277, 1392
\__draw_scope_bb_begin: .....
..... 1385, 1385, 1404
\__draw_scope_bb_end: 1385, 1394, 1414

\__draw_softpath_add:n .....
..... 772, 1422, 1422, 1424, 1441,
1450, 1459, 1464, 1474, 1482, 1737
\c__draw_softpath_arc_fp .....
..... 1516, 1679, 1683, 1688, 1692
\__draw_softpath_clear: .....
. 659, 706, 764, 768, 1282, 1425, 1430
\__draw_softpath_close_op:nn ...
.. 1443, 1487, 1487, 1578, 1614, 1716
\__draw_softpath_closepath: ....
..... 287, 510, 1439
\l__draw_softpath_corneri_dim ...
..... 1510, 1563, 1566, 1652
\l__draw_softpath_cornerii_dim ..
..... 1510, 1564, 1567, 1643
\g__draw_softpath_corners_bool ..
763, 770, 1421, 1433, 1484, 1519, 1535
\l__draw_softpath_corners_bool ..
..... 739, 762, 771
\l__draw_softpath_curve_end_tl ..
..... 1509, 1640, 1659, 1708, 1719
\__draw_softpath_curveto:nnnnnn .
..... 281, 1439, 1448
\__draw_softpath_curveto_opi:nn .
.. 1452, 1487, 1489, 1575, 1613, 1676
\__draw_softpath_curveto_-
opi:nnNnnNnn ..... 1487, 1490, 1491
\__draw_softpath_curveto_opii:nn
..... 1453, 1487, 1493, 1494, 1685
\__draw_softpath_curveto_-
opiii:nn 1454, 1487, 1495, 1496, 1694
\l__draw_softpath_first_tl .....
..... 1510, 1526, 1543,
1544, 1554, 1573, 1574, 1600, 1604
\l__draw_softpath_internal_tl ...
..... 1420, 1427, 1428, 1528, 1530
\g__draw_softpath_lastx_dim ....
..... 757, 773, 1435, 1444, 1468
\l__draw_softpath_lastx_dim ....
..... 745, 757, 773
\l__draw_softpath_lastx_fp .....
.. 1510, 1524, 1545, 1593, 1655, 1662
\g__draw_softpath_lasty_dim ....
..... 758, 774, 1435, 1445, 1469
\l__draw_softpath_lasty_dim ....
..... 746, 758, 774
\l__draw_softpath_lasty_fp .....
.. 1510, 1525, 1546, 1594, 1656, 1663
\__draw_softpath_lineto:nn .....
..... 262, 1439, 1457
\__draw_softpath_lineto_op:nn ...
.. 1460, 1487, 1497, 1581, 1612, 1725
\g__draw_softpath_main_tl .....
760, 1419, 1423, 1427, 1432, 1528, 1736

```

\l__draw_softpath_main_tl	__draw_softpath_roundpoint_-
19, 760, 772, 1507,	op:nn . . 1483, 1487, 1501, 1540, 1622
1522, 1549, 1551, 1732, 1734, 1737	__draw_softpath_use: 683, 1425, 1425
\g__draw_softpath_move_bool	\l__draw_stroke_color_tl 1364
1437, 1466	\l__draw_tmp_box 13, 37, 48,
\l__draw_softpath_move_tl	52, 54, 55, 56, 57, 63, 65, 66, 67, 68, 69
1510, 1527,	\l__draw_tmp_seq 1750
1550, 1553, 1601, 1703, 1726, 1733	__draw_transform:nnnn
__draw_softpath_moveto:nn	1825, 1829, 1836
249, 1439, 1462	__draw_transform_invert:n
__draw_softpath_moveto_op:nn	1857, 1861, 1874, 1885
1465, 1487, 1499, 1547, 1705	__draw_transform_rotate:n
\l__draw_softpath_part_tl	1927, 1928, 1929, 1935
1508, 1523,	__draw_transform_rotate:nn
1552, 1555, 1557, 1591, 1646, 1735	1927, 1931, 1936, 1938
__draw_softpath_rectangle:nnnn	__draw_transform_shift:nn
573, 1439, 1472	1825, 1850, 1852
