

File I

Implementation

1 l3draw implementation

```
1 <*package>
2 <@@=draw>
3 \ProvidesExplPackage{l3draw}{2023-11-01}{}
4 {L3 Experimental core drawing support}
```

1.1 Internal auxiliaries

`\s__draw_mark` Internal scan marks.

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

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

`\q__draw_recursion_tail` Internal recursion quarks.

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

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

`__draw_if_recursion_tail_stop_do:Nn` Functions to query recursion quarks.

```
9 \__kernel_quark_new_test:N \__draw_if_recursion_tail_stop_do:Nn
```

(End of definition for __draw_if_recursion_tail_stop_do:Nn.)

Everything else is in the sub-files!

```
10 </package>
```

2 l3draw-boxes implementation

```
11 <*package>
```

```
12 <@@=draw>
```

Inserting boxes requires us to “interrupt” the drawing state, so is closely linked to scoping. At the same time, there are a few additional features required to make text work in a flexible way.

`\l__draw_tmp_box`

```
13 \box_new:N \l__draw_tmp_box
```

(End of definition for \l__draw_tmp_box.)

`\draw_box_use:N` Before inserting a box, we need to make sure that the bounding box is being updated
`__draw_box_use:Nnnnn` correctly. As drawings track transformations as a whole, rather than as separate operations, we do the insertion using an almost-row matrix. The process is split into two so that coffins are also supported.

```
14 \cs_new_protected:Npn \draw_box_use:N #1
```

```
15 {
```

```

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

```

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

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

```

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

```

```

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

```

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

```
72 </package>
```

3 I3draw-layers implementation

```
73 <*package>
```

```
74 <@@=draw>
```

3.1 User interface

`\draw_layer_new:n`

```

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

```

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

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

```

84 \tl_new:N \l__draw_layer_tl
85 \tl_set:Nn \l__draw_layer_tl { main }

```

(End of definition for `\l__draw_layer_tl`.)

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

```
86 \bool_new:N \l__draw_layer_close_bool
```

(End of definition for `\l__draw_layer_close_bool`.)

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

```

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

```

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

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

```

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

```

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

3.2 Internal cross-links

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

```

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

```

```

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

```

(End of definition for `__draw_layers_insert:`.)

```

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

```

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

```

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

```

4 l3draw-paths implementation

```

173 <*package>
174 <@@=draw>

```

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

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

`\l__draw_path_tmp_tl` Scratch space.

```

\l__draw_path_tmpa_fp 175 \tl_new:N \l__draw_path_tmp_tl
\l__draw_path_tmpb_fp 176 \fp_new:N \l__draw_path_tmpa_fp
177 \fp_new:N \l__draw_path_tmpb_fp

```

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

4.1 Tracking paths

`\g__draw_path_lastx_dim` The last point visited on a path.

```

\g__draw_path_lasty_dim 178 \dim_new:N \g__draw_path_lastx_dim
179 \dim_new:N \g__draw_path_lasty_dim

```

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

`\g__draw_path_xmax_dim` The limiting size of a path.

```

\g__draw_path_xmin_dim 180 \dim_new:N \g__draw_path_xmax_dim
\g__draw_path_ymax_dim 181 \dim_new:N \g__draw_path_xmin_dim
\g__draw_path_ymin_dim 182 \dim_new:N \g__draw_path_ymax_dim
183 \dim_new:N \g__draw_path_ymin_dim

```

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

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

```

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

```

```

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

```

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

__draw_path_update_last:nn A simple auxiliary to avoid repetition.

```

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

```

(End of definition for __draw_path_update_last:nn.)

4.2 Corner arcs

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

\l__draw_corner_xarc_dim The two arcs in use.

```

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

```

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

\l__draw_corner_arc_bool A flag to speed up the repeated checks.

```

220 \bool_new:N \l__draw_corner_arc_bool

```

(End of definition for \l__draw_corner_arc_bool.)

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

```

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

```

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

```

\__draw_path_mark_corner: Mark up corners for arc post-processing.
231 \cs_new_protected:Npn \__draw_path_mark_corner:
232   {
233     \bool_if:NT \l__draw_corner_arc_bool
234     {
235       \__draw_softpath_roundpoint:VV
236       \l__draw_corner_xarc_dim
237       \l__draw_corner_yarc_dim
238     }
239   }

```

(End of definition for `__draw_path_mark_corner:.`)

4.3 Basic path constructions

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

```

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

```



```

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

```

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

`\draw_path_close:` A simple wrapper.

```

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

```

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

4.4 Canvas path constructions

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

```

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

```

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

4.5 Computed curves

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

`\draw_path_curveto:nn` A quadratic curve with one control point (x_c, y_c) . The two required control points are then

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

`\c__draw_path_curveto_a_fp`
`\c__draw_path_curveto_b_fp`

and

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

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

```

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

```

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

`\draw_path_arc:nnn` Drawing an arc means dividing the total curve required into sections: using Bézier curves we can cover at most 90° at once. To allow for later manipulations, we aim to have roughly equal last segments to the line, with the split set at a final part of 115° .

```

\draw_path_arc:nnnn
\__draw_path_arc:nnNnn
\__draw_path_arc_auxi:nnnnNnn
\__draw_path_arc_auxi: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

```

```

345 \cs_new_protected:Npn \draw_path_arc:nnn #1#2#3
346 { \draw_path_arc:nnnn {#1} {#2} {#3} {#3} }
347 \cs_new_protected:Npn \draw_path_arc:nnnn #1#2#3#4
348 {
349   \use:e

```

```

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

```

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

```

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

```

```

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

```

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

```

421 \cs_new_protected:Npn \__draw_path_arc_auxii: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:nnnnnn \l__draw_path_tmp_tl

```

```

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

```

The first control point.

```

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

```

The end point: simple arithmetic.

```

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

```

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

```

462 \cs_new_protected:Npn \__draw_path_arc_auxv:nn #1#2
463 {
464   \exp_after:wN \__draw_path_arc_auxvi:nn
465   \l__draw_path_tmp_tl {#1} {#2}
466 }
467 \cs_new_protected:Npn \__draw_path_arc_auxvi:nn #1#2#3#4#5#6
468 {
469   \tl_set:Nn \l__draw_path_tmp_tl { {#1} {#2} }
470   \__draw_path_arc_aux_add:nn
471   { #5 + #3 }
472   { #6 + #4 }
473   \tl_put_right:Nn \l__draw_path_tmp_tl { {#3} {#4} }
474 }
475 \cs_new_protected:Npn \__draw_path_arc_aux_add:nn #1#2
476 {
477   \tl_put_right:Ne \l__draw_path_tmp_tl
478   { { \fp_to_dim:n {#1} } { \fp_to_dim:n {#2} } }
479 }
480 \fp_new:N \l__draw_path_arc_delta_fp
481 \fp_new:N \l__draw_path_arc_start_fp
482 \fp_const:cn { c__draw_path_arc_90_fp } { 4/3 * (sqrt(2) - 1) }
483 \fp_const:cn { c__draw_path_arc_60_fp } { 4/3 * tand(15) }

```

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

`\draw_path_arc_axes:nnnn` A simple wrapper.

```

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

```

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


```

541     { \fp_to_dim:n { #2 - #6 } }
542   }
543 \cs_new:Npn \__draw_path_ellipse_arciv:nnnnnn #1#2#3#4#5#6
544   {
545     \__draw_path_curveto:nnnnnn
546     { \fp_to_dim:n { #1 + #3 * \c__draw_path_ellipse_fp - #5 } }
547     { \fp_to_dim:n { #2 + #4 * \c__draw_path_ellipse_fp - #6 } }
548     { \fp_to_dim:n { #1 + #3 - #5 * \c__draw_path_ellipse_fp } }
549     { \fp_to_dim:n { #2 + #4 - #6 * \c__draw_path_ellipse_fp } }
550     { \fp_to_dim:n { #1 + #3 } }
551     { \fp_to_dim:n { #2 + #4 } }
552   }
553 \fp_const:Nn \c__draw_path_ellipse_fp { \fp_use:c { c__draw_path_arc_90_fp } }

```

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

\draw_path_circle:nn A shortcut.

```

554 \cs_new_protected:Npn \draw_path_circle:nn #1#2
555   { \draw_path_ellipse:nnn {#1} { #2 , Opt } { Opt , #2 } }

```

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

4.6 Rectangles

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

```

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

```

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

```

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

```

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

4.7 Grids

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

```

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

```



```

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

```

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

4.8 Using paths

```

\l__draw_path_use_clip_bool
\l__draw_path_use_fill_bool
  \l__draw_path_use_stroke_bool

```

Actions to pass to the driver.

```

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

```

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

```

\l__draw_path_use_bb_bool
\l__draw_path_use_clear_bool

```

Actions handled at the macro layer.

```

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

```

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

```

\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

```

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.

```

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

```

Map over the actions and set up the data: mainly just booleans, but with the possibility to cover more complex cases. The business end of the function is a series of checks on the various flags, then taking the appropriate action(s).

```

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

```

```

715 \bool_set_true:N \l__draw_path_use_stroke_bool
716 }

```

Where the path is relevant to size and is stroked, we need to allow for the part which overlaps the edge of the bounding box.

```

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

```

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

4.9 Scoping paths

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

```

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

```

(End of definition for \l__draw_path_lastx_dim and others.)

