

File I

Implementation

1 l3draw implementation

```

1 <*package>
2 <@@=draw>
3 \ProvidesExplPackage{l3draw}{2023-03-30}{}
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 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 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 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 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:x
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 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 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 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 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 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 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 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 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 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 \</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 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 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 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 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 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 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 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 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 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 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 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 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:x
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 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:x </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:x
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:nnnnnnn #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:nnnnnnn \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:Nx \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 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   \draw_transform_triangle:nnn { 0cm , 0cm } {#3} {#4}
487   \draw_path_arc:nnn {#1} {#2} { 1pt }
488 }

```

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

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

```

489 \cs_new_protected:Npn \draw_path_ellipse:nnn #1#2#3
490 {
491   \__draw_point_process:nnnn
492   { \__draw_path_ellipse:nnnnnn }
493   { \draw_point_transform:n {#1} }
494   { \__draw_point_transform_noshift:n {#2} }
495   { \__draw_point_transform_noshift:n {#3} }
496 }
497 \cs_new_protected:Npn \__draw_path_ellipse:nnnnnn #1#2#3#4#5#6
498 {
499   \use:x
500   {
501     \__draw_path_moveto:nn
502     { \fp_to_dim:n { #1 + #3 } } { \fp_to_dim:n { #2 + #4 } }
503     \__draw_path_ellipse_arci:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
504     \__draw_path_ellipse_arcii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
505     \__draw_path_ellipse_arciiii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
506     \__draw_path_ellipse_arciv:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
507   }
508   \__draw_softpath_closepath:
509   \__draw_path_moveto:nn {#1} {#2}
510 }
511 \cs_new:Npn \__draw_path_ellipse_arci:nnnnnn #1#2#3#4#5#6
512 {
513   \__draw_path_curveto:nnnnnn
514   { \fp_to_dim:n { #1 + #3 + #5 * \c__draw_path_ellipse_fp } }
515   { \fp_to_dim:n { #2 + #4 + #6 * \c__draw_path_ellipse_fp } }
516   { \fp_to_dim:n { #1 + #3 * \c__draw_path_ellipse_fp + #5 } }
517   { \fp_to_dim:n { #2 + #4 * \c__draw_path_ellipse_fp + #6 } }
518   { \fp_to_dim:n { #1 + #5 } }
519   { \fp_to_dim:n { #2 + #6 } }
520 }
521 \cs_new:Npn \__draw_path_ellipse_arcii:nnnnnn #1#2#3#4#5#6
522 {
523   \__draw_path_curveto:nnnnnn
524   { \fp_to_dim:n { #1 - #3 * \c__draw_path_ellipse_fp + #5 } }
525   { \fp_to_dim:n { #2 - #4 * \c__draw_path_ellipse_fp + #6 } }
526   { \fp_to_dim:n { #1 - #3 + #5 * \c__draw_path_ellipse_fp } }
527   { \fp_to_dim:n { #2 - #4 + #6 * \c__draw_path_ellipse_fp } }
528   { \fp_to_dim:n { #1 - #3 } }
529   { \fp_to_dim:n { #2 - #4 } }
530 }
531 \cs_new:Npn \__draw_path_ellipse_arciiii:nnnnnn #1#2#3#4#5#6
532 {
533   \__draw_path_curveto:nnnnnn
534   { \fp_to_dim:n { #1 - #3 - #5 * \c__draw_path_ellipse_fp } }
535   { \fp_to_dim:n { #2 - #4 - #6 * \c__draw_path_ellipse_fp } }
536   { \fp_to_dim:n { #1 - #3 * \c__draw_path_ellipse_fp - #5 } }
537   { \fp_to_dim:n { #2 - #4 * \c__draw_path_ellipse_fp - #6 } }
538   { \fp_to_dim:n { #1 - #5 } }

```

```

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

```

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

\draw_path_circle:nn A shortcut.

```

552   \cs_new_protected:Npn \draw_path_circle:nn #1#2
553   { \draw_path_ellipse:nnn {#1} { #2 , Opt } { Opt , #2 } }

```

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

4.6 Rectangles

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

__draw_path_rectangle:nnnn

_draw_path_rectangle_rounded:nnnn

```

554   \cs_new_protected:Npn \draw_path_rectangle:nn #1#2
555   {
556     \__draw_point_process:nnn
557     {
558       \bool_lazy_or:nnTF
559       { \l__draw_corner_arc_bool }
560       { \l__draw_matrix_active_bool }
561       { \__draw_path_rectangle_rounded:nnnn }
562       { \__draw_path_rectangle:nnnn }
563     }
564     { \draw_point_transform:n {#1} }
565     {#2}
566   }
567   \cs_new_protected:Npn \__draw_path_rectangle:nnnn #1#2#3#4
568   {
569     \__draw_path_update_limits:nn {#1} {#2}
570     \__draw_path_update_limits:nn { #1 + #3 } { #2 + #4 }
571     \__draw_softpath_rectangle:nnnn {#1} {#2} {#3} {#4}
572     \__draw_path_update_last:nn {#1} {#2}
573   }
574   \cs_new_protected:Npn \__draw_path_rectangle_rounded:nnnn #1#2#3#4
575   {
576     \draw_path_moveto:n { #1 + #3 , #2 + #4 }
577     \draw_path_lineto:n { #1 , #2 + #4 }
578     \draw_path_lineto:n { #1 , #2 }
579     \draw_path_lineto:n { #1 + #3 , #2 }
580     \draw_path_close:
581     \draw_path_moveto:n { #1 , #2 }
582   }

```

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

`\draw_path_rectangle_corners:nn`
`__draw_path_rectangle_corners:nnnn`

Another shortcut wrapper.

```
583 \cs_new_protected:Npn \draw_path_rectangle_corners:nn #1#2
584 {
585   \__draw_point_process:nnn
586   { \__draw_path_rectangle_corners:nnnnn {#1} }
587   {#1} {#2}
588 }
589 \cs_new_protected:Npn \__draw_path_rectangle_corners:nnnnn #1#2#3#4#5
590 { \draw_path_rectangle:nn {#1} { #4 - #2 , #5 - #3 } }
```

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

4.7 Grids

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

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

```
591 \cs_new_protected:Npn \draw_path_grid:nnnn #1#2#3#4
592 {
593   \__draw_point_process:nnn
594   {
595     \__draw_path_grid_auxi:ffnnnn
596     { \dim_eval:n { \dim_abs:n {#1} } }
597     { \dim_eval:n { \dim_abs:n {#2} } }
598   }
599   {#3} {#4}
600 }
601 \cs_new_protected:Npn \__draw_path_grid_auxi:nnnnnn #1#2#3#4#5#6
602 {
603   \dim_compare:nNnTF {#3} > {#5}
604   { \__draw_path_grid_auxii:nnnnnn {#1} {#2} {#5} {#4} {#3} {#6} }
605   { \__draw_path_grid_auxii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6} }
606 }
607 \cs_generate_variant:Nn \__draw_path_grid_auxi:nnnnnn { ff }
608 \cs_new_protected:Npn \__draw_path_grid_auxii:nnnnnn #1#2#3#4#5#6
609 {
610   \dim_compare:nNnTF {#4} > {#6}
611   { \__draw_path_grid_auxiii:nnnnnn {#1} {#2} {#3} {#6} {#5} {#4} }
612   { \__draw_path_grid_auxiii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6} }
613 }
614 \cs_new_protected:Npn \__draw_path_grid_auxiii:nnnnnn #1#2#3#4#5#6
615 {
616   \__draw_path_grid_auxiv:ffnnnnnn
617   { \fp_to_dim:n { #1 * trunc(#3/(#1)) } }
618   { \fp_to_dim:n { #2 * trunc(#4/(#2)) } }
619   {#1} {#2} {#3} {#4} {#5} {#6}
620 }
621 \cs_new_protected:Npn \__draw_path_grid_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
622 {
623   \dim_step_inline:nnnn
624   {#1}
```



```

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

```

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

4.8 Using paths

Actions to pass to the driver.

```

\l__draw_path_use_clip_bool
\l__draw_path_use_fill_bool
\l__draw_path_use_stroke_bool

```

```

641 \bool_new:N \l__draw_path_use_clip_bool
642 \bool_new:N \l__draw_path_use_fill_bool
643 \bool_new:N \l__draw_path_use_stroke_bool

```

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

Actions handled at the macro layer.

```

\l__draw_path_use_bb_bool
\l__draw_path_use_clear_bool

```

```

644 \bool_new:N \l__draw_path_use_bb_bool
645 \bool_new:N \l__draw_path_use_clear_bool

```

(End definition for \l__draw_path_use_bb_bool and \l__draw_path_use_clear_bool.)

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

```

\draw_path_use:n
\draw_path_use_clear:n
\__draw_path_use:n
\__draw_path_use_action_draw:
\__draw_path_use_action_fillstroke:
\__draw_path_use_stroke_bb:
\__draw_path_use_stroke_bb_aux:NnN

```

```

646 \cs_new_protected:Npn \draw_path_use:n #1
647 {
648     \tl_if_blank:nF {#1}
649     { \__draw_path_use:n {#1} }
650 }
651 \cs_new_protected:Npn \draw_path_use_clear:n #1
652 {
653     \bool_lazy_or:nnTF
654     { \tl_if_blank_p:n {#1} }
655     { \str_if_eq_p:nn {#1} { clear } }
656     {
657         \__draw_softpath_clear:
658         \__draw_path_reset_limits:
659     }
660     { \__draw_path_use:n { #1 , clear } }
661 }

```

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).

```

662 \cs_new_protected:Npn \__draw_path_use:n #1
663 {
664   \bool_set_false:N \l__draw_path_use_clip_bool
665   \bool_set_false:N \l__draw_path_use_fill_bool
666   \bool_set_false:N \l__draw_path_use_stroke_bool
667   \clist_map_inline:nn {#1}
668   {
669     \cs_if_exist:cTF { l__draw_path_use_ ##1 _ bool }
670     { \bool_set_true:c { l__draw_path_use_ ##1 _ bool } }
671     {
672       \cs_if_exist_use:cF { __draw_path_use_action_ ##1 : }
673       { \msg_error:nnn { draw } { invalid-path-action } {##1} }
674     }
675   }
676   \__draw_softpath_round_corners:
677   \bool_lazy_and:nnT
678   { \l_draw_bb_update_bool }
679   { \l__draw_path_use_stroke_bool }
680   { \__draw_path_use_stroke_bb: }
681   \__draw_softpath_use:
682   \bool_if:NT \l__draw_path_use_clip_bool
683   {
684     \__draw_backend_clip:
685     \bool_set_false:N \l_draw_bb_update_bool
686     \bool_lazy_or:nnF
687     { \l__draw_path_use_fill_bool }
688     { \l__draw_path_use_stroke_bool }
689     { \__draw_backend_discardpath: }
690   }
691   \bool_lazy_or:nnT
692   { \l__draw_path_use_fill_bool }
693   { \l__draw_path_use_stroke_bool }
694   {
695     \use:c
696     {
697       __draw_backend_
698       \bool_if:NT \l__draw_path_use_fill_bool { fill }
699       \bool_if:NT \l__draw_path_use_stroke_bool { stroke }
700       :
701     }
702   }
703   \bool_if:NT \l__draw_path_use_clear_bool
704   { \__draw_softpath_clear: }
705 }
706 \cs_new_protected:Npn \__draw_path_use_action_draw:
707 {
708   \bool_set_true:N \l__draw_path_use_stroke_bool
709 }
710 \cs_new_protected:Npn \__draw_path_use_action_fillstroke:
711 {
712   \bool_set_true:N \l__draw_path_use_fill_bool

```

```

713 \bool_set_true:N \l__draw_path_use_stroke_bool
714 }

```

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.

```

715 \cs_new_protected:Npn \__draw_path_use_stroke_bb:
716 {
717   \__draw_path_use_stroke_bb_aux:NnN x { max } +
718   \__draw_path_use_stroke_bb_aux:NnN y { max } +
719   \__draw_path_use_stroke_bb_aux:NnN x { min } -
720   \__draw_path_use_stroke_bb_aux:NnN y { min } -
721 }
722 \cs_new_protected:Npn \__draw_path_use_stroke_bb_aux:NnN #1#2#3
723 {
724   \dim_compare:nNnF { \dim_use:c { g__draw_ #1#2 _dim } } = { #3 -\c_max_dim }
725   {
726     \dim_gset:cn { g__draw_ #1#2 _dim }
727     {
728       \use:c { dim_ #2 :nn }
729       { \dim_use:c { g__draw_ #1#2 _dim } }
730       {
731         \dim_use:c { g__draw_path_ #1#2 _dim }
732         #3 0.5 \g__draw_linewidth_dim
733       }
734     }
735   }
736 }

```

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

4.9 Scoping paths

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

```

\l__draw_path_xmax_dim
\l__draw_path_xmin_dim
\l__draw_path_ymax_dim
\l__draw_path_ymin_dim
\l__draw_softpath_corners_bool
737 \dim_new:N \l__draw_path_lastx_dim
738 \dim_new:N \l__draw_path_lasty_dim
739 \dim_new:N \l__draw_path_xmax_dim
740 \dim_new:N \l__draw_path_xmin_dim
741 \dim_new:N \l__draw_path_ymax_dim
742 \dim_new:N \l__draw_path_ymin_dim
743 \dim_new:N \l__draw_softpath_lastx_dim
744 \dim_new:N \l__draw_softpath_lasty_dim
745 \bool_new:N \l__draw_softpath_corners_bool

```

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

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

```

746 \cs_new_protected:Npn \draw_path_scope_begin:
747 {
748   \group_begin:
749   \dim_set_eq:NN \l__draw_path_lastx_dim \g__draw_path_lastx_dim
750   \dim_set_eq:NN \l__draw_path_lasty_dim \g__draw_path_lasty_dim

```

```

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

```

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

```

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