__draw_softpath_rectangle_-	__draw_transform_shift_absolute:nn
opi:nn 1476, 1487, 1502	1799, 1818, 1820
__draw_softpath_rectangle_-	__draw_transform_triangle:nnnnnn
opi:nnNnn 1487, 1503, 1504	1896, 1901
__draw_softpath_rectangle_-	__draw_vec:nn
opii:nn 1477, 1487, 1506	1136, 1137, 1139, 1141, 1142
__draw_softpath_round_action:nn	__draw_vec:nnn 1136, 1144, 1146
1517, 1541, 1561	\g__draw_xmax_dim 196,
__draw_softpath_round_action:Nnn	197, 1256, 1266, 1302, 1309, 1388, 1396
1517, 1569, 1571	\l__draw_xmax_dim . . . 1381, 1388, 1396
__draw_softpath_round_action_-	\g__draw_xmin_dim
close: 1517, 1579, 1597	198, 199, 1256, 1267, 1300,
__draw_softpath_round_action_-	1303, 1309, 1318, 1331, 1389, 1397
curveto:NnnNnn . . 1517, 1576, 1588	\l__draw_xmin_dim . . . 1381, 1389, 1397
__draw_softpath_round_calc:NnnNnn	\l__draw_xshift_dim 50, 1211,
1517, 1617, 1632, 1638, 1701	1225, 1779, 1794, 1822, 1854, 1888
__draw_softpath_round_calc:nnnnnn	\l__draw_xvec_x_dim
1517, 1649, 1666, 1671	1130, 1160, 1174, 1192
__draw_softpath_round_calc:nnnnw	\l__draw_xvec_y_dim . . 1130, 1161, 1178
1517, 1668, 1672	\g__draw_ymax_dim . . . 200, 201, 1256,
__draw_softpath_round_close:nn	1268, 1304, 1325, 1343, 1390, 1398
1517, 1603, 1697	\l__draw_ymax_dim . . . 1381, 1390, 1398
__draw_softpath_round_close:w	\g__draw_ymin_dim
1517, 1707, 1718, 1729	202, 203, 1256, 1269,
__draw_softpath_round_corners:	1305, 1320, 1325, 1339, 1391, 1399
678, 1517, 1517	\l__draw_ymin_dim . . . 1381, 1391, 1399
__draw_softpath_round_end:	\l__draw_yshift_dim 51, 1217,
1517, 1539, 1730	1225, 1779, 1795, 1823, 1855, 1889
__draw_softpath_round_lookahead:NnnNnn	\l__draw_yvec_x_dim . . 1130, 1160, 1175
1517, 1582, 1595, 1608	\l__draw_yvec_y_dim
__draw_softpath_round_loop:Nnn	1130, 1161, 1179, 1193
1517, 1529, 1537, 1558, 1568,	\l__draw_zvec_x_dim 1130, 1176
1583, 1606, 1618, 1624, 1633, 1715	\l__draw_zvec_y_dim 1130, 1180
__draw_softpath_round_roundpoint:NnnNnnNnn	
1517, 1623, 1629	
__draw_softpath_round_roundpoint:nn	
235, 1439, 1480, 1486	
	E
	\end 169, 786

exp commands:		<code>\group_end:</code> 58, 70, 117, 120, 489, 781, 1312, 1378, 1400, 1533, 1762
<code>\exp_after:wN</code>		
446, 464, 1529, 1603, 1706, 1717, 1726		
<code>\exp_args:Ne</code>	1766	
<code>\exp_args:Nf</code>	794, 860	
<code>\exp_args:Nff</code>	804	
<code>\exp_args:Nfff</code>	815	
<code>\exp_args:Nffff</code>	828	
<code>\exp_args:NNNV</code>	1295	
<code>\exp_not:N</code>		
1648, 1676, 1685, 1694, 1703, 1706, 1707, 1708, 1713, 1717, 1718, 1719		
<code>\exp_not:n</code>	1358	
F		
fp commands:		
<code>\fp_compare:nNnTF</code>	360, 370, 866	
<code>\fp_compare_p:nNn</code>		
.	1807, 1808, 1809, 1810	
<code>\fp_const:Nn</code>		
.	343, 344, 482, 483, 553, 1516	