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

```

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

```

```

753     \dim_set_eq:NN \l__draw_path_xmax_dim \g__draw_path_xmax_dim
754     \dim_set_eq:NN \l__draw_path_xmin_dim \g__draw_path_xmin_dim
755     \dim_set_eq:NN \l__draw_path_ymax_dim \g__draw_path_ymax_dim
756     \dim_set_eq:NN \l__draw_path_ymin_dim \g__draw_path_ymin_dim
757     \dim_set_eq:NN \l__draw_softpath_lastx_dim \g__draw_softpath_lastx_dim
758     \dim_set_eq:NN \l__draw_softpath_lasty_dim \g__draw_softpath_lasty_dim
759     \__draw_path_reset_limits:
760     \__draw_softpath_save:
761   }
762   \cs_new_protected:Npn \draw_path_scope_end:
763   {
764     \__draw_softpath_restore:
765     \dim_gset_eq:NN \g__draw_softpath_lastx_dim \l__draw_softpath_lastx_dim
766     \dim_gset_eq:NN \g__draw_softpath_lasty_dim \l__draw_softpath_lasty_dim
767     \dim_gset_eq:NN \g__draw_path_xmax_dim \l__draw_path_xmax_dim
768     \dim_gset_eq:NN \g__draw_path_xmin_dim \l__draw_path_xmin_dim
769     \dim_gset_eq:NN \g__draw_path_ymax_dim \l__draw_path_ymax_dim
770     \dim_gset_eq:NN \g__draw_path_ymin_dim \l__draw_path_ymin_dim
771     \dim_gset_eq:NN \g__draw_path_lastx_dim \l__draw_path_lastx_dim
772     \dim_gset_eq:NN \g__draw_path_lasty_dim \l__draw_path_lasty_dim
773   \group_end:
774   }

```

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

```

775 \msg_new:nnnn { draw } { invalid-path-action }
776 { Invalid-action~'#1'~for-path. }
777 { Paths~can~be~used~with~actions~'draw',~'clip',~'fill'~or~'stroke'. }
778 % \end{macrocode}
779 %
780 % \begin{macrocode}
781 </package>

```

5 I3draw-points implementation

```

782 <*package>
783 <@@=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

`__draw_point_process:nn` 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

```

```

784 \cs_new:Npn \__draw_point_process:nn #1#2
785   {
786     \exp_args:Nf \__draw_point_process_auxi:nn
787     { \draw_point:n {#2} }
788     {#1}
789   }
790 \cs_new:Npn \__draw_point_process_auxi:nn #1#2
791   { \__draw_point_process_auxii:nw {#2} #1 \s__draw_stop }
792 \cs_new:Npn \__draw_point_process_auxii:nw #1 #2 , #3 \s__draw_stop
793   { #1 {#2} {#3} }
794 \cs_new:Npn \__draw_point_process:nnn #1#2#3
795   {
796     \exp_args:Nff \__draw_point_process_auxiii:nnn
797     { \draw_point:n {#2} }
798     { \draw_point:n {#3} }
799     {#1}
800   }
801 \cs_new:Npn \__draw_point_process_auxiii:nnn #1#2#3
802   { \__draw_point_process_auxiv:nw {#3} #1 \s__draw_mark #2 \s__draw_stop }
803 \cs_new:Npn \__draw_point_process_auxiv:nw #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_stop
804   { #1 {#2} {#3} {#4} {#5} }
805 \cs_new:Npn \__draw_point_process:nnnn #1#2#3#4
806   {
807     \exp_args:Nfff \__draw_point_process_auxv:nnnn
808     { \draw_point:n {#2} }
809     { \draw_point:n {#3} }
810     { \draw_point:n {#4} }
811     {#1}
812   }
813 \cs_new:Npn \__draw_point_process_auxv:nnnn #1#2#3#4
814   { \__draw_point_process_auxvi:nw {#4} #1 \s__draw_mark #2 \s__draw_mark #3 \s__draw_stop }
815 \cs_new:Npn \__draw_point_process_auxvi:nw
816   #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_mark #6 , #7 \s__draw_stop
817   { #1 {#2} {#3} {#4} {#5} {#6} {#7} }
818 \cs_new:Npn \__draw_point_process:nnnnn #1#2#3#4#5
819   {

```

```

820 \exp_args:Nffff \__draw_point_process_auxvii:nnnnn
821 { \draw_point:n {#2} }
822 { \draw_point:n {#3} }
823 { \draw_point:n {#4} }
824 { \draw_point:n {#5} }
825 {#1}
826 }
827 \cs_new:Npn \__draw_point_process_auxvii:nnnnn #1#2#3#4#5
828 {
829 \__draw_point_process_auxviii:nw
830 {#5} #1 \s__draw_mark #2 \s__draw_mark #3 \s__draw_mark #4 \s__draw_stop
831 }
832 \cs_new:Npn \__draw_point_process_auxviii:nw
833 #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_mark #6 , #7 \s__draw_mark #8 , #9 \s__draw_stop
834 { #1 {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9} }

```

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

5.2 Basic points

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

```

\__draw_point_to_dim:n 835 \cs_new:Npn \draw_point:n #1
\__draw_point_to_dim:f 836 { \__draw_point_to_dim:f { \fp_eval:n {#1} } }
\__draw_point_to_dim:w 837 \cs_new:Npn \__draw_point_to_dim:n #1
838 { \__draw_point_to_dim:w #1 }
839 \cs_generate_variant:Nn \__draw_point_to_dim:n { f }
840 \cs_new:Npn \__draw_point_to_dim:w ( #1 , ~ #2 ) { #1pt , #2pt }

```

5.3 Polar co-ordinates

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

```

\draw_point_polar:nn 841 \cs_new:Npn \draw_point_polar:nn #1#2
\draw_point_polar:nnn 842 { \draw_point_polar:nn {#1} {#1} {#2} }
\__draw_draw_polar:nnn 843 \cs_new:Npn \draw_point_polar:nnn #1#2#3
\__draw_draw_polar:fnn 844 { \__draw_draw_polar:fnn { \fp_eval:n {#3} } {#1} {#2} }
845 \cs_new:Npn \__draw_draw_polar:nnn #1#2#3
846 { \draw_point:n { cosd(#1) * (#2) , sind(#1) * (#3) } }
847 \cs_generate_variant:Nn \__draw_draw_polar:nnn { f }

```

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.*

```

\draw_point_unit_vector:n
\__draw_point_unit_vector:nn
\__draw_point_unit_vector:nnn

```

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

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

```

848 \cs_new:Npn \draw_point_unit_vector:n #1
849 { \__draw_point_process:nn { \__draw_point_unit_vector:nn } {#1} }

```

```

850 \cs_new:Npn \__draw_point_unit_vector:nn #1#2
851 {
852   \exp_args:Nf \__draw_point_unit_vector:nnn
853   { \fp_eval:n { (sqrt(#1 * #1 + #2 * #2)) } }
854   {#1} {#2}
855 }
856 \cs_new:Npn \__draw_point_unit_vector:nnn #1#2#3
857 {
858   \fp_compare:nNnTF {#1} = \c_zero_fp
859   { Opt, 1pt }
860   {
861     \draw_point:n
862     { ( #2 , #3 ) / #1 }
863   }
864 }