```

5 l3draw-points implementation

```

788 <*package>
789 <@@=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
790 \cs_new:Npn \__draw_point_process:nn #1#2
791 {
792   \exp_args:Nf \__draw_point_process_auxi:nn
793   { \draw_point:n {#2} }
794   {#1}
795 }
796 \cs_new:Npn \__draw_point_process_auxi:nn #1#2
797 { \__draw_point_process_auxii:nw {#2} #1 \s__draw_stop }
798 \cs_new:Npn \__draw_point_process_auxii:nw #1 #2 , #3 \s__draw_stop
799 { #1 {#2} {#3} }
800 \cs_new:Npn \__draw_point_process:nnn #1#2#3
801 {
802   \exp_args:Nff \__draw_point_process_auxiii:nnn
803   { \draw_point:n {#2} }
804   { \draw_point:n {#3} }
805   {#1}
806 }
807 \cs_new:Npn \__draw_point_process_auxiii:nnn #1#2#3
808 { \__draw_point_process_auxiv:nw {#3} #1 \s__draw_mark #2 \s__draw_stop }
809 \cs_new:Npn \__draw_point_process_auxiv:nw #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_stop
810 { #1 {#2} {#3} {#4} {#5} }
811 \cs_new:Npn \__draw_point_process:nnnn #1#2#3#4
812 {
813   \exp_args:Nfff \__draw_point_process_auxv:nnnn
814   { \draw_point:n {#2} }
815   { \draw_point:n {#3} }
816   { \draw_point:n {#4} }
817   {#1}
818 }

```

```

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

```

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

5.2 Basic points

<code>\draw_point:n</code> <code>__draw_point_to_dim:n</code> <code>__draw_point_to_dim:f</code> <code>__draw_point_to_dim:w</code>	Co-ordinates are always returned as two dimensions. <pre> 841 \cs_new:Npn \draw_point:n #1 842 { __draw_point_to_dim:f { \fp_eval:n {#1} } } 843 \cs_new:Npn __draw_point_to_dim:n #1 844 { __draw_point_to_dim:w #1 } 845 \cs_generate_variant:Nn __draw_point_to_dim:n { f } 846 \cs_new:Npn __draw_point_to_dim:w (#1 , ~ #2) { #1pt , #2pt } </pre>
---	---

5.3 Polar co-ordinates

<code>\draw_point_polar:nn</code> <code>\draw_point_polar:nnn</code> <code>__draw_draw_polar:nnn</code> <code>__draw_draw_polar:fnn</code>	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. <pre> 847 \cs_new:Npn \draw_point_polar:nn #1#2 848 { \draw_point_polar:nnn {#1} {#1} {#2} } 849 \cs_new:Npn \draw_point_polar:nnn #1#2#3 850 { __draw_draw_polar:fnn { \fp_eval:n {#3} } {#1} {#2} } 851 \cs_new:Npn __draw_draw_polar:nnn #1#2#3 852 { \draw_point:n { cosd(#1) * (#2) , sind(#1) * (#3) } } 853 \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.

```

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

```

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.)

```

871 \cs_new:Npn \draw_point_intersect_lines:nnnn #1#2#3#4
872 {
873   \__draw_point_process:nnnnn
874   { \__draw_point_intersect_lines:nnnnnnnn }
875   {#1} {#2} {#3} {#4}
876 }

```

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.

```
877 \cs_new:Npn \__draw_point_intersect_lines:nnnnnnnn #1#2#3#4#5#6#7#8
878 {
879   \__draw_point_intersect_lines_aux:ffffff
880   { \fp_eval:n { #1 * #4 - #2 * #3 } }
881   { \fp_eval:n { #5 * #8 - #6 * #7 } }
882   { \fp_eval:n { #1 - #3 } }
883   { \fp_eval:n { #5 - #7 } }
884   { \fp_eval:n { #2 - #4 } }
885   { \fp_eval:n { #6 - #8 } }
886 }
887 \cs_new:Npn \__draw_point_intersect_lines_aux:nnnnnn #1#2#3#4#5#6
888 {
889   \draw_point:n
890   {
891     ( #2 * #3 - #1 * #4 , #2 * #5 - #1 * #6 )
892     / ( #4 * #5 - #6 * #3 )
893   }
894 }
895 \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:nnnnnnnnnn
draw_point_intersect_circles_auxvi:fnnnnnnnnn
draw_point_intersect_circles_auxvii:nnnnnnnn
draw_point_intersect_circles_auxvii:ffnnnnnn
```

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.

```

896 \cs_new:Npn \draw_point_intersect_circles:nnnnn #1#2#3#4#5
897 {
898   \__draw_point_process:nnn
899   { \__draw_point_intersect_circles_auxi:nnnnnnn {#2} {#4} {#5} }
900   {#1} {#3}
901 }
902 \cs_new:Npn \__draw_point_intersect_circles_auxi:nnnnnnn #1#2#3#4#5#6#7
903 {
904   \__draw_point_intersect_circles_auxii:ffnnnnnn
905   { \fp_eval:n {#1} } { \fp_eval:n {#2} } {#4} {#5} {#6} {#7} {#3}
906 }

```

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.

```

907 \cs_new:Npn \__draw_point_intersect_circles_auxii:nnnnnnn #1#2#3#4#5#6#7
908 {
909   \__draw_point_intersect_circles_auxiii:ffnnnnnn
910   { \fp_eval:n { #5 - #3 } }
911   { \fp_eval:n { #6 - #4 } }
912   {#1} {#2} {#3} {#4} {#7}
913 }
914 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxii:nnnnnnn { ff }
915 \cs_new:Npn \__draw_point_intersect_circles_auxiii:nnnnnnn #1#2#3#4#5#6#7
916 {
917   \__draw_point_intersect_circles_auxiv:fnnnnnnn
918   { \fp_eval:n { sqrt( #1 * #1 + #2 * #2 ) } }
919   {#1} {#2} {#3} {#4} {#5} {#6} {#7}
920 }
921 \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.

```

922 \cs_new:Npn \__draw_point_intersect_circles_auxiv:fnnnnnnn #1#2#3#4#5#6#7#8
923 {
924   \__draw_point_intersect_circles_auxv:ffnnnnnnnn
925   { \fp_eval:n { 1 / #1 } }
926   { \fp_eval:n { #4 * #4 } }
927   {#1} {#2} {#3} {#5} {#6} {#7} {#8}

```

```

928 }
929 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxiv:nnnnnnnn { f }
930 \cs_new:Npn \__draw_point_intersect_circles_auxv:nnnnnnnn #1#2#3#4#5#6#7#8#9
931 {
932   \__draw_point_intersect_circles_auxvi:fnnnnnnnn
933   { \fp_eval:n { 0.5 * #1 * ( #2 + #3 * #3 - #6 * #6 ) } }
934   {#1} {#2} {#4} {#5} {#7} {#8} {#9}
935 }
936 \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.

```

937 \cs_new:Npn \__draw_point_intersect_circles_auxvi:nnnnnnnn #1#2#3#4#5#6#7#8
938 {
939   \__draw_point_intersect_circles_auxvii:ffnnnnnn
940   { \fp_eval:n { #1 * #2 } }
941   { \int_if_odd:nTF {#8} { 1 } { -1 } }
942   { \fp_eval:n { sqrt ( #3 - #1 * #1 ) * #2 } }
943   {#4} {#5} {#6} {#7}
944 }
945 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxvi:nnnnnnnn { f }
946 \cs_new:Npn \__draw_point_intersect_circles_auxvii:nnnnnnnn #1#2#3#4#5#6#7
947 {
948   \draw_point:n
949   { #6 + #4 * #1 + #2 * #3 * #5 , #7 + #5 * #1 + -1 * #2 * #3 * #4 }
950 }
951 \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.

```

952 \cs_new:Npn \draw_point_intersect_line_circle:nnnnn #1#2#3#4#5
953 {
954   \__draw_point_process:nnnn
955   { \__draw_point_intersect_line_circle_auxi:nnnnnnnn {#4} {#5} }
956   {#1} {#2} {#3}
957 }
958 \cs_new:Npn \__draw_point_intersect_line_circle_auxi:nnnnnnnn #1#2#3#4#5#6#7#8
959 {
960   \__draw_point_intersect_line_circle_auxii:fnnnnnnnn
961   { \fp_eval:n {#1} } {#3} {#4} {#5} {#6} {#7} {#8} {#2}
962 }

```

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.

```

963 \cs_new:Npn \__draw_point_intersect_line_circle_auxii:nnnnnnnn #1#2#3#4#5#6#7#8
964 {
965   \__draw_point_intersect_line_circle_auxiii:ffnnnnnn
966   { \fp_eval:n { (#4-#2)*(#4-#2)+(#5-#3)*(#5-#3) } }
967   { \fp_eval:n { 2*((#4-#2)*(#2-#6)+(#5-#3)*(#3-#7)) } }
968   { \fp_eval:n { (#6*#6+#7*#7)+(#2*#2+#3*#3)-(2*(#6*#2+#7*#3))-(#1*#1) } }
969   {#2} {#3} {#4} {#5} {#6} {#7} {#8}
970 }
971 \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 .

```

972 \cs_new:Npn \__draw_point_intersect_line_circle_auxiii:nnnnnnnn #1#2#3#4#5#6#7#8
973 {
974   \__draw_point_intersect_line_circle_auxiv:ffnnnnnn
975   { \fp_eval:n { #2 * #2 - 4 * #1 * #3 } }
976   { \int_if_odd:nTF {#8} { 1 } { -1 } }
977   {#1} {#2} {#4} {#5} {#6} {#7}
978 }
979 \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.

```

980 \cs_new:Npn \__draw_point_intersect_line_circle_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
981 {
982   \__draw_point_intersect_line_circle_auxv:fnnnn
983   { \fp_eval:n { (-1 * #4 + #2 * sqrt(#1)) / (2 * #3) } }
984   {#5} {#6} {#7} {#8}
985 }
986 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxiv:nnnnnnnn { ff }
987 \cs_new:Npn \__draw_point_intersect_line_circle_auxv:nnnnn #1#2#3#4#5
988 {
989   \draw_point:n
990   { #2 + #1 * (#4 - #2), #3 + #1 * (#5 - #3) }
991 }
992 \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 993 \cs_new:Npn \draw_point_interpolate_line:nnn #1#2#3
\_draw_point_interpolate_line_aux:nnnnn 994 {
\_draw_point_interpolate_line_aux:fnnnn 995 \__draw_point_process:nnn
\_draw_point_interpolate_line_aux:nnnnnn 996 { \__draw_point_interpolate_line_aux:fnnnn { \fp_eval:n {#1} } }
\_draw_point_interpolate_line_aux:fnnnnnn 997 {#2} {#3}
998 }
999 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnnn #1#2#3#4#5
1000 {
1001 \__draw_point_interpolate_line_aux:fnnnnnn { \fp_eval:n { 1 - #1 } }
1002 {#1} {#2} {#3} {#4} {#5}
1003 }
1004 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnnn { f }
1005 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnnn #1#2#3#4#5#6
1006 { \draw_point:n { #2 * #3 + #1 * #5 , #2 * #4 + #1 * #6 } }
1007 \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 1008 \cs_new:Npn \draw_point_interpolate_distance:nnn #1#2#3
\_draw_point_interpolate_distance:nnnnnn 1009 {
\_draw_point_interpolate_distance:nnnnnn 1010 \__draw_point_process:nn
\_draw_point_interpolate_distance:fnnnnnn 1011 { \__draw_point_interpolate_distance:nnnn {#1} {#3} }
1012 {#2}
1013 }
1014 \cs_new:Npn \__draw_point_interpolate_distance:nnnn #1#2#3#4
1015 {
1016 \__draw_point_process:nn
1017 {
1018 \__draw_point_interpolate_distance:fnnnnn
1019 { \fp_eval:n {#1} } {#3} {#4}
1020 }
1021 { \draw_point_unit_vector:n { ( #2 ) - ( #3 , #4 ) } }
1022 }
1023 \cs_new:Npn \__draw_point_interpolate_distance:nnnnnn #1#2#3#4#5
1024 { \draw_point:n { #2 + #1 * #4 , #3 + #1 * #5 } }
1025 \cs_generate_variant:Nn \__draw_point_interpolate_distance:nnnnnn { f }

```

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

```

\draw_point_interpolate_arcaxes:nnnnnn 2026 \cs_new:Npn \draw_point_interpolate_arcaxes:nnnnnn #1#2#3#4#5#6
draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn 2027 {
draw_point_interpolate_arcaxes_auxii:nnnnnnnnnn 2028 \__draw_point_process:nnnn
draw_point_interpolate_arcaxes_auxiii:nnnnnnnnnn 2029 { \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn {#1} {#5} {#6} }
draw_point_interpolate_arcaxes_auxiv:nnnnnnnnnn 2030 {#2} {#3} {#4}
draw_point_interpolate_arcaxes_auxiv:fnnnnnnnnnn 2031 }
2032 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
2033 {
2034 \__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.

```

1035     { \fp_eval:n {#1} } {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1036   }

```

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.

```

1037 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnn #1#2#3#4#5#6#7#8#9
1038   {
1039     \__draw_point_interpolate_arcaxes_auxiii:fnnnnnnn
1040     { \fp_eval:n { #1 * (#3) + ( 1 - #1 ) * (#2) } }
1041     {#4} {#5} {#6} {#7} {#8} {#9}
1042   }
1043 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnn { f }
1044 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnn #1#2#3#4#5#6#7
1045   {
1046     \__draw_point_interpolate_arcaxes_auxiv:ffnnnnnnn
1047     { \fp_eval:n { cosd (#1) } }
1048     { \fp_eval:n { sind (#1) } }
1049     {#2} {#3} {#4} {#5} {#6} {#7}
1050   }
1051 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnn { f }
1052 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnnn #1#2#3#4#5#6#7#8
1053   {
1054     \draw_point:n
1055     { #3 + #1 * #5 + #2 * #7 , #4 + #1 * #6 + #2 * #8 }
1056   }
1057 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnnn { ff }

```

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

```

\draw_point_interpolate_curve:nnnnn
\draw_point_interpolate_curve_auxi:nnnnnnnnn
\draw_point_interpolate_curve_auxii:nnnnnnnnn
\draw_point_interpolate_curve_auxiii:fnnnnnnnnn
\draw_point_interpolate_curve_auxiiii:nnnnnnn
\draw_point_interpolate_curve_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.