<code>\fp_eval:n</code>	352, 353, 374, 381, 390, 844, 852, 861, 882, 883, 884, 885, 886, 887, 907, 912, 913, 920, 927, 928, 935, 942, 944, 963, 968, 969, 970, 977, 985, 998, 1003, 1021, 1037, 1042, 1049, 1050, 1069, 1076, 1098, 1099, 1118, 1119, 1120, 1121, 1155, 1168, 1187, 1766, 1830, 1831, 1832, 1833, 1863, 1928, 1932, 1933	
<code>\fp_new:N</code>	176, 177, 480, 481, 1510, 1511, 1779, 1780, 1781, 1782	
<code>\fp_set:Nn</code>	311, 312, 366, 367, 447, 448, 1545, 1546, 1593, 1594, 1662, 1663, 1787, 1790, 1801, 1802, 1803, 1804, 1876, 1878, 1880, 1882	
<code>\fp_to_decimal:N</code>	373, 380, 388	
<code>\fp_to_dim:n</code>	318, 325, 332, 336, 354, 355, 401, 410, 478, 504, 516, 517, 518, 519, 520, 521, 526, 527, 528, 529, 530, 531, 536, 537, 538, 539, 540, 541, 546, 547, 548, 549, 550, 551, 619, 620, 1350, 1678, 1682, 1687, 1691, 1747, 1755, 1760	
<code>\fp_use:N</code>	42, 43, 44, 45, 553	
<code>\fp_while_do:nNnn</code>	368	
<code>\fp_zero:N</code>	1524, 1525, 1788, 1789	
<code>\c_one_fp</code>	1807, 1810	
<code>\c_zero_fp</code>	866, 1808, 1809	
G		
group commands:		
<code>\group_begin:</code>	36, 62, 92, 103, 486, 750, 1273, 1370, 1387, 1521, 1752	
<code>\group_end:</code>	58, 70, 117, 120, 489, 781, 1312, 1378, 1400, 1533, 1762	
H		
hbox commands:		
<code>\hbox_gset:Nw</code>	101	
<code>\hbox_gset_end:</code>	118	
<code>\hbox_set:Nn</code>	37, 48, 63, 1316, 1329	
<code>\hbox_set:Nw</code>	1275, 1290	
<code>\hbox_set_end:</code>	1295, 1299	
I		
int commands:		
<code>\int_gincr:N</code>	1274	
<code>\int_if_odd:nTF</code>	943, 978	
<code>\int_new:N</code>	1263	
K		
kernel internal commands:		
<code>__kernel_kern:n</code>	50	
<code>__kernel_quark_new_test:N</code>	9	
M		
mode commands:		
<code>\mode_leave_vertical:</code>	1310	
msg commands:		
<code>\msg_error:nnn</code>	78, 109, 110, 675	
<code>\msg_new:nnn</code>	164	
<code>\msg_new:nnnn</code>	161, 166, 783	
P		
<code>\pgfextractx</code>	21	
<code>\pgfextracty</code>	21	
<code>\pgfgetlastxy</code>	21	
<code>\pgfgettransform</code>	50	
<code>\pgfgettransformentries</code>	50	
<code>\pgfinnerlinewidth</code>	48	
<code>\pgflowlevel</code>	50	
<code>\pgflowlevelsynccm</code>	50	
<code>\pgfpatharcto</code>	6	
<code>\pgfpatharctoprecomputed</code>	6	
<code>\pgfpathcosine</code>	6	
<code>\pgfpathcurvebetweentime</code>	6	
<code>\pgfpathcurvebetweentimecontinue</code>	6	
<code>\pgfpathparabola</code>	6	
<code>\pgfpathsine</code>	6	
<code>\pgfpointadd</code>	20	
<code>\pgfpointborderellipse</code>	21	
<code>\pgfpointborderrectangle</code>	21	
<code>\pgfpointcylindrical</code>	21	
<code>\pgfpointdiff</code>	20	
<code>\pgfpointorigin</code>	20	
<code>\pgfpointscale</code>	20	
<code>\pgfpointspherical</code>	21	
<code>\pgfqpoint</code>	21	