```

5.5 Intersection calculations

The intersection point P between a line joining points (x_1, y_1) and (x_2, y_2) with a second line joining points (x_3, y_3) and (x_4, y_4) can be calculated using the formulae

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

and

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

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

```

865 \cs_new:Npn \draw_point_intersect_lines:nnnn #1#2#3#4
866 {
867   \__draw_point_process:nnnnn
868   { \__draw_point_intersect_lines:nnnnnnn }
869   {#1} {#2} {#3} {#4}
870 }

```

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

```

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

```

so now just have to do all of the calculation.

```

871 \cs_new:Npn \__draw_point_intersect_lines:nnnnnnn #1#2#3#4#5#6#7#8
872 {
873   \__draw_point_intersect_lines_aux:ffffff
874   { \fp_eval:n { #1 * #4 - #2 * #3 } }
875   { \fp_eval:n { #5 * #8 - #6 * #7 } }
876   { \fp_eval:n { #1 - #3 } }
877   { \fp_eval:n { #5 - #7 } }
878   { \fp_eval:n { #2 - #4 } }
879   { \fp_eval:n { #6 - #8 } }
880 }
881 \cs_new:Npn \__draw_point_intersect_lines_aux:nnnnnn #1#2#3#4#5#6
882 {
883   \draw_point:n
884   {
885     ( #2 * #3 - #1 * #4 , #2 * #5 - #1 * #6 )
886     / ( #4 * #5 - #6 * #3 )
887   }
888 }
889 \cs_generate_variant:Nn \__draw_point_intersect_lines_aux:nnnnnn { fffffff }

```

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

```

\draw_point_intersect_circles:nnmn
__draw_point_intersect_circles_auxi:nnnnnnn
__draw_point_intersect_circles_auxii:nnnnnnn
__draw_point_intersect_circles_auxiii:ffnnnnn
__draw_point_intersect_circles_auxiiii:ffnnnnn
__draw_point_intersect_circles_auxiv:nnnnnnn
__draw_point_intersect_circles_auxv:ffnnnnnnn
__draw_point_intersect_circles_auxvi:ffnnnnnnn
__draw_point_intersect_circles_auxvii:ffnnnnn

```

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

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.

```

890 \cs_new:Npn \draw_point_intersect_circles:nnnnn #1#2#3#4#5
891 {
892   \__draw_point_process:nnn
893   { \__draw_point_intersect_circles_auxi:nnnnnnn {#2} {#4} {#5} }
894   {#1} {#3}

```



```

895 }
896 \cs_new:Npn \__draw_point_intersect_circles_auxi:nnnnnnn #1#2#3#4#5#6#7
897 {
898   \__draw_point_intersect_circles_auxii:ffnnnnn
899   { \fp_eval:n {#1} } { \fp_eval:n {#2} } {#4} {#5} {#6} {#7} {#3}
900 }

```

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.

```

901 \cs_new:Npn \__draw_point_intersect_circles_auxii:nnnnnnn #1#2#3#4#5#6#7
902 {
903   \__draw_point_intersect_circles_auxiii:ffnnnnn
904   { \fp_eval:n { #5 - #3 } }
905   { \fp_eval:n { #6 - #4 } }
906   {#1} {#2} {#3} {#4} {#7}
907 }
908 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxii:nnnnnnn { ff }
909 \cs_new:Npn \__draw_point_intersect_circles_auxiii:nnnnnnn #1#2#3#4#5#6#7
910 {
911   \__draw_point_intersect_circles_auxiv:fnnnnnnn
912   { \fp_eval:n { sqrt( #1 * #1 + #2 * #2 ) } }
913   {#1} {#2} {#3} {#4} {#5} {#6} {#7}
914 }
915 \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.

```

916 \cs_new:Npn \__draw_point_intersect_circles_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
917 {
918   \__draw_point_intersect_circles_auxv:ffnnnnnnn
919   { \fp_eval:n { 1 / #1 } }
920   { \fp_eval:n { #4 * #4 } }
921   {#1} {#2} {#3} {#5} {#6} {#7} {#8}
922 }
923 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxiv:nnnnnnnn { f }
924 \cs_new:Npn \__draw_point_intersect_circles_auxv:nnnnnnnnn #1#2#3#4#5#6#7#8#9
925 {
926   \__draw_point_intersect_circles_auxvi:fnnnnnnn
927   { \fp_eval:n { 0.5 * #1 * ( #2 + #3 * #3 - #6 * #6 ) } }
928   {#1} {#2} {#4} {#5} {#7} {#8} {#9}
929 }
930 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxv:nnnnnnnnn { 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.

```

931 \cs_new:Npn \__draw_point_intersect_circles_auxvi:nnnnnnnn #1#2#3#4#5#6#7#8
932   {
933     \__draw_point_intersect_circles_auxvii:fffnnnn
934     { \fp_eval:n { #1 * #2 } }
935     { \int_if_odd:nTF {#8} { 1 } { -1 } }
936     { \fp_eval:n { sqrt ( #3 - #1 * #1 ) * #2 } }
937     {#4} {#5} {#6} {#7}
938   }
939 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxvi:nnnnnnnn { f }
940 \cs_new:Npn \__draw_point_intersect_circles_auxvii:nnnnnnnn #1#2#3#4#5#6#7
941   {
942     \draw_point:n
943     { #6 + #4 * #1 + #2 * #3 * #5 , #7 + #5 * #1 + -1 * #2 * #3 * #4 }
944   }
945 \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 a circle with center (x_3, y_3) and radius r . We use the intermediate values

```

\draw_point_intersect_line_circle:nnmn
w_point_intersect_line_circle_auxi:nnnnnnnn
_point_intersect_line_circle_auxii:nnnnnnnn
_point_intersect_line_circle_auxiii:fnnnnnnn
_point_intersect_line_circle_auxiiii:nnnnnnnn
_point_intersect_line_circle_auxv:fffnnnnnn
_point_intersect_line_circle_auxvi:nnnnnnnn
_point_intersect_line_circle_auxvii:fffnnnnnn
draw_point_intersect_line_circle_auxviii:nnnnn
draw_point_intersect_line_circle_auxviiii:fnnnn

```

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

in either

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

or

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

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

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

```

946 \cs_new:Npn \draw_point_intersect_line_circle:nnnnn #1#2#3#4#5
947 {
948   \__draw_point_process:nnnn
949   { \__draw_point_intersect_line_circle_auxi:nnnnnnnn {#4} {#5} }
950   {#1} {#2} {#3}
951 }
952 \cs_new:Npn \__draw_point_intersect_line_circle_auxi:nnnnnnnn #1#2#3#4#5#6#7#8
953 {
954   \__draw_point_intersect_line_circle_auxii:fnnnnnnn
955   { \fp_eval:n {#1} } {#3} {#4} {#5} {#6} {#7} {#8} {#2}
956 }

```

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

```

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

```

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

```

957 \cs_new:Npn \__draw_point_intersect_line_circle_auxii:nnnnnnnn #1#2#3#4#5#6#7#8
958 {
959   \__draw_point_intersect_line_circle_auxiii:ffnnnnnn
960   { \fp_eval:n { (#4-#2)*(#4-#2)+(#5-#3)*(#5-#3) } }
961   { \fp_eval:n { 2*((#4-#2)*(#2-#6)+(#5-#3)*(#3-#7)) } }
962   { \fp_eval:n { (#6*#6+#7*#7)+(#2*#2+#3*#3)-(2*(#6*#2+#7*#3))-(#1*#1) } }
963   {#2} {#3} {#4} {#5} {#8}
964 }
965 \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 .

```

966 \cs_new:Npn \__draw_point_intersect_line_circle_auxiii:nnnnnnnn #1#2#3#4#5#6#7#8
967 {
968   \__draw_point_intersect_line_circle_auxiv:ffnnnnnn
969   { \fp_eval:n { #2 * #2 - 4 * #1 * #3 } }
970   { \int_if_odd:nTF {#8} { 1 } { -1 } }
971   {#1} {#2} {#4} {#5} {#6} {#7}
972 }
973 \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.