```

1058 \cs_new:Npn \draw_point_interpolate_curve:nnnnnn #1#2#3#4#5
1059 {
1060   \__draw_point_process:nnnnn
1061   { \__draw_point_interpolate_curve_auxi:nnnnnnnnn {#1} }
1062   {#2} {#3} {#4} {#5}
1063 }
1064 \cs_new:Npn \__draw_point_interpolate_curve_auxi:nnnnnnnnn #1#2#3#4#5#6#7#8#9
1065 {
1066   \__draw_point_interpolate_curve_auxii:fnnnnnnnn
1067   { \fp_eval:n {#1} }
1068   {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1069 }

```

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!

```

1070 \cs_new:Npn \__draw_point_interpolate_curve_auxii:nnnnnnnnn
1071   #1#2#3#4#5#6#7#8#9
1072 {
1073   \__draw_point_interpolate_curve_auxiii:fnnnnnn
1074   { \fp_eval:n { 1 - #1 } }
1075   {#1}
1076   { {#2} {#3} } { {#4} {#5} } { {#6} {#7} } { {#8} {#9} }
1077 }
1078 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxii:nnnnnnnnn { f }
1079 % \begin{macrocode}
1080 % We need to do the first cycle, but haven't got enough arguments to keep

```

```

1081 % everything in play at once. So her ewe use a but of argument re-ordering
1082 % and a single auxiliary to get the job done.
1083 % \begin{macrocode}
1084 \cs_new:Npn \__draw_point_interpolate_curve_auxiii:nnnnnn #1#2#3#4#5#6
1085 {
1086   \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #3 #4
1087   \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #4 #5
1088   \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #5 #6
1089   \prg_do_nothing:
1090   \__draw_point_interpolate_curve_auxvi:n { {#1} {#2} }
1091 }
1092 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxiii:nnnnnn { f }
1093 \cs_new:Npn \__draw_point_interpolate_curve_auxiv:nnnnnn #1#2#3#4#5#6
1094 {
1095   \__draw_point_interpolate_curve_auxv:ffw
1096   { \fp_eval:n { #1 * #3 + #2 * #5 } }
1097   { \fp_eval:n { #1 * #4 + #2 * #6 } }
1098 }
1099 \cs_new:Npn \__draw_point_interpolate_curve_auxv:nnw
1100 #1#2#3 \prg_do_nothing: #4#5
1101 {
1102   #3
1103   \prg_do_nothing:
1104   #4 { #5 {#1} {#2} }
1105 }
1106 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxv:nnw { ff }
1107 % \begin{macrocode}
1108 % Get the arguments back into the right places and to the second and
1109 % third cycles directly.
1110 % \begin{macrocode}
1111 \cs_new:Npn \__draw_point_interpolate_curve_auxvi:n #1
1112 { \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1 }
1113 \cs_new:Npn \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1#2#3#4#5#6#7#8
1114 {
1115   \__draw_point_interpolate_curve_auxviii:ffffnn
1116   { \fp_eval:n { #1 * #5 + #2 * #3 } }
1117   { \fp_eval:n { #1 * #6 + #2 * #4 } }
1118   { \fp_eval:n { #1 * #7 + #2 * #5 } }
1119   { \fp_eval:n { #1 * #8 + #2 * #6 } }
1120   {#1} {#2}
1121 }
1122 \cs_new:Npn \__draw_point_interpolate_curve_auxviii:nnnnnn #1#2#3#4#5#6
1123 {
1124   \draw_point:n
1125   { #5 * #3 + #6 * #1 , #5 * #4 + #6 * #2 }
1126 }
1127 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxviii:nnnnnn { ffff }

```

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

5.7 Vector support

As well as co-ordinates relative to the drawing


```

\l__draw_xvec_x_dim Base vectors to map to the underlying two-dimensional drawing space.
\l__draw_xvec_y_dim
\l__draw_yvec_x_dim
\l__draw_yvec_y_dim
\l__draw_zvec_x_dim
\l__draw_zvec_y_dim

1128 \dim_new:N \l__draw_xvec_x_dim
1129 \dim_new:N \l__draw_xvec_y_dim
1130 \dim_new:N \l__draw_yvec_x_dim
1131 \dim_new:N \l__draw_yvec_y_dim
1132 \dim_new:N \l__draw_zvec_x_dim
1133 \dim_new:N \l__draw_zvec_y_dim

(End definition for \l__draw_xvec_x_dim and others.)

\draw_xvec:n Calculate the underlying position and store it.
\draw_yvec:n
\draw_zvec:n
1134 \cs_new_protected:Npn \draw_xvec:n #1
1135 { \__draw_vec:nn { x } {#1} }
1136 \cs_new_protected:Npn \draw_yvec:n #1
1137 { \__draw_vec:nn { y } {#1} }
1138 \cs_new_protected:Npn \draw_zvec:n #1
1139 { \__draw_vec:nn { z } {#1} }
1140 \cs_new_protected:Npn \__draw_vec:nn #1#2
1141 {
1142   \__draw_point_process:nn { \__draw_vec:nnn {#1} } {#2}
1143 }
1144 \cs_new_protected:Npn \__draw_vec:nnn #1#2#3
1145 {
1146   \dim_set:cn { l__draw_ #1 vec_x_dim } {#2}
1147   \dim_set:cn { l__draw_ #1 vec_y_dim } {#3}
1148 }

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

Initialise the vectors.

1149 \draw_xvec:n { 1cm , 0cm }
1150 \draw_yvec:n { 0cm , 1cm }
1151 \draw_zvec:n { -0.385cm , -0.385cm }

\draw_point_vec:nn Force a single evaluation of each factor, then use these to work out the underlying point.
\__draw_point_vec:nn
\__draw_point_vec:ff
1152 \cs_new:Npn \draw_point_vec:nn #1#2
1153 { \__draw_point_vec:ff { \fp_eval:n {#1} } { \fp_eval:n {#2} } }
1154 \cs_new:Npn \__draw_point_vec:nn #1#2
1155 {
1156   \draw_point:n
1157   {
1158     #1 * \l__draw_xvec_x_dim + #2 * \l__draw_yvec_x_dim ,
1159     #1 * \l__draw_xvec_y_dim + #2 * \l__draw_yvec_y_dim
1160   }
1161 }
1162 \cs_generate_variant:Nn \__draw_point_vec:nn { ff }
1163 \cs_new:Npn \draw_point_vec:nnn #1#2#3
1164 {
1165   \__draw_point_vec:fff
1166   { \fp_eval:n {#1} } { \fp_eval:n {#2} } { \fp_eval:n {#3} }
1167 }
1168 \cs_new:Npn \__draw_point_vec:nnn #1#2#3
1169 {
1170   \draw_point:n
1171   {

```

```

1172         #1 * \l__draw_xvec_x_dim
1173       + #2 * \l__draw_yvec_x_dim
1174       + #3 * \l__draw_zvec_x_dim
1175     ,
1176     #1 * \l__draw_xvec_y_dim
1177   + #2 * \l__draw_yvec_y_dim
1178   + #3 * \l__draw_zvec_y_dim
1179   }
1180 }
1181 \cs_generate_variant:Nn \__draw_point_vec:nnn { fff }

```

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

`\draw_point_vec_polar:nn` Much the same as the core polar approach.
`\draw_point_vec_polar:nnn`
`__draw_point_vec_polar:nnn`
`__draw_point_vec_polar:fnn`

```

1182 \cs_new:Npn \draw_point_vec_polar:nn #1#2
1183 { \draw_point_vec_polar:nnn {#1} {#1} {#2} }
1184 \cs_new:Npn \draw_point_vec_polar:nnn #1#2#3
1185 { \__draw_draw_vec_polar:fnn { \fp_eval:n {#3} } {#1} {#2} }
1186 \cs_new:Npn \__draw_draw_vec_polar:nnn #1#2#3
1187 {
1188   \draw_point:n
1189   {
1190     cosd(#1) * (#2) * \l__draw_xvec_x_dim ,
1191     sind(#1) * (#3) * \l__draw_yvec_y_dim
1192   }
1193 }
1194 \cs_generate_variant:Nn \__draw_draw_vec_polar:nnn { f }

```

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

5.8 Transformations

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

```

1195 \cs_new:Npn \draw_point_transform:n #1
1196 {
1197   \__draw_point_process:nn
1198   { \__draw_point_transform:nn } {#1}
1199 }
1200 \cs_new:Npn \__draw_point_transform:nn #1#2
1201 {
1202   \bool_if:NTF \l__draw_matrix_active_bool
1203   {
1204     \draw_point:n
1205     {
1206       (
1207         \l__draw_matrix_a_fp * #1
1208         + \l__draw_matrix_c_fp * #2
1209         + \l__draw_xshift_dim
1210       )
1211       ,
1212       (
1213         \l__draw_matrix_b_fp * #1

```

```

1214         + \l__draw_matrix_d_fp * #2
1215         + \l__draw_yshift_dim
1216     )
1217 }
1218 }
1219 {
1220     \draw_point:n
1221     {
1222         (#1, #2)
1223         + ( \l__draw_xshift_dim , \l__draw_yshift_dim )
1224     }
1225 }
1226 }

```

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

`__draw_point_transform_noshift:n`
`__draw_point_transform_noshift:nn`

A version with no shift: used for internal purposes.

```

1227 \cs_new:Npn \__draw_point_transform_noshift:n #1
1228 {
1229     \__draw_point_process:nn
1230     { \__draw_point_transform_noshift:nn } {#1}
1231 }
1232 \cs_new:Npn \__draw_point_transform_noshift:nn #1#2
1233 {
1234     \bool_if:NTF \l__draw_matrix_active_bool
1235     {
1236         \draw_point:n
1237         {
1238             (
1239                 \l__draw_matrix_a_fp * #1
1240                 + \l__draw_matrix_c_fp * #2
1241             )
1242             ,
1243             (
1244                 \l__draw_matrix_b_fp * #1
1245                 + \l__draw_matrix_d_fp * #2
1246             )
1247         }
1248     }
1249     { \draw_point:n { (#1, #2) } }
1250 }

```

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

```
1251 \</package>
```

6 l3draw-scopes implementation

```

1252 <*package>
1253 <@@=draw>

```

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

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

6.1 Drawing environment

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

```
\g__draw_xmin_dim
\g__draw_ymax_dim
\g__draw_ymin_dim
1254 \dim_new:N \g__draw_xmax_dim
1255 \dim_new:N \g__draw_xmin_dim
1256 \dim_new:N \g__draw_ymax_dim
1257 \dim_new:N \g__draw_ymin_dim
```

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

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

```
1258 \bool_new:N \l_draw_bb_update_bool
```

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

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

```
1259 \box_new:N \l__draw_main_box
1260 \box_new:N \l__draw_layer_main_box
```

(End definition for `\l__draw_layer_main_box`.)

`\g__draw_id_int` The drawing number.

```
1261 \int_new:N \g__draw_id_int
```

(End definition for `\g__draw_id_int`.)

`__draw_reset_bb:` A simple auxiliary.

```
1262 \cs_new_protected:Npn \__draw_reset_bb:
1263 {
1264   \dim_gset:Nn \g__draw_xmax_dim { -\c_max_dim }
1265   \dim_gset:Nn \g__draw_xmin_dim { \c_max_dim }
1266   \dim_gset:Nn \g__draw_ymax_dim { -\c_max_dim }
1267   \dim_gset:Nn \g__draw_ymin_dim { \c_max_dim }
1268 }
```

(End definition for `__draw_reset_bb:.`)

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

```
1269 \cs_new_protected:Npn \draw_begin:
1270 {
1271   \group_begin:
1272     \int_gincr:N \g__draw_id_int
1273     \hbox_set:Nw \l__draw_main_box
1274       \__draw_backend_begin:
1275       \__draw_reset_bb:
```

```

1276     \__draw_path_reset_limits:
1277     \bool_set_true:N \l_draw_bb_update_bool
1278     \draw_transform_matrix_reset:
1279     \draw_transform_shift_reset:
1280     \__draw_softpath_clear:
1281     \draw_linewidth:n { \l_draw_default_linewidth_dim }
1282     \color_select:n { . }
1283     \draw_nonzero_rule:
1284     \draw_cap_but:
1285     \draw_join_miter:
1286     \draw_miterlimit:n { 10 }
1287     \draw_dash_pattern:nn { } { 0cm }
1288     \hbox_set:Nw \l__draw_layer_main_box
1289   }
1290 \cs_new_protected:Npn \draw_end:
1291 {
1292     \__draw_baseline_finalise:w
1293     \exp_args:NNNV \hbox_set_end:
1294     \clist_set:Nn \l_draw_layers_clist \l_draw_layers_clist
1295     \__draw_layers_insert:
1296     \__draw_backend_end:
1297     \hbox_set_end:
1298     \dim_compare:nNnT \g__draw_xmin_dim = \c_max_dim
1299     {
1300         \dim_gzero:N \g__draw_xmax_dim
1301         \dim_gzero:N \g__draw_xmin_dim
1302         \dim_gzero:N \g__draw_ymax_dim
1303         \dim_gzero:N \g__draw_ymin_dim
1304     }
1305     \__draw_finalise:
1306     \box_set_wd:Nn \l__draw_main_box
1307     { \g__draw_xmax_dim - \g__draw_xmin_dim }
1308     \mode_leave_vertical:
1309     \box_use_drop:N \l__draw_main_box
1310 \group_end:
1311 }

```

(End 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.

```

1312 \cs_new_protected:Npn \__draw_finalise:
1313 {
1314     \hbox_set:Nn \l__draw_main_box
1315     {
1316         \skip_horizontal:n { -\g__draw_xmin_dim }
1317         \box_move_down:nn
1318         { \g__draw_ymin_dim }
1319         { \box_use_drop:N \l__draw_main_box }
1320     }
1321     \box_set_dp:Nn \l__draw_main_box { Opt }

```

```

1322     \box_set_ht:Nn \l__draw_main_box
1323     { \g__draw_ymax_dim - \g__draw_ymin_dim }
1324   }
1325   \cs_new_protected:Npn \__draw_finalise_baseline:n #1
1326   {
1327     \hbox_set:Nn \l__draw_main_box
1328     {
1329       \skip_horizontal:n { -\g__draw_xmin_dim }
1330       \box_move_down:nn
1331       { #1 }
1332       { \box_use_drop:N \l__draw_main_box }
1333     }
1334     \box_set_dp:Nn \l__draw_main_box
1335     {
1336       \dim_max:nn
1337       { #1 - \g__draw_ymin_dim }
1338       { 0pt }
1339     }
1340     \box_set_ht:Nn \l__draw_main_box
1341     { \g__draw_ymax_dim + #1 }
1342   }

```

(End 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

```

1343 \bool_new:N \l__draw_baseline_bool
1344 \dim_new:N \l__draw_baseline_dim

```

(End 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.