```

974 \cs_new:Npn \__draw_point_intersect_line_circle_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
975   {
976     \__draw_point_intersect_line_circle_auxv:fnnnn
977     { \fp_eval:n { (-1 * #4 + #2 * sqrt(#1)) / (2 * #3) } }
978     {#5} {#6} {#7} {#8}
979   }
980 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxiv:nnnnnnnn { ff }
981 \cs_new:Npn \__draw_point_intersect_line_circle_auxv:nnnn #1#2#3#4#5
982   {
983     \draw_point:n
984     { #2 + #1 * (#4 - #2), #3 + #1 * (#5 - #3) }
985   }
986 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxv:nnnn { f }

```

5.6 Interpolation on a line (vector) or arc

Simple maths after expansion.

```

\draw_point_interpolate_line:nnm
\__draw_point_interpolate_line_aux:nnnnn
\__draw_point_interpolate_line_aux:fnnnn
\__draw_point_interpolate_line_aux:nnnnnn
\__draw_point_interpolate_line_aux:fnnnnn
987 \cs_new:Npn \draw_point_interpolate_line:nnn #1#2#3
988   {
989     \__draw_point_process:nnn
990     { \__draw_point_interpolate_line_aux:fnnnn { \fp_eval:n {#1} } }
991     {#2} {#3}
992   }
993 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnn #1#2#3#4#5
994   {
995     \__draw_point_interpolate_line_aux:fnnnnn { \fp_eval:n { 1 - #1 } }
996     {#1} {#2} {#3} {#4} {#5}
997   }
998 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnn { f }
999 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnnn #1#2#3#4#5#6
1000   { \draw_point:n { #2 * #3 + #1 * #5 , #2 * #4 + #1 * #6 } }
1001 \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
\__draw_point_interpolate_distance:nnnnn
\__draw_point_interpolate_distance:nnnnnn
\__draw_point_interpolate_distance:fnnnnn
1002 \cs_new:Npn \draw_point_interpolate_distance:nnn #1#2#3
1003 {
1004   \__draw_point_process:nn
1005   { \__draw_point_interpolate_distance:nnnn {#1} {#3} }
1006   {#2}
1007 }
1008 \cs_new:Npn \__draw_point_interpolate_distance:nnnn #1#2#3#4
1009 {
1010   \__draw_point_process:nn
1011   {
1012     \__draw_point_interpolate_distance:fnnnn
1013     { \fp_eval:n {#1} } {#3} {#4}
1014   }
1015   { \draw_point_unit_vector:n { ( #2 ) - ( #3 , #4 ) } }
1016 }
1017 \cs_new:Npn \__draw_point_interpolate_distance:nnnnn #1#2#3#4#5
1018 { \draw_point:n { #2 + #1 * #4 , #3 + #1 * #5 } }
1019 \cs_generate_variant:Nn \__draw_point_interpolate_distance:nnnnn { f }

```

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

```

\draw_point_interpolate_arcaxes:nnnnnn
\draw_point_interpolate_arcaxes_auxi:nnnnnnnnn
\draw_point_interpolate_arcaxes_auxii:nnnnnnnnn
\draw_point_interpolate_arcaxes_auxiii:nnnnnnnnn
\draw_point_interpolate_arcaxes_auxiiii:nnnnnnnnn
\draw_point_interpolate_arcaxes_auxiv:nnnnnnnnn
\draw_point_interpolate_arcaxes_auxiv:ffnnnnnnn

```

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.

```

1020 \cs_new:Npn \draw_point_interpolate_arcaxes:nnnnnn #1#2#3#4#5#6
1021 {
1022   \__draw_point_process:nnnnn
1023   { \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnn {#1} {#5} {#6} }
1024   {#2} {#3} {#4}
1025 }
1026 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnn #1#2#3#4#5#6#7#8#9
1027 {
1028   \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnn
1029   { \fp_eval:n {#1} } {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1030 }

```

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

- #1 p
- #2 θ_1
- #3 θ_2
- #4 x_c
- #5 y_c
- #6 x_{a1}
- #7 y_{a1}
- #8 x_{a2}

#9 y_{a2}

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

```

1031 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnn #1#2#3#4#5#6#7#8#9
1032 {
1033   \__draw_point_interpolate_arcaxes_auxiii:fnnnnnnn
1034   { \fp_eval:n { #1 * (#3) + ( 1 - #1 ) * (#2) } }
1035   {#4} {#5} {#6} {#7} {#8} {#9}
1036 }
1037 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnn { f }
1038 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnn #1#2#3#4#5#6#7
1039 {
1040   \__draw_point_interpolate_arcaxes_auxiv:ffnnnnnnn
1041   { \fp_eval:n { cosd (#1) } }
1042   { \fp_eval:n { sind (#1) } }
1043   {#2} {#3} {#4} {#5} {#6} {#7}
1044 }
1045 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnn { f }
1046 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
1047 {
1048   \draw_point:n
1049   { #3 + #1 * #5 + #2 * #7 , #4 + #1 * #6 + #2 * #8 }
1050 }
1051 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnn { ff }

```

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

```

\draw_point_interpolate_curve:nnnnn
draw_point_interpolate_curve_auxi:nnnnnnnnn
raw_point_interpolate_curve_auxii:nnnnnnnnn
\draw_point_interpolate_curve_auxiii:nnnnnnn
\draw_point_interpolate_curve_auxiiii:fnnnnnn
\draw_point_interpolate_curve_auxiv:nnnnnnn
\draw_point_interpolate_curve_auxv:nnw
\draw_point_interpolate_curve_auxv:ffw
\draw_point_interpolate_curve_auxvi:n
raw_point_interpolate_curve_auxvii:nnnnnnnnn
draw_point_interpolate_curve_auxviii:nnnnnnn
draw_point_interpolate_curve_auxviii:ffnnnnn

```

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.

```

1052 \cs_new:Npn \draw_point_interpolate_curve:nnnnnn #1#2#3#4#5
1053 {
1054   \__draw_point_process:nnnnn
1055   { \__draw_point_interpolate_curve_auxi:nnnnnnnnn {#1} }
1056   {#2} {#3} {#4} {#5}
1057 }
1058 \cs_new:Npn \__draw_point_interpolate_curve_auxi:nnnnnnnnn #1#2#3#4#5#6#7#8#9
1059 {
1060   \__draw_point_interpolate_curve_auxii:fnnnnnnnnn
1061   { \fp_eval:n {#1} }
1062   {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1063 }

```

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!

```

1064 \cs_new:Npn \__draw_point_interpolate_curve_auxii:nnnnnnnn
1065   #1#2#3#4#5#6#7#8#9
1066   {
1067     \__draw_point_interpolate_curve_auxiii:fnnnnn
1068     { \fp_eval:n { 1 - #1 } }
1069     {#1}
1070     { {#2} {#3} } { {#4} {#5} } { {#6} {#7} } { {#8} {#9} }
1071   }
1072 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxii:nnnnnnnn { f }
1073 % \begin{macrocode}
1074 % We need to do the first cycle, but haven't got enough arguments to keep
1075 % everything in play at once. So here we use a bit of argument re-ordering
1076 % and a single auxiliary to get the job done.
1077 % \begin{macrocode}
1078 \cs_new:Npn \__draw_point_interpolate_curve_auxiii:nnnnnn #1#2#3#4#5#6
1079   {
1080     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #3 #4
1081     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #4 #5
1082     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #5 #6
1083     \prg_do_nothing:
1084     \__draw_point_interpolate_curve_auxvi:n { {#1} {#2} }
1085   }
1086 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxiii:nnnnnn { f }
1087 \cs_new:Npn \__draw_point_interpolate_curve_auxiv:nnnnnn #1#2#3#4#5#6
1088   {
1089     \__draw_point_interpolate_curve_auxv:ffw
1090     { \fp_eval:n { #1 * #3 + #2 * #5 } }

```

```

1091     { \fp_eval:n { #1 * #4 + #2 * #6 } }
1092   }
1093 \cs_new:Npn \__draw_point_interpolate_curve_auxv:nnw
1094   #1#2#3 \prg_do_nothing: #4#5
1095   {
1096     #3
1097     \prg_do_nothing:
1098     #4 { #5 {#1} {#2} }
1099   }
1100 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxv:nnw { ff }
1101 %   \begin{macrocode}
1102 %   Get the arguments back into the right places and to the second and
1103 %   third cycles directly.
1104 %   \begin{macrocode}
1105 \cs_new:Npn \__draw_point_interpolate_curve_auxvi:n #1
1106   { \__draw_point_interpolate_curve_auxvii:nnnnnnn #1 }
1107 \cs_new:Npn \__draw_point_interpolate_curve_auxvii:nnnnnnn #1#2#3#4#5#6#7#8
1108   {
1109     \__draw_point_interpolate_curve_auxviii:ffffnn
1110     { \fp_eval:n { #1 * #5 + #2 * #3 } }
1111     { \fp_eval:n { #1 * #6 + #2 * #4 } }
1112     { \fp_eval:n { #1 * #7 + #2 * #5 } }
1113     { \fp_eval:n { #1 * #8 + #2 * #6 } }
1114     {#1} {#2}
1115   }
1116 \cs_new:Npn \__draw_point_interpolate_curve_auxviii:nnnnn #1#2#3#4#5#6
1117   {
1118     \draw_point:n
1119     { #5 * #3 + #6 * #1 , #5 * #4 + #6 * #2 }
1120   }
1121 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxviii:nnnnn { ffff }

```

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

5.7 Vector support

As well as co-ordinates relative to the drawing

```

\l__draw_xvec_x_dim Base vectors to map to the underlying two-dimensional drawing space.
\l__draw_xvec_y_dim 1122 \dim_new:N \l__draw_xvec_x_dim
\l__draw_yvec_x_dim 1123 \dim_new:N \l__draw_xvec_y_dim
\l__draw_yvec_y_dim 1124 \dim_new:N \l__draw_yvec_x_dim
\l__draw_zvec_x_dim 1125 \dim_new:N \l__draw_yvec_y_dim
\l__draw_zvec_y_dim 1126 \dim_new:N \l__draw_zvec_x_dim
1127 \dim_new:N \l__draw_zvec_y_dim

```

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

```

\draw_xvec:n Calculate the underlying position and store it.
\draw_yvec:n 1128 \cs_new_protected:Npn \draw_xvec:n #1
\draw_zvec:n 1129 { \__draw_vec:nn { x } {#1} }
\__draw_vec:nn 1130 \cs_new_protected:Npn \draw_yvec:n #1
\__draw_vec:nnn 1131 { \__draw_vec:nn { y } {#1} }

```



```

1132 \cs_new_protected:Npn \draw_zvec:n #1
1133 { \__draw_vec:nn { z } {#1} }
1134 \cs_new_protected:Npn \__draw_vec:nn #1#2
1135 {
1136   \__draw_point_process:nn { \__draw_vec:nnn {#1} } {#2}
1137 }
1138 \cs_new_protected:Npn \__draw_vec:nnn #1#2#3
1139 {
1140   \dim_set:cn { l__draw_ #1 vec_x_dim } {#2}
1141   \dim_set:cn { l__draw_ #1 vec_y_dim } {#3}
1142 }

```

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

Initialise the vectors.

```

1143 \draw_xvec:n { 1cm , 0cm }
1144 \draw_yvec:n { 0cm , 1cm }
1145 \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 1146 \cs_new:Npn \draw_point_vec:nn #1#2
\__draw_point_vec:ff 1147 { \__draw_point_vec:ff { \fp_eval:n {#1} } { \fp_eval:n {#2} } }
\draw_point_vec:nnn 1148 \cs_new:Npn \__draw_point_vec:nn #1#2
\__draw_point_vec:nnn 1149 {
\__draw_point_vec:fff 1150   \draw_point:n
1151   {
1152     #1 * \l__draw_xvec_x_dim + #2 * \l__draw_yvec_x_dim ,
1153     #1 * \l__draw_xvec_y_dim + #2 * \l__draw_yvec_y_dim
1154   }
1155 }
1156 \cs_generate_variant:Nn \__draw_point_vec:nn { ff }
1157 \cs_new:Npn \draw_point_vec:nnn #1#2#3
1158 {
1159   \__draw_point_vec:fff
1160   { \fp_eval:n {#1} } { \fp_eval:n {#2} } { \fp_eval:n {#3} }
1161 }
1162 \cs_new:Npn \__draw_point_vec:nnn #1#2#3
1163 {
1164   \draw_point:n
1165   {
1166     #1 * \l__draw_xvec_x_dim
1167     + #2 * \l__draw_yvec_x_dim
1168     + #3 * \l__draw_zvec_x_dim
1169     ,
1170     #1 * \l__draw_xvec_y_dim
1171     + #2 * \l__draw_yvec_y_dim
1172     + #3 * \l__draw_zvec_y_dim
1173   }
1174 }
1175 \cs_generate_variant:Nn \__draw_point_vec:nnn { fff }

```

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

\draw_point_vec_polar:nn Much the same as the core polar approach.

```

\draw_point_vec_polar:nnn 1176 \cs_new:Npn \draw_point_vec_polar:nn #1#2
\__draw_point_vec_polar:nnn
\__draw_point_vec_polar:fnn

```

```

1177 { \draw_point_vec_polar:nnn {#1} {#1} {#2} }
1178 \cs_new:Npn \draw_point_vec_polar:nnn #1#2#3
1179 { \__draw_draw_vec_polar:fnn { \fp_eval:n {#3} } {#1} {#2} }
1180 \cs_new:Npn \__draw_draw_vec_polar:nnn #1#2#3
1181 {
1182   \draw_point:n
1183   {
1184     cosd(#1) * (#2) * \l__draw_xvec_x_dim ,
1185     sind(#1) * (#3) * \l__draw_yvec_y_dim
1186   }
1187 }
1188 \cs_generate_variant:Nn \__draw_draw_vec_polar:nnn { f }

```

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

5.8 Transformations

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

```

1189 \cs_new:Npn \draw_point_transform:n #1
1190 {
1191   \__draw_point_process:nn
1192   { \__draw_point_transform:nn } {#1}
1193 }
1194 \cs_new:Npn \__draw_point_transform:nn #1#2
1195 {
1196   \bool_if:NTF \l__draw_matrix_active_bool
1197   {
1198     \draw_point:n
1199     {
1200       (
1201         \l__draw_matrix_a_fp * #1
1202         + \l__draw_matrix_c_fp * #2
1203         + \l__draw_xshift_dim
1204       )
1205       ,
1206       (
1207         \l__draw_matrix_b_fp * #1
1208         + \l__draw_matrix_d_fp * #2
1209         + \l__draw_yshift_dim
1210       )
1211     }
1212   }
1213   {
1214     \draw_point:n
1215     {
1216       (#1, #2)
1217       + ( \l__draw_xshift_dim , \l__draw_yshift_dim )
1218     }
1219   }
1220 }

```

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

```

\__draw_point_transform_noshift:n A version with no shift: used for internal purposes.
\__draw_point_transform_noshift:nn
1221 \cs_new:Npn \__draw_point_transform_noshift:n #1
1222 {
1223   \__draw_point_process:nn
1224   { \__draw_point_transform_noshift:nn } {#1}
1225 }
1226 \cs_new:Npn \__draw_point_transform_noshift:nn #1#2
1227 {
1228   \bool_if:NTF \l__draw_matrix_active_bool
1229   {
1230     \draw_point:n
1231     {
1232       (
1233         \l__draw_matrix_a_fp * #1
1234         + \l__draw_matrix_c_fp * #2
1235       )
1236       ,
1237       (
1238         \l__draw_matrix_b_fp * #1
1239         + \l__draw_matrix_d_fp * #2
1240       )
1241     }
1242   }
1243   { \draw_point:n { (#1, #2) } }
1244 }

```

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

```
1245 </package>
```

6 I3draw-scopes implementation

```
1246 <*package>
```

```
1247 <@@=draw>
```

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

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

6.1 Drawing environment

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

```

\g__draw_ymax_dim 1248 \dim_new:N \g__draw_xmax_dim
\g__draw_ymin_dim 1249 \dim_new:N \g__draw_xmin_dim
1250 \dim_new:N \g__draw_ymax_dim
1251 \dim_new:N \g__draw_ymin_dim

```

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

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

```
1252 \bool_new:N \l_draw_bb_update_bool
```

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

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

```
1253 \box_new:N \l__draw_main_box
1254 \box_new:N \l__draw_layer_main_box
```

(End of definition for `\l__draw_layer_main_box`.)

`\g__draw_id_int` The drawing number.

```
1255 \int_new:N \g__draw_id_int
```

(End of definition for `\g__draw_id_int`.)

`__draw_reset_bb:` A simple auxiliary.