```

1345 \cs_new_protected:Npn \draw_baseline:n #1
1346   {
1347     \bool_set_true:N \l__draw_baseline_bool
1348     \dim_set:Nn \l__draw_baseline_dim { \fp_to_dim:n {#1} }
1349   }

```

(End 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.

```

1350 \cs_new_protected:Npn \__draw_baseline_finalise:w #1 \__draw_finalise:
1351   {
1352     \bool_if:NTF \l__draw_baseline_bool
1353     {
1354       \use:x
1355       {
1356         \exp_not:n {#1}
1357         \__draw_finalise_baseline:n { \dim_use:N \l__draw_baseline_dim }
1358       }
1359     }
1360     { #1 \__draw_finalise: }
1361   }

```

(End definition for `_draw_baseline_finalise:w`.)

6.3 Scopes

`\l__draw_linewidth_dim` Storage for local variables.
`\l__draw_fill_color_tl` 1362 `\dim_new:N \l__draw_linewidth_dim`
`\l__draw_stroke_color_tl` 1363 `\tl_new:N \l__draw_fill_color_tl`
1364 `\tl_new:N \l__draw_stroke_color_tl`

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

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

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

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

`\l__draw_xmax_dim` Storage for the bounding box.
`\l__draw_xmin_dim` 1379 `\dim_new:N \l__draw_xmax_dim`
`\l__draw_ymax_dim` 1380 `\dim_new:N \l__draw_xmin_dim`
`\l__draw_ymin_dim` 1381 `\dim_new:N \l__draw_ymax_dim`
1382 `\dim_new:N \l__draw_ymin_dim`

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

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

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

(End definition for `_draw_scope_bb_begin:` and `_draw_scope_bb_end:`.)

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

```
\draw_suspend_end: 1400 \cs_new_protected:Npn \draw_suspend_begin:
1401 {
1402   \_draw_scope_bb_begin:
1403   \draw_path_scope_begin:
1404   \draw_transform_matrix_reset:
1405   \draw_transform_shift_reset:
1406   \_draw_layers_save:
1407 }
1408 \cs_new_protected:Npn \draw_suspend_end:
1409 {
1410   \_draw_layers_restore:
1411   \draw_path_scope_end:
1412   \_draw_scope_bb_end:
1413 }
```

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

```
1414 \</package>
```

7 l3draw-softpath implementation

```
1415 \*package>
```

```
1416 \@@=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.

```
1417 \tl_new:N \g__draw_softpath_main_tl
```

(End definition for `\g__draw_softpath_main_tl`.)

`\l__draw_softpath_internal_tl` The soft path itself.

```
1418 \tl_new:N \l__draw_softpath_internal_tl
```

(End definition for `\l__draw_softpath_internal_tl`.)

`\g__draw_softpath_corners_bool` Allow for optimised path use.

```
1419 \bool_new:N \g__draw_softpath_corners_bool
```


(End definition for \g__draw_softpath_corners_bool.)

```

\__draw_softpath_add:n
\__draw_softpath_add:o 1420 \cs_new_protected:Npn \__draw_softpath_add:n
\__draw_softpath_add:x 1421 { \tl_build_gput_right:Nn \g__draw_softpath_main_tl }
1422 \cs_generate_variant:Nn \__draw_softpath_add:n { o, x }

```

(End definition for __draw_softpath_add:n.)

```

\__draw_softpath_use: Using and clearing is trivial.
\__draw_softpath_clear: 1423 \cs_new_protected:Npn \__draw_softpath_use:
1424 {
1425   \tl_build_get:NN \g__draw_softpath_main_tl \l__draw_softpath_internal_tl
1426   \l__draw_softpath_internal_tl
1427 }
1428 \cs_new_protected:Npn \__draw_softpath_clear:
1429 {
1430   \tl_build_gclear:N \g__draw_softpath_main_tl
1431   \bool_gset_false:N \g__draw_softpath_corners_bool
1432 }

```

(End 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 1433 \dim_new:N \g__draw_softpath_lastx_dim
1434 \dim_new:N \g__draw_softpath_lasty_dim

```

(End 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.
1435 \bool_new:N \g__draw_softpath_move_bool
1436 \bool_gset_true:N \g__draw_softpath_move_bool

```

(End 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 1437 \cs_new_protected:Npn \__draw_softpath_closepath:
\__draw_softpath_moveto:nn 1438 {
1439   \__draw_softpath_add:x
1440   {
1441     \__draw_softpath_close_op:nn
1442     { \dim_use:N \g__draw_softpath_lastx_dim }
1443     { \dim_use:N \g__draw_softpath_lasty_dim }
1444   }
1445 }
1446 \cs_new_protected:Npn \__draw_softpath_curveto:nnnnnn #1#2#3#4#5#6
1447 {
1448   \__draw_softpath_add:n
1449   {
1450     \__draw_softpath_curveto_opi:nn {#1} {#2}
1451     \__draw_softpath_curveto_opii:nn {#3} {#4}
1452     \__draw_softpath_curveto_opiii:nn {#5} {#6}
1453   }
1454 }
1455 \cs_new_protected:Npn \__draw_softpath_lineto:nn #1#2

```

```

1456 {
1457   \__draw_softpath_add:n
1458   { \__draw_softpath_lineto_op:nn {#1} {#2} }
1459 }
1460 \cs_new_protected:Npn \__draw_softpath_moveto:nn #1#2
1461 {
1462   \__draw_softpath_add:n
1463   { \__draw_softpath_moveto_op:nn {#1} {#2} }
1464   \bool_if:NT \g__draw_softpath_move_bool
1465   {
1466     \dim_gset:Nn \g__draw_softpath_lastx_dim {#1}
1467     \dim_gset:Nn \g__draw_softpath_lasty_dim {#2}
1468   }
1469 }
1470 \cs_new_protected:Npn \__draw_softpath_rectangle:nnnn #1#2#3#4
1471 {
1472   \__draw_softpath_add:n
1473   {
1474     \__draw_softpath_rectangle_opi:nn {#1} {#2}
1475     \__draw_softpath_rectangle_opii:nn {#3} {#4}
1476   }
1477 }
1478 \cs_new_protected:Npn \__draw_softpath_roundpoint:nn #1#2
1479 {
1480   \__draw_softpath_add:n
1481   { \__draw_softpath_roundpoint_op:nn {#1} {#2} }
1482   \bool_gset_true:N \g__draw_softpath_corners_bool
1483 }
1484 \cs_generate_variant:Nn \__draw_softpath_roundpoint:nn { VV }

```

(End definition for __draw_softpath_curveto:nnnnnn and others.)

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

```

1485 \cs_new_protected:Npn \__draw_softpath_close_op:nn #1#2
1486 { \__draw_backend_closepath: }
1487 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nn #1#2
1488 { \__draw_softpath_curveto_opi:nnNnnNnn {#1} {#2} }
1489 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nnNnnNnn #1#2#3#4#5#6#7#8
1490 { \__draw_backend_curveto:nnnnnn {#1} {#2} {#4} {#5} {#7} {#8} }
1491 \cs_new_protected:Npn \__draw_softpath_curveto_opii:nn #1#2
1492 { \__draw_softpath_curveto_opii:nn }
1493 \cs_new_protected:Npn \__draw_softpath_curveto_opiii:nn #1#2
1494 { \__draw_softpath_curveto_opiii:nn }
1495 \cs_new_protected:Npn \__draw_softpath_lineto_op:nn #1#2
1496 { \__draw_backend_lineto:nn {#1} {#2} }
1497 \cs_new_protected:Npn \__draw_softpath_moveto_op:nn #1#2
1498 { \__draw_backend_moveto:nn {#1} {#2} }
1499 \cs_new_protected:Npn \__draw_softpath_roundpoint_op:nn #1#2 { }
1500 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nn #1#2
1501 { \__draw_softpath_rectangle_opi:nnNnn {#1} {#2} }
1502 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nnNnn #1#2#3#4#5
1503 { \__draw_backend_rectangle:nnnn {#1} {#2} {#4} {#5} }

```

```
1504 \cs_new_protected:Npn \__draw_softpath_rectangle_opii:nn #1#2 { }
```

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

7.2 Rounding soft path corners

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

`\l__draw_softpath_main_tl` For constructing the updated path.

```
1505 \tl_new:N \l__draw_softpath_main_tl
```

(End definition for `\l__draw_softpath_main_tl`.)

`\l__draw_softpath_part_tl` Data structures.

```
1506 \tl_new:N \l__draw_softpath_part_tl
```

```
1507 \tl_new:N \l__draw_softpath_curve_end_tl
```

(End definition for `\l__draw_softpath_part_tl`.)

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

`\l__draw_softpath_lasty_fp`

```
1508 \fp_new:N \l__draw_softpath_lastx_fp
```

```
1509 \fp_new:N \l__draw_softpath_lasty_fp
```

`\l__draw_softpath_corneri_dim`

```
1510 \dim_new:N \l__draw_softpath_corneri_dim
```

`\l__draw_softpath_cornerii_dim`

```
1511 \dim_new:N \l__draw_softpath_cornerii_dim
```

`\l__draw_softpath_first_tl`

```
1512 \tl_new:N \l__draw_softpath_first_tl
```

`\l__draw_softpath_move_tl`

```
1513 \tl_new:N \l__draw_softpath_move_tl
```

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

`\c__draw_softpath_arc_fp` The magic constant.

```
1514 \fp_const:Nn \c__draw_softpath_arc_fp { 4/3 * (sqrt(2) - 1) }
```

(End definition for `\c__draw_softpath_arc_fp`.)

`__draw_softpath_round_corners:`

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

`__draw_softpath_round_loop:Nnn`

`__draw_softpath_round_action:nn`

`__draw_softpath_round_action:Nnn`

`__draw_softpath_round_action_curveto:NnnNnn`

`__draw_softpath_round_action_close:`

`__draw_softpath_round_lookahead:NnnNnn`

`__draw_softpath_round_roundpoint:NnnNnnNnn`

`__draw_softpath_round_calc:NnnNnn`

`__draw_softpath_round_calc:nnnnnn`

`__draw_softpath_round_calc:fVnnnn`

`__draw_softpath_round_calc:nnnnw`

`__draw_softpath_round_close:nn`

`__draw_softpath_round_close:w`

`__draw_softpath_round_end:`

```
1515 \cs_new_protected:Npn \__draw_softpath_round_corners:
```

```
1516 {
```

```
1517   \bool_if:NT \g__draw_softpath_corners_bool
```

```
1518   {
```

```
1519     \group_begin:
```

```
1520       \tl_clear:N \l__draw_softpath_main_tl
```

```
1521       \tl_clear:N \l__draw_softpath_part_tl
```

```
1522       \fp_zero:N \l__draw_softpath_lastx_fp
```

```
1523       \fp_zero:N \l__draw_softpath_lasty_fp
```

```
1524       \tl_clear:N \l__draw_softpath_first_tl
```

```
1525       \tl_clear:N \l__draw_softpath_move_tl
```

```
1526       \tl_build_get:NN \g__draw_softpath_main_tl \l__draw_softpath_internal_tl
```

```
1527       \exp_after:wN \__draw_softpath_round_loop:Nnn
```

```
1528       \l__draw_softpath_internal_tl
```

```
1529       \q__draw_recursion_tail ? ?
```

```
1530       \q__draw_recursion_stop
```

```

1531     \group_end:
1532   }
1533   \bool_gset_false:N \g__draw_softpath_corners_bool
1534 }

```

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.

```

1535 \cs_new_protected:Npn \__draw_softpath_round_loop:Nnn #1#2#3
1536 {
1537   \__draw_if_recursion_tail_stop_do:Nn #1 { \__draw_softpath_round_end: }
1538   \token_if_eq_meaning:NNTF #1 \__draw_softpath_roundpoint_op:nn
1539   { \__draw_softpath_round_action:nn {#2} {#3} }
1540   {
1541     \tl_if_empty:NT \l__draw_softpath_first_tl
1542     { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1543     \fp_set:Nn \l__draw_softpath_lastx_fp {#2}
1544     \fp_set:Nn \l__draw_softpath_lasty_fp {#3}
1545     \token_if_eq_meaning:NNTF #1 \__draw_softpath_moveto_op:nn
1546     {
1547       \tl_put_right:No \l__draw_softpath_main_tl
1548       \l__draw_softpath_move_tl
1549       \tl_put_right:No \l__draw_softpath_main_tl
1550       \l__draw_softpath_part_tl
1551       \tl_set:Nn \l__draw_softpath_move_tl { #1 {#2} {#3} }
1552       \tl_clear:N \l__draw_softpath_first_tl
1553       \tl_clear:N \l__draw_softpath_part_tl
1554     }
1555     { \tl_put_right:Nn \l__draw_softpath_part_tl { #1 {#2} {#3} } }
1556     \__draw_softpath_round_loop:Nnn
1557   }
1558 }
1559 \cs_new_protected:Npn \__draw_softpath_round_action:nn #1#2
1560 {
1561   \dim_set:Nn \l__draw_softpath_corneri_dim {#1}
1562   \dim_set:Nn \l__draw_softpath_cornerii_dim {#2}
1563   \bool_lazy_and:nnTF
1564   { \dim_compare_p:nNn \l__draw_softpath_corneri_dim = { Opt } }
1565   { \dim_compare_p:nNn \l__draw_softpath_cornerii_dim = { Opt } }
1566   { \__draw_softpath_round_loop:Nnn }
1567   { \__draw_softpath_round_action:Nnn }
1568 }

```

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.