```
1256 \cs_new_protected:Npn \__draw_reset_bb:
1257 {
1258   \dim_gset:Nn \g__draw_xmax_dim { -\c_max_dim }
1259   \dim_gset:Nn \g__draw_xmin_dim { \c_max_dim }
1260   \dim_gset:Nn \g__draw_ymax_dim { -\c_max_dim }
1261   \dim_gset:Nn \g__draw_ymin_dim { \c_max_dim }
1262 }
```

(End of definition for `__draw_reset_bb:`.)

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

```
1263 \cs_new_protected:Npn \draw_begin:
1264 {
1265   \group_begin:
1266     \int_gincr:N \g__draw_id_int
1267     \hbox_set:Nw \l__draw_main_box
1268     \__draw_backend_begin:
1269     \__draw_reset_bb:
1270     \__draw_path_reset_limits:
1271     \bool_set_true:N \l_draw_bb_update_bool
1272     \draw_transform_matrix_reset:
1273     \draw_transform_shift_reset:
1274     \__draw_softpath_clear:
1275     \draw_linewidth:n { \l_draw_default_linewidth_dim }
1276     \color_select:n { . }
1277     \draw_nonzero_rule:
1278     \draw_cap_but:
1279     \draw_join_miter:
1280     \draw_miterlimit:n { 10 }
1281     \draw_dash_pattern:nn { } { 0cm }
1282     \hbox_set:Nw \l__draw_layer_main_box
1283 }
1284 \cs_new_protected:Npn \draw_end:
1285 {
1286     \__draw_baseline_finalise:w
```

```

1287         \exp_args:NNNV \hbox_set_end:
1288         \clist_set:Nn \l_draw_layers_clist \l_draw_layers_clist
1289         \__draw_layers_insert:
1290         \__draw_backend_end:
1291     \hbox_set_end:
1292     \dim_compare:nNnT \g__draw_xmin_dim = \c_max_dim
1293     {
1294         \dim_gzero:N \g__draw_xmax_dim
1295         \dim_gzero:N \g__draw_xmin_dim
1296         \dim_gzero:N \g__draw_ymax_dim
1297         \dim_gzero:N \g__draw_ymin_dim
1298     }
1299     \__draw_finalise:
1300     \box_set_wd:Nn \l__draw_main_box
1301     { \g__draw_xmax_dim - \g__draw_xmin_dim }
1302     \mode_leave_vertical:
1303     \box_use_drop:N \l__draw_main_box
1304 \group_end:
1305 }

```

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

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

```

1306 \cs_new_protected:Npn \__draw_finalise:
1307 {
1308     \hbox_set:Nn \l__draw_main_box
1309     {
1310         \skip_horizontal:n { -\g__draw_xmin_dim }
1311         \box_move_down:nn
1312         { \g__draw_ymin_dim }
1313         { \box_use_drop:N \l__draw_main_box }
1314     }
1315     \box_set_dp:Nn \l__draw_main_box { Opt }
1316     \box_set_ht:Nn \l__draw_main_box
1317     { \g__draw_ymax_dim - \g__draw_ymin_dim }
1318 }
1319 \cs_new_protected:Npn \__draw_finalise_baseline:n #1
1320 {
1321     \hbox_set:Nn \l__draw_main_box
1322     {
1323         \skip_horizontal:n { -\g__draw_xmin_dim }
1324         \box_move_down:nn
1325         { #1 }
1326         { \box_use_drop:N \l__draw_main_box }
1327     }
1328     \box_set_dp:Nn \l__draw_main_box
1329     {
1330         \dim_max:nn
1331         { #1 - \g__draw_ymin_dim }
1332         { Opt }

```

```

1333     }
1334     \box_set_ht:Nn \l__draw_main_box
1335     { \g__draw_ymax_dim + #1 }
1336 }

```

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

6.2 Baseline position

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

```

\l__draw_baseline_dim
1337 \bool_new:N \l__draw_baseline_bool
1338 \dim_new:N \l__draw_baseline_dim

```

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

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

```

1339 \cs_new_protected:Npn \draw_baseline:n #1
1340 {
1341     \bool_set_true:N \l__draw_baseline_bool
1342     \dim_set:Nn \l__draw_baseline_dim { \fp_to_dim:n {#1} }
1343 }

```

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

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

```

1344 \cs_new_protected:Npn \__draw_baseline_finalise:w #1 \__draw_finalise:
1345 {
1346     \bool_if:NTF \l__draw_baseline_bool
1347     {
1348         \use:e
1349         {
1350             \exp_not:n {#1}
1351             \__draw_finalise_baseline:n { \dim_use:N \l__draw_baseline_dim }
1352         }
1353     }
1354     { #1 \__draw_finalise: }
1355 }

```

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

6.3 Scopes

`\l__draw_linewidth_dim` Storage for local variables.

```

\l__draw_fill_color_tl
\l__draw_stroke_color_tl
1356 \dim_new:N \l__draw_linewidth_dim
1357 \tl_new:N \l__draw_fill_color_tl
1358 \tl_new:N \l__draw_stroke_color_tl

```

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

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

```
\draw_scope_begin: 1359 \cs_new_protected:Npn \draw_scope_begin:
1360 {
1361   \__draw_backend_scope_begin:
1362   \group_begin:
1363     \dim_set_eq:NN \l__draw_linewidth_dim \g__draw_linewidth_dim
1364     \draw_path_scope_begin:
1365   }
1366 \cs_new_protected:Npn \draw_scope_end:
1367 {
1368   \draw_path_scope_end:
1369   \dim_gset_eq:NN \g__draw_linewidth_dim \l__draw_linewidth_dim
1370   \group_end:
1371   \__draw_backend_scope_end:
1372 }
```

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

`\l__draw_xmax_dim` Storage for the bounding box.

```
\l__draw_xmin_dim 1373 \dim_new:N \l__draw_xmax_dim
\l__draw_ymax_dim 1374 \dim_new:N \l__draw_xmin_dim
\l__draw_ymin_dim 1375 \dim_new:N \l__draw_ymax_dim
1376 \dim_new:N \l__draw_ymin_dim
```

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

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

```
\__draw_scope_bb_end: 1377 \cs_new_protected:Npn \__draw_scope_bb_begin:
1378 {
1379   \group_begin:
1380     \dim_set_eq:NN \l__draw_xmax_dim \g__draw_xmax_dim
1381     \dim_set_eq:NN \l__draw_xmin_dim \g__draw_xmin_dim
1382     \dim_set_eq:NN \l__draw_ymax_dim \g__draw_ymax_dim
1383     \dim_set_eq:NN \l__draw_ymin_dim \g__draw_ymin_dim
1384     \__draw_reset_bb:
1385   }
1386 \cs_new_protected:Npn \__draw_scope_bb_end:
1387 {
1388   \dim_gset_eq:NN \g__draw_xmax_dim \l__draw_xmax_dim
1389   \dim_gset_eq:NN \g__draw_xmin_dim \l__draw_xmin_dim
1390   \dim_gset_eq:NN \g__draw_ymax_dim \l__draw_ymax_dim
1391   \dim_gset_eq:NN \g__draw_ymin_dim \l__draw_ymin_dim
1392   \group_end:
1393 }
```

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

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

```
\draw_suspend_end: 1394 \cs_new_protected:Npn \draw_suspend_begin:
1395 {
1396   \__draw_scope_bb_begin:
1397   \draw_path_scope_begin:
1398   \draw_transform_matrix_reset:
1399   \draw_transform_shift_reset:
```

```

1400     \__draw_layers_save:
1401   }
1402   \cs_new_protected:Npn \draw_suspend_end:
1403     {
1404       \__draw_layers_restore:
1405       \draw_path_scope_end:
1406       \__draw_scope_bb_end:
1407     }

```

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

```

1408 </package>

```

7 l3draw-softpath implementation

```

1409 <*package>

```

```

1410 <@@=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.

```

1411 \tl_new:N \g__draw_softpath_main_tl

```

(End of definition for `\g__draw_softpath_main_tl`.)

`\l__draw_softpath_tmp_tl` Scratch space.

```

1412 \tl_new:N \l__draw_softpath_tmp_tl

```

(End of definition for `\l__draw_softpath_tmp_tl`.)

`\g_draw_softpath_corners_bool` Allow for optimised path use.

```

1413 \bool_new:N \g_draw_softpath_corners_bool

```

(End of definition for `\g_draw_softpath_corners_bool`.)

`__draw_softpath_add:n`

`__draw_softpath_add:o`

`__draw_softpath_add:e`

```

1414 \cs_new_protected:Npn \__draw_softpath_add:n

```

```

1415   { \tl_build_gput_right:Nn \g__draw_softpath_main_tl }

```

```

1416 \cs_generate_variant:Nn \__draw_softpath_add:n { o, e }

```

(End of definition for `__draw_softpath_add:n`.)

`__draw_softpath_use:` Using and clearing is trivial.

`__draw_softpath_clear:`

```

1417 \cs_new_protected:Npn \__draw_softpath_use:
1418 {
1419   \tl_build_gend:N \g__draw_softpath_main_tl
1420   \tl_set_eq:NN \l__draw_softpath_tmp_tl \g__draw_softpath_main_tl
1421   \l__draw_softpath_tmp_tl
1422   \tl_build_gbegin:N \g__draw_softpath_main_tl
1423   \exp_args:NNV \tl_build_gput_right:Nn
1424     \g__draw_softpath_main_tl \l__draw_softpath_tmp_tl
1425 }
1426 \cs_new_protected:Npn \__draw_softpath_clear:
1427 {
1428   \tl_build_gbegin:N \g__draw_softpath_main_tl
1429   \bool_gset_false:N \g__draw_softpath_corners_bool
1430 }

```

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

`__draw_softpath_save:` Abstracted ideas to keep variables inside this submodule.

`__draw_softpath_restore:`

```

1431 \cs_new_protected:Npn \__draw_softpath_save:
1432 {
1433   \tl_build_gend:N \g__draw_softpath_main_tl
1434   \tl_set_eq:NN
1435     \l__draw_softpath_main_tl
1436     \g__draw_softpath_main_tl
1437   \bool_set_eq:NN
1438     \l__draw_softpath_corners_bool
1439     \g__draw_softpath_corners_bool
1440   \__draw_softpath_clear:
1441 }
1442 \cs_new_protected:Npn \__draw_softpath_restore:
1443 {
1444   \__draw_softpath_clear:
1445   \__draw_softpath_add:o \l__draw_softpath_main_tl
1446   \bool_gset_eq:NN
1447     \g__draw_softpath_corners_bool
1448     \l__draw_softpath_corners_bool
1449 }

```

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

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

`\g__draw_softpath_lasty_dim`

```

1450 \dim_new:N \g__draw_softpath_lastx_dim
1451 \dim_new:N \g__draw_softpath_lasty_dim

```

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

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

```

1452 \bool_new:N \g__draw_softpath_move_bool
1453 \bool_gset_true:N \g__draw_softpath_move_bool

```

(End of definition for \g__draw_softpath_move_bool.)

```

    \_draw_softpath_curveto:nnnnn
\__draw_softpath_lineto:nn
\__draw_softpath_moveto:nn
    \_draw_softpath_rectangle:nnnn
    \_draw_softpath_roundpoint:nn
    \_draw_softpath_roundpoint:VV

```

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

```

1454 \cs_new_protected:Npn \__draw_softpath_closepath:
1455 {
1456   \__draw_softpath_add:e
1457   {
1458     \__draw_softpath_close_op:nn
1459     { \dim_use:N \g__draw_softpath_lastx_dim }
1460     { \dim_use:N \g__draw_softpath_lasty_dim }
1461   }
1462 }
1463 \cs_new_protected:Npn \__draw_softpath_curveto:nnnnnn #1#2#3#4#5#6
1464 {
1465   \__draw_softpath_add:n
1466   {
1467     \__draw_softpath_curveto_opi:nn {#1} {#2}
1468     \__draw_softpath_curveto_opii:nn {#3} {#4}
1469     \__draw_softpath_curveto_opiii:nn {#5} {#6}
1470   }
1471 }
1472 \cs_new_protected:Npn \__draw_softpath_lineto:nn #1#2
1473 {
1474   \__draw_softpath_add:n
1475   { \__draw_softpath_lineto_op:nn {#1} {#2} }
1476 }
1477 \cs_new_protected:Npn \__draw_softpath_moveto:nn #1#2
1478 {
1479   \__draw_softpath_add:n
1480   { \__draw_softpath_moveto_op:nn {#1} {#2} }
1481   \bool_if:NT \g__draw_softpath_move_bool
1482   {
1483     \dim_gset:Nn \g__draw_softpath_lastx_dim {#1}
1484     \dim_gset:Nn \g__draw_softpath_lasty_dim {#2}
1485   }
1486 }
1487 \cs_new_protected:Npn \__draw_softpath_rectangle:nnnn #1#2#3#4
1488 {
1489   \__draw_softpath_add:n
1490   {
1491     \__draw_softpath_rectangle_opi:nn {#1} {#2}
1492     \__draw_softpath_rectangle_opii:nn {#3} {#4}
1493   }
1494 }
1495 \cs_new_protected:Npn \__draw_softpath_roundpoint:nn #1#2
1496 {
1497   \__draw_softpath_add:n
1498   { \__draw_softpath_roundpoint_op:nn {#1} {#2} }
1499   \bool_gset_true:N \g__draw_softpath_corners_bool
1500 }
1501 \cs_generate_variant:Nn \__draw_softpath_roundpoint:nn { VV }

```

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

_draw_softpath_close_op:nn The markers for operations: all the top-level ones take two arguments. The support

```

    \_draw_softpath_curveto_opi:nn
    \_draw_softpath_curveto_opii:nn
    \_draw_softpath_curveto_opiii:nn
    \_draw_softpath_lineto_op:nn
    \_draw_softpath_moveto_op:nn
    \_draw_softpath_roundpoint_op:nn
    \_draw_softpath_rectangle_opi:nn
    \_draw_softpath_rectangle_opii:nn
    \_draw_softpath_curveto_opi:nnNnnNnn
    \_draw_softpath_rectangle_opi:nnNnn

```

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

```

1502 \cs_new_protected:Npn \__draw_softpath_close_op:nn #1#2
1503   { \__draw_backend_closepath: }
1504 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nn #1#2
1505   { \__draw_softpath_curveto_opi:nnNnnNnn {#1} {#2} }
1506 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nnNnnNnn #1#2#3#4#5#6#7#8
1507   { \__draw_backend_curveto:nnnnnn {#1} {#2} {#4} {#5} {#7} {#8} }
1508 \cs_new_protected:Npn \__draw_softpath_curveto_opii:nn #1#2
1509   { \__draw_softpath_curveto_opii:nn }
1510 \cs_new_protected:Npn \__draw_softpath_curveto_opiii:nn #1#2
1511   { \__draw_softpath_curveto_opiii:nn }
1512 \cs_new_protected:Npn \__draw_softpath_lineto_op:nn #1#2
1513   { \__draw_backend_lineto:nn {#1} {#2} }
1514 \cs_new_protected:Npn \__draw_softpath_moveto_op:nn #1#2
1515   { \__draw_backend_moveto:nn {#1} {#2} }
1516 \cs_new_protected:Npn \__draw_softpath_roundpoint_op:nn #1#2 { }
1517 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nn #1#2
1518   { \__draw_softpath_rectangle_opi:nnNnn {#1} {#2} }
1519 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nnNnn #1#2#3#4#5
1520   { \__draw_backend_rectangle:nnnn {#1} {#2} {#4} {#5} }
1521 \cs_new_protected:Npn \__draw_softpath_rectangle_opii:nn #1#2 { }

```

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

7.2 Rounding soft path corners

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

`\l__draw_softpath_main_tl` For constructing the updated path.

```
1522 \tl_new:N \l__draw_softpath_main_tl
```

(End of definition for \l__draw_softpath_main_tl.)

`\l__draw_softpath_part_tl` Data structures.

```
1523 \tl_new:N \l__draw_softpath_part_tl
1524 \tl_new:N \l__draw_softpath_curve_end_tl
```

(End of definition for \l__draw_softpath_part_tl.)