```

1569 \cs_new_protected:Npn \__draw_softpath_round_action:Nnn #1#2#3
1570 {
1571   \tl_if_empty:NT \l__draw_softpath_first_tl
1572   { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1573   \token_if_eq_meaning:NNTF #1 \__draw_softpath_curveto_opi:nn
1574   { \__draw_softpath_round_action_curveto:NnnNnn }

```

```

1575     {
1576         \token_if_eq_meaning:NNTF #1 \__draw_softpath_close_op:nn
1577         { \__draw_softpath_round_action_close: }
1578         {
1579             \token_if_eq_meaning:NNTF #1 \__draw_softpath_lineto_op:nn
1580             { \__draw_softpath_round_lookahead:NnnNnn }
1581             { \__draw_softpath_round_loop:Nnn }
1582         }
1583     }
1584     #1 {#2} {#3}
1585 }

```

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.

```

1586 \cs_new_protected:Npn \__draw_softpath_round_action_curveto:NnnNnn
1587   #1#2#3#4#5#6
1588   {
1589     \tl_put_right:Nn \l__draw_softpath_part_tl
1590     { #1 {#2} {#3} #4 {#5} {#6} }
1591     \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1592     \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1593     \__draw_softpath_round_lookahead:NnnNnn
1594   }
1595 \cs_new_protected:Npn \__draw_softpath_round_action_close:
1596   {
1597     \bool_lazy_and:nnTF
1598     { ! \tl_if_empty_p:N \l__draw_softpath_first_tl }
1599     { ! \tl_if_empty_p:N \l__draw_softpath_move_tl }
1600     {
1601       \exp_after:wN \__draw_softpath_round_close:nn
1602       \l__draw_softpath_first_tl
1603     }
1604     { \__draw_softpath_round_loop:Nnn }
1605   }

```

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.

```

1606 \cs_new_protected:Npn \__draw_softpath_round_lookahead:NnnNnn #1#2#3#4#5#6
1607   {
1608     \bool_lazy_any:nTF
1609     {
1610       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_lineto_op:nn }
1611       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_curveto_opi:nn }
1612       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_close_op:nn }
1613     }
1614     {
1615       \__draw_softpath_round_calc:NnnNnn
1616       \__draw_softpath_round_loop:Nnn
1617       {#5} {#6}
1618     }
1619     {
1620       \token_if_eq_meaning:NNTF #4 \__draw_softpath_roundpoint_op:nn
1621       { \__draw_softpath_round_roundpoint:NnnNnnNnn }
1622       { \__draw_softpath_round_loop:Nnn }

```

```

1623     }
1624     #1 {#2} {#3}
1625     #4 {#5} {#6}
1626 }
1627 \cs_new_protected:Npn \__draw_softpath_round_roundpoint:NnnNnnNnn
1628 #1#2#3#4#5#6#7#8#9
1629 {
1630     \__draw_softpath_round_calc:NnnNnn
1631     \__draw_softpath_round_loop:Nnn
1632     {#8} {#9}
1633     #1 {#2} {#3}
1634     #4 {#5} {#6} #7 {#8} {#9}
1635 }

```

We now have all of the data needed to construct a rounded corner: all that is left to do is to work out the detail! At this stage, we have details of where the corner itself is (#5, #6), and where the next point is (#2, #3). There are two types of calculations to do. First, we need to interpolate from those two points in the direction of the corner, in order to work out where the curve we are adding will start and end. From those, plus the points we already have, we work out where the control points will lie. All of this is done in an expansion to avoid multiple calls to `\tl_put_right:Nx`. The end point of the line is worked out up-front and saved: we need that if dealing with a close-path operation.

```

1636 \cs_new_protected:Npn \__draw_softpath_round_calc:NnnNnn #1#2#3#4#5#6
1637 {
1638     \tl_set:Nx \l__draw_softpath_curve_end_tl
1639     {
1640         \draw_point_interpolate_distance:nnn
1641         \l__draw_softpath_cornerii_dim
1642         { #5 , #6 } { #2 , #3 }
1643     }
1644     \tl_put_right:Nx \l__draw_softpath_part_tl
1645     {
1646         \exp_not:N #4
1647         \__draw_softpath_round_calc:fVnnnn
1648         {
1649             \draw_point_interpolate_distance:nnn
1650             \l__draw_softpath_corneri_dim
1651             { #5 , #6 }
1652             {
1653                 \l__draw_softpath_lastx_fp ,
1654                 \l__draw_softpath_lasty_fp
1655             }
1656         }
1657         \l__draw_softpath_curve_end_tl
1658         {#5} {#6} {#2} {#3}
1659     }
1660     \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1661     \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1662     #1
1663 }

```

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).

```

1664 \cs_new:Npn \__draw_softpath_round_calc:nnnnnn #1#2#3#4#5#6

```

```

1665 {
1666   \__draw_softpath_round_calc:nnnw {#3} {#4} {#5} {#6}
1667   #1 \s__draw_mark #2 \s__draw_stop
1668 }
1669 \cs_generate_variant:Nn \__draw_softpath_round_calc:nnnnnn { fV }

```

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

```

1670 \cs_new:Npn \__draw_softpath_round_calc:nnnw
1671 #1#2#3#4 #5 , #6 \s__draw_mark #7 , #8 \s__draw_stop
1672 {
1673   {#5} {#6}
1674   \exp_not:N \__draw_softpath_curveto_opi:nn
1675   {
1676     \fp_to_dim:n
1677     { #5 + \c__draw_softpath_arc_fp * ( #1 - #5 ) }
1678   }
1679   {
1680     \fp_to_dim:n
1681     { #6 + \c__draw_softpath_arc_fp * ( #2 - #6 ) }
1682   }
1683   \exp_not:N \__draw_softpath_curveto_opii:nn
1684   {
1685     \fp_to_dim:n
1686     { #7 + \c__draw_softpath_arc_fp * ( #1 - #7 ) }
1687   }
1688   {
1689     \fp_to_dim:n
1690     { #8 + \c__draw_softpath_arc_fp * ( #2 - #8 ) }
1691   }
1692   \exp_not:N \__draw_softpath_curveto_opiii:nn
1693   {#7} {#8}
1694 }

```

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.

```

1695 \cs_new_protected:Npn \__draw_softpath_round_close:nn #1#2
1696 {
1697   \use:x
1698   {
1699     \__draw_softpath_round_calc:NnnNnn
1700     {
1701       \tl_set:Nx \exp_not:N \l__draw_softpath_move_tl
1702       {
1703         \__draw_softpath_moveto_op:nn
1704         \exp_not:N \exp_after:wN
1705         \exp_not:N \__draw_softpath_round_close:w
1706         \exp_not:N \l__draw_softpath_curve_end_tl
1707         \s__draw_stop
1708       }
1709       \use:x
1710       {
1711         \exp_not:N \exp_not:N \exp_not:N \use_i:nnnn

```

```

1712         {
1713             \__draw_softpath_round_loop:Nnn
1714             \__draw_softpath_close_op:nn
1715             \exp_not:N \exp_after:wN
1716             \exp_not:N \__draw_softpath_round_close:w
1717             \exp_not:N \l__draw_softpath_curve_end_tl
1718             \s__draw_stop
1719         }
1720     }
1721 }
1722 {#1} {#2}
1723 \__draw_softpath_lineto_op:nn
1724 \exp_after:wN \use_none:n \l__draw_softpath_move_tl
1725 }
1726 }
1727 \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.

```

1728 \cs_new_protected:Npn \__draw_softpath_round_end:
1729 {
1730     \tl_put_right:No \l__draw_softpath_main_tl
1731     \l__draw_softpath_move_tl
1732     \tl_put_right:No \l__draw_softpath_main_tl
1733     \l__draw_softpath_part_tl
1734     \tl_build_gclear:N \g__draw_softpath_main_tl
1735     \__draw_softpath_add:o \l__draw_softpath_main_tl
1736 }

```

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

```

1737 </package>

```

8 l3draw-state implementation

```

1738 <*package>

```

```

1739 <@@=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.

```

1740 \dim_new:N \g__draw_linewidth_dim

```

(End definition for `\g__draw_linewidth_dim`.)

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

```

1741 \dim_new:N \l_draw_default_linewidth_dim
1742 \dim_set:Nn \l_draw_default_linewidth_dim { 0.4pt }

```

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


```

\draw_linewidth:n Set the linewidth: we need a wrapper as this has to pass to the driver layer.
1743 \cs_new_protected:Npn \draw_linewidth:n #1
1744 {
1745     \dim_gset:Nn \g__draw_linewidth_dim { \fp_to_dim:n {#1} }
1746     \__draw_backend_linewidth:n \g__draw_linewidth_dim
1747 }

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

\draw_dash_pattern:nn Evaluated all of the list and pass it to the driver layer.
\l__draw_tmp_seq
1748 \cs_new_protected:Npn \draw_dash_pattern:nn #1#2
1749 {
1750     \group_begin:
1751     \seq_set_from_clist:Nn \l__draw_tmp_seq {#1}
1752     \seq_set_map:Nn \l__draw_tmp_seq \l__draw_tmp_seq
1753     { \fp_to_dim:n {##1} }
1754     \use:x
1755     {
1756         \__draw_backend_dash_pattern:nn
1757         { \seq_use:Nn \l__draw_tmp_seq { , } }
1758         { \fp_to_dim:n {#2} }
1759     }
1760     \group_end:
1761 }
1762 \seq_new:N \l__draw_tmp_seq

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

\draw_miterlimit:n Pass through to the driver layer.
1763 \cs_new_protected:Npn \draw_miterlimit:n #1
1764 { \exp_args:Nx \__draw_backend_miterlimit:n { \fp_eval:n {#1} } }

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

\draw_cap_but: All straight wrappers.
\draw_cap_rectangle:
1765 \cs_new_protected:Npn \draw_cap_but: { \__draw_backend_cap_but: }
\draw_cap_round:
1766 \cs_new_protected:Npn \draw_cap_rectangle: { \__draw_backend_cap_rectangle: }
\draw_evenodd_rule:
1767 \cs_new_protected:Npn \draw_cap_round: { \__draw_backend_cap_round: }
\draw_nonzero_rule:
1768 \cs_new_protected:Npn \draw_evenodd_rule: { \__draw_backend_evenodd_rule: }
\draw_join_bevel:
1769 \cs_new_protected:Npn \draw_nonzero_rule: { \__draw_backend_nonzero_rule: }
\draw_join_miter:
1770 \cs_new_protected:Npn \draw_join_bevel: { \__draw_backend_join_bevel: }
\draw_join_round:
1771 \cs_new_protected:Npn \draw_join_miter: { \__draw_backend_join_miter: }
1772 \cs_new_protected:Npn \draw_join_round: { \__draw_backend_join_round: }

(End definition for \draw_cap_but: and others. These functions are documented on page ??.)
1773 </package>

```

9 l3draw-transforms implementation

1774 `\package`

1775 `\@@=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.

1776 `\bool_new:N \l__draw_matrix_active_bool`

(End definition for `\l__draw_matrix_active_bool`.)

`\l__draw_matrix_a_fp` The active matrix and shifts.

`\l__draw_matrix_b_fp` 1777 `\fp_new:N \l__draw_matrix_a_fp`

`\l__draw_matrix_c_fp` 1778 `\fp_new:N \l__draw_matrix_b_fp`

`\l__draw_xshift_dim` 1779 `\fp_new:N \l__draw_matrix_c_fp`

`\l__draw_yshift_dim` 1780 `\fp_new:N \l__draw_matrix_d_fp`

1781 `\dim_new:N \l__draw_xshift_dim`

1782 `\dim_new:N \l__draw_yshift_dim`

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

`\draw_transform_matrix_reset:` Fast resetting.

`\draw_transform_shift_reset:` 1783 `\cs_new_protected:Npn \draw_transform_matrix_reset:`

1784 `{`

1785 `\fp_set:Nn \l__draw_matrix_a_fp { 1 }`

1786 `\fp_zero:N \l__draw_matrix_b_fp`

1787 `\fp_zero:N \l__draw_matrix_c_fp`

1788 `\fp_set:Nn \l__draw_matrix_d_fp { 1 }`

1789 `}`

1790 `\cs_new_protected:Npn \draw_transform_shift_reset:`

1791 `{`

1792 `\dim_zero:N \l__draw_xshift_dim`

1793 `\dim_zero:N \l__draw_yshift_dim`

1794 `}`

1795 `\draw_transform_matrix_reset:`

1796 `\draw_transform_shift_reset:`

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

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

```

1797 \cs_new_protected:Npn \draw_transform_matrix_absolute:nnnn #1#2#3#4
1798 {
1799   \fp_set:Nn \l__draw_matrix_a_fp {#1}
1800   \fp_set:Nn \l__draw_matrix_b_fp {#2}
1801   \fp_set:Nn \l__draw_matrix_c_fp {#3}
1802   \fp_set:Nn \l__draw_matrix_d_fp {#4}
1803   \bool_lazy_all:nTF
1804     {
1805       { \fp_compare_p:nNn \l__draw_matrix_a_fp = \c_one_fp }
1806       { \fp_compare_p:nNn \l__draw_matrix_b_fp = \c_zero_fp }
1807       { \fp_compare_p:nNn \l__draw_matrix_c_fp = \c_zero_fp }
1808       { \fp_compare_p:nNn \l__draw_matrix_d_fp = \c_one_fp }
1809     }
1810     { \bool_set_false:N \l__draw_matrix_active_bool }
1811     { \bool_set_true:N \l__draw_matrix_active_bool }
1812   }
1813 \cs_new_protected:Npn \draw_transform_shift_absolute:n #1
1814 {
1815   \__draw_point_process:nn
1816   { \__draw_transform_shift_absolute:nn } {#1}
1817 }
1818 \cs_new_protected:Npn \__draw_transform_shift_absolute:nn #1#2
1819 {
1820   \dim_set:Nn \l__draw_xshift_dim {#1}
1821   \dim_set:Nn \l__draw_yshift_dim {#2}
1822 }

```

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

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

```

1823 \cs_new_protected:Npn \draw_transform_matrix:nnnn #1#2#3#4
1824 {
1825   \use:x
1826   {
1827     \__draw_transform:nnnn
1828     { \fp_eval:n {#1} }
1829     { \fp_eval:n {#2} }
1830     { \fp_eval:n {#3} }
1831     { \fp_eval:n {#4} }
1832   }
1833 }
1834 \cs_new_protected:Npn \__draw_transform:nnnn #1#2#3#4
1835 {
1836   \use:x
1837   {
1838     \draw_transform_matrix_absolute:nnnn
1839     { #1 * \l__draw_matrix_a_fp + #2 * \l__draw_matrix_c_fp }
1840     { #1 * \l__draw_matrix_b_fp + #2 * \l__draw_matrix_d_fp }
1841     { #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_c_fp }
1842     { #3 * \l__draw_matrix_b_fp + #4 * \l__draw_matrix_d_fp }

```

```

1843     }
1844 }
1845 \cs_new_protected:Npn \draw_transform_shift:n #1
1846 {
1847   \__draw_point_process:nn
1848   { \__draw_transform_shift:nn } {#1}
1849 }
1850 \cs_new_protected:Npn \__draw_transform_shift:nn #1#2
1851 {
1852   \dim_set:Nn \l__draw_xshift_dim { \l__draw_xshift_dim + #1 }
1853   \dim_set:Nn \l__draw_yshift_dim { \l__draw_yshift_dim + #2 }
1854 }

```

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

<pre> \draw_transform_matrix_invert: __draw_transform_invert:n __draw_transform_invert:f \draw_transform_shift_invert: </pre>	<p>Standard mathematics: calculate the inverse matrix and use that, then undo the shifts.</p> <pre> 1855 \cs_new_protected:Npn \draw_transform_matrix_invert: 1856 { 1857 \bool_if:NT \l__draw_matrix_active_bool 1858 { 1859 __draw_transform_invert:f 1860 { 1861 \fp_eval:n 1862 { 1863 1 / 1864 (1865 \l__draw_matrix_a_fp * \l__draw_matrix_d_fp 1866 - \l__draw_matrix_b_fp * \l__draw_matrix_c_fp 1867) 1868 } 1869 } 1870 } 1871 } 1872 \cs_new_protected:Npn __draw_transform_invert:n #1 1873 { 1874 \fp_set:Nn \l__draw_matrix_a_fp 1875 { \l__draw_matrix_d_fp * #1 } 1876 \fp_set:Nn \l__draw_matrix_b_fp 1877 { -\l__draw_matrix_b_fp * #1 } 1878 \fp_set:Nn \l__draw_matrix_c_fp 1879 { -\l__draw_matrix_c_fp * #1 } 1880 \fp_set:Nn \l__draw_matrix_d_fp 1881 { \l__draw_matrix_a_fp * #1 } 1882 } 1883 \cs_generate_variant:Nn __draw_transform_invert:n { f } 1884 \cs_new_protected:Npn \draw_transform_shift_invert: 1885 { 1886 \dim_set:Nn \l__draw_xshift_dim { -\l__draw_xshift_dim } 1887 \dim_set:Nn \l__draw_yshift_dim { -\l__draw_yshift_dim } 1888 } </pre>
---	--

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

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

```

1889 \cs_new_protected:Npn \draw_transform_triangle:nnn #1#2#3
1890 {
1891   \__draw_point_process:nnn
1892   {
1893     \__draw_point_process:nn
1894     { \__draw_tranform_triangle:nnnnnn }
1895     {#1}
1896   }
1897   {#2} {#3}
1898 }
1899 \cs_new_protected:Npn \__draw_tranform_triangle:nnnnnn #1#2#3#4#5#6
1900 {
1901   \use:x
1902   {
1903     \draw_transform_matrix_absolute:nnnn
1904     { #3 - #1 }
1905     { #4 - #2 }
1906     { #5 - #1 }
1907     { #6 - #2 }
1908     \draw_transform_shift_absolute:n { #1 , #2 }
1909   }
1910 }

```

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

`\draw_transform_scale:n` Lots of shortcuts.

```

\draw_transform_xscale:n 1911 \cs_new_protected:Npn \draw_transform_scale:n #1
\draw_transform_yscale:n 1912 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { #1 } }
\draw_transform_xshift:n 1913 \cs_new_protected:Npn \draw_transform_xscale:n #1
\draw_transform_yshift:n 1914 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { 1 } }
\draw_transform_xslant:n 1915 \cs_new_protected:Npn \draw_transform_yscale:n #1
\draw_transform_yslant:n 1916 { \draw_transform_matrix:nnnn { 1 } { 0 } { 0 } { #1 } }
1917 \cs_new_protected:Npn \draw_transform_xshift:n #1
1918 { \draw_transform_shift:n { #1 , Opt } }
1919 \cs_new_protected:Npn \draw_transform_yshift:n #1
1920 { \draw_transform_shift:n { Opt , #1 } }
1921 \cs_new_protected:Npn \draw_transform_xslant:n #1
1922 { \draw_transform_matrix:nnnn { 1 } { 0 } { #1 } { 1 } }
1923 \cs_new_protected:Npn \draw_transform_yslant:n #1
1924 { \draw_transform_matrix:nnnn { 1 } { #1 } { 0 } { 1 } }

```

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

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

```

\__draw_transform_rotate:n 1925 \cs_new_protected:Npn \draw_transform_rotate:n #1
\__draw_transform_rotate:f 1926 { \__draw_transform_rotate:f { \fp_eval:n {#1} } }
\__draw_transform_rotate:nn 1927 \cs_new_protected:Npn \__draw_transform_rotate:n #1
\__draw_transform_rotate:ff 1928 {
1929   \__draw_transform_rotate:ff
1930   { \fp_eval:n { cosd(#1) } }
1931   { \fp_eval:n { sind(#1) } }
1932 }
1933 \cs_generate_variant:Nn \__draw_transform_rotate:n { f }

```

```

1934 \cs_new_protected:Npn \__draw_transform_rotate:nn #1#2
1935   { \draw_transform_matrix:nnnn {#1} {#2} { -#2 } { #1 } }
1936 \cs_generate_variant:Nn \__draw_transform_rotate:nn { ff }

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

1937 \</package>

```

Index

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

B

`\begin` . . . 171, 786, 1079, 1083, 1107, 1110

bool commands:

`\bool_gset_eq:NN` 767

`\bool_gset_false:N` 1431, 1533

`\bool_gset_true:N` 1436, 1482

`\bool_if:NTF`

. 21, 115, 194, 233, 682, 698, 699,

703, 1202, 1234, 1352, 1464, 1517, 1857

`\bool_lazy_all:nTF` 1803

`\bool_lazy_and:nnTF`

. 225, 677, 1563, 1597

`\bool_lazy_any:nTF` 1608

`\bool_lazy_or:nnTF` . 558, 653, 686, 691

`\bool_new:N`

. 86, 220, 641, 642, 643, 644,

645, 745, 1258, 1343, 1419, 1435, 1776

`\bool_set_eq:NN` 759

`\bool_set_false:N`

. 96, 228, 664, 665, 666, 685, 1810

`\bool_set_true:N` 98, 229,

670, 708, 712, 713, 1277, 1347, 1811

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` 1317, 1330

`\box_move_up:nn` 51

`\box_new:N` 13, 80, 81, 1259, 1260

`\box_set_dp:Nn` 55, 1321, 1334

`\box_set_eq:NN` 145

`\box_set_ht:Nn` 54, 1322, 1340

`\box_set_wd:Nn` 56, 128, 1306

`\box_use_drop:N` 52,

57, 102, 129, 134, 1309, 1319, 1332

`\box_wd:N` 17, 66, 68

C

clist commands:

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

`\clist_map_inline:nn` 667

`\clist_new:N` 87, 89

`\clist_set:Nn` 88, 1294

coffin commands:

`\coffin_typeset:Nnnnn` 64

`\coffin_wd:N` 66

color commands:

`\color_select:n` 1282

cs commands:

`\cs_generate_variant:Nn`

. 420, 607, 640,

845, 853, 895, 914, 921, 929, 936,

945, 951, 971, 979, 986, 992, 1004,

1007, 1025, 1043, 1051, 1057, 1078,

1092, 1106, 1127, 1162, 1181, 1194,

1422, 1484, 1669, 1883, 1933, 1936

`\cs_if_exist:NTF` 669

`\cs_if_exist_use:NTF` . . 403, 412, 672

`\cs_new:Npn` 511, 521, 531, 541,

790, 796, 798, 800, 807, 809, 811,

819, 821, 824, 833, 838, 841, 843,

846, 847, 849, 851, 854, 856, 862,

871, 877, 887, 896, 902, 907, 915,

922, 930, 937, 946, 952, 958, 963,

972, 980, 987, 993, 999, 1005, 1008,

1014, 1023, 1026, 1032, 1037, 1044,

1052, 1058, 1064, 1070, 1084, 1093,

1099, 1111, 1113, 1122, 1152, 1154,

1163, 1168, 1182, 1184, 1186, 1195,

1200, 1227, 1232, 1664, 1670, 1727

`\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, 489,

497, 552, 554, 567, 574, 583, 589,

591, 601, 608, 614, 621, 646, 651,

662, 706, 710, 715, 722, 746, 764,

1134, 1136, 1138, 1140, 1144, 1262,

1269, 1290, 1312, 1325, 1345, 1350,

1365, 1372, 1383, 1392, 1400, 1408,

1420, 1423, 1428, 1437, 1446, 1455,

1460, 1470, 1478, 1485, 1487, 1489,

1491, 1493, 1495, 1497, 1499, 1500,

1502, 1504, 1515, 1535, 1559, 1569,

1586, 1595, 1606, 1627, 1636, 1695,

1728, 1743, 1748, 1763, 1765, 1766,

1767, 1768, 1769, 1770, 1771, 1772,

1783, 1790, 1797, 1813, 1818, 1823,

1834, 1845, 1850, 1855, 1872, 1884,

1889, 1899, 1911, 1913, 1915, 1917,

1919, 1921, 1923, 1925, 1927, 1934

D

dim commands:

`\dim_abs:n` 596, 597
`\dim_compare:nNnTF` 603, 610, 724, 1298
`\dim_compare_p:nNn` 226, 227, 1564, 1565
`\dim_eval:n` 596, 597
`\dim_gset:Nn` 186, 188,
190, 192, 196, 198, 200, 202, 208,
209, 210, 211, 215, 216, 726, 1264,
1265, 1266, 1267, 1466, 1467, 1745
`\dim_gset_eq:NN`
.... 771, 772, 773, 774, 775, 776,
777, 778, 1375, 1394, 1395, 1396, 1397
`\dim_gzero:N` .. 1300, 1301, 1302, 1303
`\dim_max:nn` .. 187, 191, 197, 201, 1336
`\dim_min:nn` 189, 193, 199, 203
`\dim_new:N` 178, 179,
180, 181, 182, 183, 218, 219, 737,
738, 739, 740, 741, 742, 743, 744,
1128, 1129, 1130, 1131, 1132, 1133,
1254, 1255, 1256, 1257, 1344, 1362,
1379, 1380, 1381, 1382, 1433, 1434,
1510, 1511, 1740, 1741, 1781, 1782
`\dim_set:Nn` 223, 224,
1146, 1147, 1348, 1561, 1562, 1742,
1820, 1821, 1852, 1853, 1886, 1887
`\dim_set_eq:NN`
.... 749, 750, 751, 752, 753, 754,
755, 756, 1369, 1386, 1387, 1388, 1389
`\dim_step_inline:nnnn` 623, 631
`\dim_use:N`
.... 724, 729, 731, 1357, 1442, 1443
`\dim_zero:N` 1792, 1793
`\c_max_dim` 208, 209, 210,
211, 724, 1264, 1265, 1266, 1267, 1298

draw commands:

`\draw_baseline:n` 1345, 1345
`\l_draw_bb_update_bool`
.... 21, 194, 678, 685, 1258, 1277
`\draw_begin:` 1269, 1269
`\draw_box_use:N` 14, 14
`\draw_cap_but:` 1284, 1765, 1765
`\draw_cap_rectangle:` 1765, 1766
`\draw_cap_round:` 1765, 1767
`\draw_coffin_use:Nnn` 36, 60, 60
`\draw_dash_pattern:nn` 1287, 1748, 1748
`\l_draw_default_linewidth_dim` ...
.... 105, 1281, 1741
`\draw_end:` 1269, 1290
`\draw_evenodd_rule:` 1765, 1768
`\draw_join_bevel:` 1765, 1770
`\draw_join_miter:` ... 1285, 1765, 1771
`\draw_join_round:` 1765, 1772
`\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, 1294
`\draw_linewidth:n` 105, 1281, 1743, 1743
`\draw_miterlimit:n` .. 1286, 1763, 1763
`\draw_nonzero_rule:` . 1283, 1765, 1769
`\draw_path_arc:nnn` 345, 345, 487
`\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` 552, 552
`\draw_path_close:` 284, 284, 580
`\draw_path_corner_arc:nn` ... 221, 221
`\draw_path_curveto:nn` 302, 302
`\draw_path_curveto:nnn` 240, 265
`\draw_path_ellipse:nnn` . 489, 489, 553
`\draw_path_grid:nnnn` 591, 591
`\draw_path_lineto:n`
.... 240, 252, 577, 578, 579, 629, 637
`\draw_path_moveto:n`
.... 240, 240, 576, 581, 628, 636
`\draw_path_rectangle:nn` 554, 554, 590
`\draw_path_rectangle_corners:nn` .
.... 583, 583
`\draw_path_scope_begin:`
.... 746, 746, 1370, 1403
`\draw_path_scope_end:`
.... 746, 764, 1374, 1411
`\draw_path_use:n` 646, 646
`\draw_path_use_clear:n` 646, 651
`\draw_point:n` 793, 803,
804, 814, 815, 816, 827, 828, 829,
830, 841, 841, 852, 867, 889, 948,
989, 1006, 1024, 1054, 1124, 1156,
1170, 1188, 1204, 1220, 1236, 1249
`\draw_point_interpolate_arcaxes:nnnnnn`
.... 1026, 1026
`\draw_point_interpolate_curve:nnnnn`
.... 1058
`\draw_point_interpolate_curve:nnnnnn`
.... 1058
`\draw_point_interpolate_curve_-`
auxi:nnnnnnnn 1058
`\draw_point_interpolate_curve_-`
auxii:nnnnnnnn 1058
`\draw_point_interpolate_curve_-`
auxiii:nnnnnn 1058
`\draw_point_interpolate_curve_-`
auxiv:nnnnnn 1058

<code>\draw_point_interpolate_curve-</code> <code>auxv:nnw</code>	1058	<code>\draw_transform_triangle:nnn</code>	486 , 1889 , 1889
<code>\draw_point_interpolate_curve-</code> <code>auxvi:n</code>	1058	<code>\draw_transform_xscale:n</code>	1911 , 1913
<code>\draw_point_interpolate_curve-</code> <code>auxvii:nnnnnnnn</code>	1058	<code>\draw_transform_xshift:n</code>	1911 , 1917
<code>\draw_point_interpolate_curve-</code> <code>auxviii:nnnnnn</code>	1058	<code>\draw_transform_xslant:n</code>	1911 , 1921
<code>\draw_point_interpolate_distance:nnn</code>	1008 , 1008 , 1640 , 1649	<code>\draw_transform_yscale:n</code>	1911 , 1915
<code>\draw_point_interpolate_line:nnn</code>	993 , 993	<code>\draw_transform_yshift:n</code>	1911 , 1919
<code>\draw_point_intersect_circles:nnnnn</code>	896 , 896	<code>\draw_transform_yslant:n</code>	1911 , 1923
<code>\draw_point_intersect_line-</code> <code>circle:nnnnn</code>	952 , 952	<code>\draw_xvec:n</code>	1134 , 1134 , 1149
<code>\draw_point_intersect_lines:nnnn</code>	871 , 871	<code>\draw_yvec:n</code>	1134 , 1136 , 1150
<code>\draw_point_polar:nn</code>	847 , 847	<code>\draw_zvec:n</code>	1134 , 1138 , 1151
<code>\draw_point_polar:nnn</code>	draw internal commands:	
.	428 , 434 , 438 , 444 , 847 , 848 , 849	<code>__draw_backend_begin:</code>	1274
<code>\draw_point_transform:n</code>	25 , 28 , 31 , 34 , 244 , 256 , 272 , 273 , 274 , 306 , 307 , 433 , 437 , 493 , 564 , 1195 , 1195	<code>__draw_backend_box_use:Nnnnn</code>	41
<code>\draw_point_unit_vector:n</code>	<code>__draw_backend_cap_but:</code>	1765
.	854 , 854 , 1021	<code>__draw_backend_cap_rectangle:</code>	1766
<code>\draw_point_vec:nn</code>	1152 , 1152	<code>__draw_backend_cap_round:</code>	1767
<code>\draw_point_vec:nnn</code>	1152 , 1163	<code>__draw_backend_clip:</code>	684
<code>\draw_point_vec_polar:nn</code>	1182 , 1182	<code>__draw_backend_closepath:</code>	1486
<code>\draw_point_vec_polar:nnn</code>	<code>__draw_backend_curveto:nnnnnn</code>	1490
.	1182 , 1183 , 1184	<code>__draw_backend_dash_pattern:nn</code>	1756
<code>\draw_scope_begin:</code>	1365 , 1365	<code>__draw_backend_discardpath:</code>	689
<code>\draw_scope_end:</code>	1372	<code>__draw_backend_end:</code>	1296
<code>\draw_suspend_begin:</code>	1400 , 1400	<code>__draw_backend_evenodd_rule:</code>	1768
<code>\draw_suspend_end:</code>	1400 , 1408	<code>__draw_backend_join_bevel:</code>	1770
<code>\draw_transform_matrix:nnnn</code>	<code>__draw_backend_join_miter:</code>	1771
.	1823 , 1823 , 1912 , 1914 , 1916 , 1922 , 1924 , 1935	<code>__draw_backend_join_round:</code>	1772
<code>\draw_transform_matrix_absolute:nnnn</code>	1797 , 1797 , 1838 , 1903	<code>__draw_backend_lineto:nn</code>	1496
<code>\draw_transform_matrix_invert:</code>	<code>__draw_backend_linewidth:n</code>	1746
.	1855 , 1855	<code>__draw_backend_miterlimit:n</code>	1764
<code>\draw_transform_matrix_reset:</code>	<code>__draw_backend_moveto:nn</code>	1498
.	1278 , 1404 , 1783 , 1783 , 1795	<code>__draw_backend_nonzero_rule:</code>	1769
<code>\draw_transform_rotate:n</code>	1925 , 1925	<code>__draw_backend_rectangle:nnnn</code>	1503
<code>\draw_transform_scale:n</code>	1911 , 1911	<code>__draw_backend_scope_begin:</code>
<code>\draw_transform_shift:n</code>	132 , 1367
.	1823 , 1845 , 1918 , 1920	<code>__draw_backend_scope_end:</code>	135 , 1377
<code>\draw_transform_shift_absolute:n</code>	1797 , 1813 , 1908	<code>\l_draw_baseline_bool</code>
<code>\draw_transform_shift_invert:</code>	1343 , 1347 , 1352
.	1855 , 1884	<code>\l_draw_baseline_dim</code>	1343 , 1348 , 1357
<code>\draw_transform_shift_reset:</code>	<code>\draw_baseline_finalise:w</code>
.	1279 , 1405 , 1783 , 1790 , 1796	1292 , 1350 , 1350
		<code>__draw_box_use:Nnnnn</code>	14 , 16 , 19 , 65
		<code>\l_draw_corner_arc_bool</code>
		220 , 228 , 229 , 233 , 559
		<code>\l_draw_corner_xarc_dim</code>
		218 , 223 , 226 , 236
		<code>\l_draw_corner_yarc_dim</code>
		218 , 224 , 227 , 237
		<code>__draw_draw_polar:nnn</code>
		847 , 850 , 851 , 853
		<code>__draw_draw_vec_polar:nnn</code>
		1185 , 1186 , 1194
		<code>\l_draw_fill_color_tl</code>	1362

_draw_finalise:	_draw_path_arc_auxvi:nn
..... 1305, 1312, 1312, 1350, 1360 345, 464, 467
_draw_finalise_baseline:n	\l_draw_path_arc_delta_fp 345
..... 1312, 1325, 1357	\l_draw_path_arc_start_fp 345
\g_draw_id_int 1261, 1272	_draw_path_curveto:nnnn
_draw_if_recursion_tail_stop_- 302, 305, 309
do:Nn 9, 9, 1537	_draw_path_curveto:nnnnnn
\l_draw_layer_close_bool 240, 270,
..... 86, 96, 98, 115	276, 298, 316, 446, 513, 523, 533, 543
\l_draw_layer_main_box	\c_draw_path_curveto_a_fp 302
..... 128, 129, 1259, 1288	\c_draw_path_curveto_b_fp 302
\l_draw_layer_tl 84, 95, 99	_draw_path_ellipse:nnnnnn
\g_draw_layers_clist 87 489, 492, 497
_draw_layers_insert: 122, 122, 1295	_draw_path_ellipse_arci:nnnnnn
_draw_layers_restore: 139, 150, 1410 489, 503, 511
_draw_layers_save: . 139, 139, 1406	_draw_path_ellipse_arci:nnnnnn
\g_draw_linewidth_dim 489, 504, 521
... 732, 1369, 1375, 1740, 1745, 1746	_draw_path_ellipse_arci:nnnnnn
\l_draw_linewidth_dim 489, 505, 531
..... 1362, 1369, 1375	_draw_path_ellipse_arci:nnnnnn
\l_draw_main_box 489, 506, 541
1259, 1273, 1306, 1309, 1314, 1319,	\c_draw_path_ellipse_fp 489
1321, 1322, 1327, 1332, 1334, 1340	_draw_path_grid_auxi:nnnnnn
\l_draw_matrix_a_fp 591, 595, 601, 607
. 42, 1207, 1239, 1777, 1785, 1799,	_draw_path_grid_auxii:nnnnnn
1805, 1839, 1841, 1865, 1874, 1881 591, 604, 605, 608
\l_draw_matrix_active_bool	_draw_path_grid_auxiii:nnnnnn
560, 1202, 1234, 1776, 1810, 1811, 1857 591, 611, 612, 614
\l_draw_matrix_b_fp	_draw_path_grid_auxiiii:nnnnnn 591
. 43, 1213, 1244, 1777, 1786, 1800,	_draw_path_grid_auxiv:nnnnnnnn
1806, 1840, 1842, 1866, 1876, 1877 591, 616, 621, 640
\l_draw_matrix_c_fp	\g_draw_path_lastx_dim
. 44, 1208, 1240, 1777, 1787, 1801, 178, 215, 320, 453, 459, 749, 777
1807, 1839, 1841, 1866, 1878, 1879	\l_draw_path_lastx_dim 737, 749, 777
\l_draw_matrix_d_fp	\g_draw_path_lasty_dim
. 45, 1214, 1245, 1780, 1788, 1802, 178, 216, 327, 454, 460, 750, 778
1808, 1840, 1842, 1865, 1875, 1880	\l_draw_path_lasty_dim 737, 750, 778
_draw_path_arc:nnnn . 345, 351, 358	_draw_path_lineto:nn
_draw_path_arc:nnNnn 240, 255, 258, 292
..... 345, 361, 362, 364	_draw_path_mark_corner:
\c_draw_path_arc_60_fp 345	231, 231, 260, 269, 286, 297, 315, 386
\c_draw_path_arc_90_fp 345	_draw_path_moveto:nn
_draw_path_arc_add:nnnn 345 240, 243, 246, 290, 501, 509
_draw_path_arc_aux_add:nn	_draw_path_rectangle:nnnn
..... 452, 458, 470, 475 554, 562, 567
_draw_path_arc_auxi:nnnnNnn ...	_draw_path_rectangle_corners:nnnn
..... 345, 372, 379, 387, 394, 420 583
_draw_path_arc_auxii:nnnnnn	_draw_path_rectangle_corners:nnnnn
..... 345, 398, 421 586, 589
_draw_path_arc_auxiii:nn	_draw_path_rectangle_rounded:nnnn
..... 345, 425, 450 554, 561, 574
_draw_path_arc_auxiv:nnnn	_draw_path_reset_limits:
..... 345, 431, 456 184, 206, 658, 757, 1276
_draw_path_arc_auxv:nn 345, 441, 462	

\l__draw_path_tmp_tl	__draw_point_interpolate_curve_-
.... 175 , 423 , 446 , 465 , 469 , 473 , 477	auxii:nnnnnnnnn . 1066 , 1070 , 1078
\l__draw_path_tmpa_fp	__draw_point_interpolate_curve_-
..... 175 , 311 , 321 , 333	auxiii:nnnnnn . 1073 , 1084 , 1092
\l__draw_path_tmpb_fp	__draw_point_interpolate_curve_-
..... 175 , 312 , 328 , 337	auxiv:nnnnnn 1086 , 1087 , 1088 , 1093
__draw_path_update_last:nn	__draw_point_interpolate_curve_-
..... 213 , 213 , 250 , 263 , 282 , 572	auxv:nnw 1095 , 1099 , 1106
__draw_path_update_limits:nn ...	__draw_point_interpolate_curve_-
..... 24 , 27 , 30 , 33 , 184 ,	auxvi:n 1090 , 1111
184 , 248 , 261 , 278 , 279 , 280 , 569 , 570	__draw_point_interpolate_curve_-
__draw_path_use:n . 646 , 649 , 660 , 662	auxvii:nnnnnnnn 1112 , 1113
__draw_path_use_action_draw: ...	__draw_point_interpolate_curve_-
..... 646 , 706	auxviii:nnnnnn . 1115 , 1122 , 1127
__draw_path_use_action_fillstroke:	__draw_point_interpolate_-
..... 646 , 710	distance:nnnn 1011 , 1014
\l__draw_path_use_bb_bool 644	__draw_point_interpolate_-
\l__draw_path_use_clear_bool 644 , 703	distance:nnnnn
\l__draw_path_use_clip_bool 1008 , 1018 , 1023 , 1025
..... 641 , 664 , 682	__draw_point_interpolate_-
\l__draw_path_use_fill_bool	distance:nnnnnn 1008
..... 641 , 665 , 687 , 692 , 698 , 712	__draw_point_interpolate_line_-
__draw_path_use_stroke_bb:	aux:nnnnnn 993 , 996 , 999 , 1004
..... 646 , 680 , 715	__draw_point_interpolate_line_-
__draw_path_use_stroke_bb_-	aux:nnnnnn . 993 , 1001 , 1005 , 1007
aux:NnN 646 , 717 , 718 , 719 , 720 , 722	__draw_point_intersect_circles_-
\l__draw_path_use_stroke_bool ...	auxi:nnnnnnnn 896 , 899 , 902
641 , 666 , 679 , 688 , 693 , 699 , 708 , 713	__draw_point_intersect_circles_-
\g__draw_path_xmax_dim	auxii:nnnnnnnn . 896 , 904 , 907 , 914
..... 180 , 186 , 187 , 208 , 751 , 773	__draw_point_intersect_circles_-
\l__draw_path_xmax_dim . 737 , 751 , 773	auxiii:nnnnnnnn . 896 , 909 , 915 , 921
\g__draw_path_xmin_dim	__draw_point_intersect_circles_-
..... 180 , 188 , 189 , 209 , 752 , 774	auxiv:nnnnnnnn . 896 , 917 , 922 , 929
\l__draw_path_xmin_dim . 737 , 752 , 774	__draw_point_intersect_circles_-
\g__draw_path_ymax_dim	auxv:nnnnnnnnnn . 896 , 924 , 930 , 936
..... 180 , 190 , 191 , 210 , 753 , 775	__draw_point_intersect_circles_-
\l__draw_path_ymax_dim . 737 , 753 , 775	auxvi:nnnnnnnnnn . 896 , 932 , 937 , 945
\g__draw_path_ymin_dim	__draw_point_intersect_circles_-
..... 180 , 192 , 193 , 211 , 754 , 776	auxvii:nnnnnnnn . 896 , 939 , 946 , 951
\l__draw_path_ymin_dim . 737 , 754 , 776	__draw_point_intersect_line_-
__draw_point_interpolate_-	circle_auxi:nnnnnnnn 952 , 955 , 958
arcaxes_auxi:nnnnnnnnnn	__draw_point_intersect_line_-
..... 1026 , 1029 , 1032	circle_auxii:nnnnnnnnnn
__draw_point_interpolate_- 952 , 960 , 963 , 971
arcaxes_auxii:nnnnnnnnnn	__draw_point_intersect_line_-
..... 1026 , 1034 , 1037 , 1043	circle_auxiii:nnnnnnnnnn
__draw_point_interpolate_- 952 , 965 , 972 , 979
arcaxes_auxiii:nnnnnnnn 1026 , 1039 , 1044 , 1051	__draw_point_intersect_line_-
__draw_point_interpolate_-	circle_auxiv:nnnnnnnnnn
arcaxes_auxiv:nnnnnnnnnn 1026 , 1046 , 1052 , 1057 952 , 974 , 980 , 986
__draw_point_interpolate_curve_-	__draw_point_intersect_line_-
auxi:nnnnnnnnnn 1061 , 1064	circle_auxv:nnnnnn 952 , 982 , 987 , 992
	__draw_point_intersect_lines:nnnnnn
 871

```

\__draw_point_intersect_lines:nnnnnnnn
..... 871, 874, 877
\__draw_point_intersect_lines_-
aux:nnnnnn ..... 871, 879, 887, 895
\__draw_point_process:nn .....
23, 26, 29, 32, 242, 254, 290, 292,
424, 440, 790, 790, 855, 1010, 1016,
1142, 1197, 1229, 1815, 1847, 1893
\__draw_point_process:nnn 304, 430,
556, 585, 593, 790, 800, 898, 995, 1891
\__draw_point_process:nnnn .....
... 267, 295, 491, 790, 811, 954, 1028
\__draw_point_process:nnnnn .....
..... 790, 824, 873, 1060
\__draw_point_process_auxi:nn ...
..... 790, 792, 796
\__draw_point_process_auxii:nw ..
..... 790, 797, 798
\__draw_point_process_auxiii:nnn
..... 790, 802, 807
\__draw_point_process_auxiv:nw ..
..... 790, 808, 809
\__draw_point_process_auxv:nnnn .
..... 790, 813, 819
\__draw_point_process_auxvi:nw ..
..... 790, 820, 821
\__draw_point_process_auxvii:nnnnn
..... 790, 826, 833
\__draw_point_process_auxviii:nw
..... 790, 835, 838
\__draw_point_to_dim:n .....
..... 841, 842, 843, 845
\__draw_point_to_dim:w . 841, 844, 846
\__draw_point_transform:nn .....
..... 1195, 1198, 1200
\__draw_point_transform_noshift:n
..... 427, 443, 494, 495, 1227, 1227
\__draw_point_transform_noshift:nn
..... 1227, 1230, 1232
\__draw_point_unit_vector:nn ...
..... 854, 855, 856
\__draw_point_unit_vector:nnn ...
..... 854, 858, 862
\__draw_point_vec:nn .....
..... 1152, 1153, 1154, 1162
\__draw_point_vec:nnn .....
..... 1152, 1165, 1168, 1181
\__draw_point_vec_polar:nnn .. 1182
\__draw_reset_bb: .....
..... 1262, 1262, 1275, 1390
\__draw_scope_bb_begin: .....
..... 1383, 1383, 1402
\__draw_scope_bb_end: 1383, 1392, 1412

\__draw_softpath_add:n .....
..... 770, 1420, 1420, 1422, 1439,
1448, 1457, 1462, 1472, 1480, 1735
\c__draw_softpath_arc_fp .....
..... 1514, 1677, 1681, 1686, 1690
\__draw_softpath_clear: .....
. 657, 704, 762, 766, 1280, 1423, 1428
\__draw_softpath_close_op:nn ...
.. 1441, 1485, 1485, 1576, 1612, 1714
\__draw_softpath_closepath: .....
..... 287, 508, 1437
\l__draw_softpath_corneri_dim ...
..... 1508, 1561, 1564, 1650
\l__draw_softpath_cornerii_dim ..
..... 1508, 1562, 1565, 1641
\g__draw_softpath_corners_bool ..
761, 768, 1419, 1431, 1482, 1517, 1533
\l__draw_softpath_corners_bool ..
..... 737, 760, 769
\l__draw_softpath_curve_end_tl ..
..... 1507, 1638, 1657, 1706, 1717
\__draw_softpath_curveto:nnnnnn .
..... 281, 1437, 1446
\__draw_softpath_curveto_opi:nn .
.. 1450, 1485, 1487, 1573, 1611, 1674
\__draw_softpath_curveto_-
opi:nnNnnNnn ..... 1485, 1488, 1489
\__draw_softpath_curveto_opii:nn
..... 1451, 1485, 1491, 1492, 1683
\__draw_softpath_curveto_-
opiii:nn 1452, 1485, 1493, 1494, 1692
\l__draw_softpath_first_tl .....
..... 1508, 1524, 1541,
1542, 1552, 1571, 1572, 1598, 1602
\l__draw_softpath_internal_tl ...
..... 1418, 1425, 1426, 1526, 1528
\g__draw_softpath_lastx_dim .....
..... 755, 771, 1433, 1442, 1466
\l__draw_softpath_lastx_dim .....
..... 743, 755, 771
\l__draw_softpath_lastx_fp .....
.. 1508, 1522, 1543, 1591, 1653, 1660
\g__draw_softpath_lasty_dim .....
..... 756, 772, 1433, 1443, 1467
\l__draw_softpath_lasty_dim .....
..... 744, 756, 772
\l__draw_softpath_lasty_fp .....
.. 1508, 1523, 1544, 1592, 1654, 1661
\__draw_softpath_lineto:nn .....
..... 262, 1437, 1455
\__draw_softpath_lineto_op:nn ...
.. 1458, 1485, 1495, 1579, 1610, 1723
\g__draw_softpath_main_tl .....
758, 1417, 1421, 1425, 1430, 1526, 1734

```

\l__draw_softpath_main_tl	__draw_softpath_roundpoint_-
. 19, 758, 770, 1505,	op:nn . . 1481, 1485, 1499, 1538, 1620
1520, 1547, 1549, 1730, 1732, 1735	__draw_softpath_use: 681, 1423, 1423
\g__draw_softpath_move_bool	\l__draw_stroke_color_tl 1362
. 1435, 1464	\l__draw_tmp_box 13, 37, 48,
\l__draw_softpath_move_tl	52, 54, 55, 56, 57, 63, 65, 66, 67, 68, 69
. 1508, 1525,	\l__draw_tmp_seq 1748
1548, 1551, 1599, 1701, 1724, 1731	__draw_tranform_triangle:nnnnnn
__draw_softpath_moveto:nn 1894, 1899
. 249, 1437, 1460	__draw_transform:nnnn
__draw_softpath_moveto_op:nn 1823, 1827, 1834
. 1463, 1485, 1497, 1545, 1703	__draw_transform_invert:n
\l__draw_softpath_part_tl 1855, 1859, 1872, 1883
. 1506, 1521,	__draw_transform_rotate:n
1550, 1553, 1555, 1589, 1644, 1733 1925, 1926, 1927, 1933
__draw_softpath_rectangle:nnnn .	__draw_transform_rotate:nn
. 571, 1437, 1470 1925, 1929, 1934, 1936
__draw_softpath_rectangle_-	__draw_transform_shift:nn
opi:nn 1474, 1485, 1500 1823, 1848, 1850
__draw_softpath_rectangle_-	__draw_transform_shift_absolute:nn
opi:nnNnn 1485, 1501, 1502 1797, 1816, 1818
__draw_softpath_rectangle_-	__draw_vec:nn
opii:nn 1475, 1485, 1504 1134, 1135, 1137, 1139, 1140
__draw_softpath_round_action:nn	__draw_vec:nnn 1134, 1142, 1144
. 1515, 1539, 1559	\g__draw_xmax_dim 196,
__draw_softpath_round_action:Nnn	197, 1254, 1264, 1300, 1307, 1386, 1394
. 1515, 1567, 1569	\l__draw_xmax_dim . . . 1379, 1386, 1394
__draw_softpath_round_action_-	\g__draw_xmin_dim
close: 1515, 1577, 1595 198, 199, 1254, 1265, 1298,
__draw_softpath_round_action_-	1301, 1307, 1316, 1329, 1387, 1395
curveto:NnnNnn . . 1515, 1574, 1586	\l__draw_xmin_dim . . . 1379, 1387, 1395
__draw_softpath_round_calc:NnnNnn	\l__draw_xshift_dim 50, 1209,
. 1515, 1615, 1630, 1636, 1699	1223, 1777, 1792, 1820, 1852, 1886
__draw_softpath_round_calc:nnnnnn	\l__draw_xvec_x_dim
. 1515, 1647, 1664, 1669 1128, 1158, 1172, 1190
__draw_softpath_round_calc:nnnnw	\l__draw_xvec_y_dim . . 1128, 1159, 1176
. 1515, 1666, 1670	\g__draw_ymax_dim . . 200, 201, 1254,
__draw_softpath_round_close:nn .	1266, 1302, 1323, 1341, 1388, 1396
. 1515, 1601, 1695	\l__draw_ymax_dim . . . 1379, 1388, 1396
__draw_softpath_round_close:w . .	\g__draw_ymin_dim
. 1515, 1705, 1716, 1727 202, 203, 1254, 1267,
__draw_softpath_round_corners: .	1303, 1318, 1323, 1337, 1389, 1397
. 676, 1515, 1515	\l__draw_ymin_dim . . . 1379, 1389, 1397
__draw_softpath_round_end:	\l__draw_yshift_dim 51, 1215,
. 1515, 1537, 1728	1223, 1777, 1793, 1821, 1853, 1887
__draw_softpath_round_lookahead:NnnNnn	\l__draw_yvec_x_dim . . 1128, 1158, 1173
. 1515, 1580, 1593, 1606	\l__draw_yvec_y_dim
__draw_softpath_round_loop:Nnn 1128, 1159, 1177, 1191
. . . 1515, 1527, 1535, 1556, 1566,	\l__draw_zvec_x_dim 1128, 1174
1581, 1604, 1616, 1622, 1631, 1713	\l__draw_zvec_y_dim 1128, 1178
__draw_softpath_round_roundpoint:NnnNnnNnn	
. 1515, 1621, 1627	
__draw_softpath_round_roundpoint:nn . .	
. 235, 1437, 1478, 1484	
	E
	\end 169, 784

exp commands:	\group_end: 58, 70, 117, 120, 779, 1310, 1376, 1398, 1531, 1760
\exp_after:wN	
446, 464, 1527, 1601, 1704, 1715, 1724	
\exp_args:Nf 792, 858	H
\exp_args:Nff 802	hbox commands:
\exp_args:Nfff 813	\hbox_gset:Nw 101
\exp_args:Nffff 826	\hbox_gset_end: 118
\exp_args:NNNV 1293	\hbox_set:Nn . . . 37, 48, 63, 1314, 1327
\exp_args:Nx 1764	\hbox_set:Nw 1273, 1288
\exp_not:N	\hbox_set_end: 1293, 1297
1646, 1674, 1683, 1692, 1701, 1704,	
1705, 1706, 1711, 1715, 1716, 1717	I
\exp_not:n 1356	int commands:
	\int_gincr:N 1272
F	\int_if_odd:nTF 941, 976
fp commands:	\int_new:N 1261
\fp_compare:nNnTF 360, 370, 864	
\fp_compare_p:nNn	K
. 1805, 1806, 1807, 1808	kernel internal commands:
\fp_const:Nn	__kernel_kern:n 50
. 343, 344, 482, 483, 551, 1514	__kernel_quark_new_test:N 9
\fp_eval:n 352, 353, 374, 381,	
390, 842, 850, 859, 880, 881, 882,	M
883, 884, 885, 905, 910, 911, 918,	mode commands:
925, 926, 933, 940, 942, 961, 966,	\mode_leave_vertical: 1308
967, 968, 975, 983, 996, 1001, 1019,	msg commands:
1035, 1040, 1047, 1048, 1067, 1074,	\msg_error:nnn 78, 109, 110, 673
1096, 1097, 1116, 1117, 1118, 1119,	\msg_new:nnn 164
1153, 1166, 1185, 1764, 1828, 1829,	\msg_new:nnnn 161, 166, 781
1830, 1831, 1861, 1926, 1930, 1931	
\fp_new:N 176, 177, 480,	P
481, 1508, 1509, 1777, 1778, 1779, 1780	\pgfextractx 21
\fp_set:Nn 311, 312, 366, 367,	\pgfextracty 21
447, 448, 1543, 1544, 1591, 1592,	\pgfgetlastxy 21
1660, 1661, 1785, 1788, 1799, 1800,	\pgfgettransform 50
1801, 1802, 1874, 1876, 1878, 1880	\pgfgettransformentries 50
\fp_to_decimal:N 373, 380, 388	\pgfinnerlinewidth 48
\fp_to_dim:n 318, 325, 332,	\pgflowlevel 50
336, 354, 355, 401, 410, 478, 502,	\pgflowlevelsynccm 50
514, 515, 516, 517, 518, 519, 524,	\pgfpatharcto 6
525, 526, 527, 528, 529, 534, 535,	\pgfpatharctoprecomputed 6
536, 537, 538, 539, 544, 545, 546,	\pgfpathcosine 6
547, 548, 549, 617, 618, 1348, 1676,	\pgfpathcurvebetweentime 6
1680, 1685, 1689, 1745, 1753, 1758	\pgfpathcurvebetweentimecontinue 6
\fp_use:N 42, 43, 44, 45, 551	\pgfpathparabola 6
\fp_while_do:nNnn 368	\pgfpathsine 6
\fp_zero:N 1522, 1523, 1786, 1787	\pgfpointadd 20
\c_one_fp 1805, 1808	\pgfpointborderellipse 21
\c_zero_fp 864, 1806, 1807	\pgfpointborderrectangle 21
	\pgfpointcylindrical 21
G	\pgfpointdiff 20
group commands:	\pgfpointorigin 20
\group_begin: 36, 62, 92,	\pgfpointscale 20
103, 748, 1271, 1368, 1385, 1519, 1750	\pgfpointspherical 21
	\pgfqpoint 21