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

```

\l__draw_softpath_lasty_fp
  \l__draw_softpath_corneri_dim
  \l__draw_softpath_cornerii_dim
\l__draw_softpath_first_tl
  \l__draw_softpath_move_tl
1525 \fp_new:N \l__draw_softpath_lastx_fp
1526 \fp_new:N \l__draw_softpath_lasty_fp
1527 \dim_new:N \l__draw_softpath_corneri_dim
1528 \dim_new:N \l__draw_softpath_cornerii_dim
1529 \tl_new:N \l__draw_softpath_first_tl
1530 \tl_new:N \l__draw_softpath_move_tl

```

(End of definition for \l__draw_softpath_lastx_fp and others.)

`\c__draw_softpath_arc_fp` The magic constant.

```
1531 \fp_const:Nn \c__draw_softpath_arc_fp { 4/3 * (sqrt(2) - 1) }
```

(End of definition for `\c__draw_softpath_arc_fp`.)

```

\__draw_softpath_round_corners: Rounding corners on a path means going through the entire path and adjusting it. As
\__draw_softpath_round_loop:Nnn such, we avoid this entirely if we know there are no corners to deal with. Assuming there
\__draw_softpath_round_action:nn is work to do, we recover the existing path and start a loop.
\__draw_softpath_round_action:Nnn 1532 \cs_new_protected:Npn \__draw_softpath_round_corners:
\__draw_softpath_round_action_curveto:NnnNnn 1533 {
\__draw_softpath_round_action_close: 1534   \bool_if:NT \g__draw_softpath_corners_bool
\__draw_softpath_round_lookahead:NnnNnn 1535   {
\__draw_softpath_round_roundpoint:NnnNnnNnn 1536     \group_begin:
\__draw_softpath_round_calc:NnnNnn 1537       \tl_clear:N \l__draw_softpath_main_tl
\__draw_softpath_round_calc:nnnnnn 1538       \tl_clear:N \l__draw_softpath_part_tl
\__draw_softpath_round_calc:fVnnnn 1539       \fp_zero:N \l__draw_softpath_lastx_fp
\__draw_softpath_round_calc:nnnnw 1540       \fp_zero:N \l__draw_softpath_lasty_fp
\__draw_softpath_round_close:nn 1541       \tl_clear:N \l__draw_softpath_first_tl
\__draw_softpath_round_close:w 1542       \tl_clear:N \l__draw_softpath_move_tl
\__draw_softpath_round_end: 1543       \tl_build_gend:N \g__draw_softpath_main_tl
1544       \exp_after:wN \__draw_softpath_round_loop:Nnn
1545         \g__draw_softpath_main_tl
1546         \q__draw_recursion_tail ? ?
1547         \q__draw_recursion_stop
1548       \group_end:
1549     }
1550   \bool_gset_false:N \g__draw_softpath_corners_bool
1551 }

```

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.

```

1552 \cs_new_protected:Npn \__draw_softpath_round_loop:Nnn #1#2#3
1553 {
1554   \__draw_if_recursion_tail_stop_do:Nn #1 { \__draw_softpath_round_end: }
1555   \token_if_eq_meaning:NNTF #1 \__draw_softpath_roundpoint_op:nn
1556   { \__draw_softpath_round_action:nn {#2} {#3} }
1557   {
1558     \tl_if_empty:NT \l__draw_softpath_first_tl
1559     { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1560     \fp_set:Nn \l__draw_softpath_lastx_fp {#2}
1561     \fp_set:Nn \l__draw_softpath_lasty_fp {#3}
1562     \token_if_eq_meaning:NNTF #1 \__draw_softpath_moveto_op:nn
1563     {
1564       \tl_put_right:No \l__draw_softpath_main_tl
1565       \l__draw_softpath_move_tl
1566       \tl_put_right:No \l__draw_softpath_main_tl
1567       \l__draw_softpath_part_tl
1568       \tl_set:Nn \l__draw_softpath_move_tl { #1 {#2} {#3} }
1569       \tl_clear:N \l__draw_softpath_first_tl
1570       \tl_clear:N \l__draw_softpath_part_tl
1571     }
1572     { \tl_put_right:Nn \l__draw_softpath_part_tl { #1 {#2} {#3} } }
1573   }
1574   \__draw_softpath_round_loop:Nnn

```

```

1575 }
1576 \cs_new_protected:Npn \__draw_softpath_round_action:nn #1#2
1577 {
1578   \dim_set:Nn \l__draw_softpath_corneri_dim {#1}
1579   \dim_set:Nn \l__draw_softpath_cornerii_dim {#2}
1580   \bool_lazy_and:nnTF
1581     { \dim_compare_p:nNn \l__draw_softpath_corneri_dim = { Opt } }
1582     { \dim_compare_p:nNn \l__draw_softpath_cornerii_dim = { Opt } }
1583     { \__draw_softpath_round_loop:Nnn }
1584     { \__draw_softpath_round_action:Nnn }
1585 }

```

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.

```

1586 \cs_new_protected:Npn \__draw_softpath_round_action:Nnn #1#2#3
1587 {
1588   \tl_if_empty:NT \l__draw_softpath_first_tl
1589     { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1590   \token_if_eq_meaning:NNTF #1 \__draw_softpath_curveto_opi:nn
1591     { \__draw_softpath_round_action_curveto:NnnNnn }
1592     {
1593       \token_if_eq_meaning:NNTF #1 \__draw_softpath_close_op:nn
1594         { \__draw_softpath_round_action_close: }
1595         {
1596           \token_if_eq_meaning:NNTF #1 \__draw_softpath_lineto_op:nn
1597             { \__draw_softpath_round_lookahead:NnnNnn }
1598             { \__draw_softpath_round_loop:Nnn }
1599         }
1600     }
1601     #1 {#2} {#3}
1602 }

```

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.

```

1603 \cs_new_protected:Npn \__draw_softpath_round_action_curveto:NnnNnn
1604   #1#2#3#4#5#6
1605 {
1606   \tl_put_right:Nn \l__draw_softpath_part_tl
1607     { #1 {#2} {#3} #4 {#5} {#6} }
1608   \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1609   \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1610   \__draw_softpath_round_lookahead:NnnNnn
1611 }
1612 \cs_new_protected:Npn \__draw_softpath_round_action_close:
1613 {
1614   \bool_lazy_and:nnTF
1615     { ! \tl_if_empty_p:N \l__draw_softpath_first_tl }
1616     { ! \tl_if_empty_p:N \l__draw_softpath_move_tl }
1617     {
1618       \exp_after:wN \__draw_softpath_round_close:nn
1619         \l__draw_softpath_first_tl
1620     }
1621     { \__draw_softpath_round_loop:Nnn }
1622 }

```

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.

```

1623 \cs_new_protected:Npn \__draw_softpath_round_lookahead:NnnNnn #1#2#3#4#5#6
1624 {
1625   \bool_lazy_any:nTF
1626   {
1627     { \token_if_eq_meaning_p:NN #4 \__draw_softpath_lineto_op:nn }
1628     { \token_if_eq_meaning_p:NN #4 \__draw_softpath_curveto_opi:nn }
1629     { \token_if_eq_meaning_p:NN #4 \__draw_softpath_close_op:nn }
1630   }
1631   {
1632     \__draw_softpath_round_calc:NnnNnn
1633     \__draw_softpath_round_loop:Nnn
1634     {#5} {#6}
1635   }
1636   {
1637     \token_if_eq_meaning:NNTF #4 \__draw_softpath_roundpoint_op:nn
1638     { \__draw_softpath_round_roundpoint:NnnNnnNnn }
1639     { \__draw_softpath_round_loop:Nnn }
1640   }
1641   #1 {#2} {#3}
1642   #4 {#5} {#6}
1643 }
1644 \cs_new_protected:Npn \__draw_softpath_round_roundpoint:NnnNnnNnn
1645 #1#2#3#4#5#6#7#8#9
1646 {
1647   \__draw_softpath_round_calc:NnnNnn
1648   \__draw_softpath_round_loop:Nnn
1649   {#8} {#9}
1650   #1 {#2} {#3}
1651   #4 {#5} {#6} #7 {#8} {#9}
1652 }

```

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

```

1653 \cs_new_protected:Npn \__draw_softpath_round_calc:NnnNnn #1#2#3#4#5#6
1654 {
1655   \tl_set:Ne \l__draw_softpath_curve_end_tl
1656   {
1657     \draw_point_interpolate_distance:nnn
1658     \l__draw_softpath_cornerii_dim
1659     { #5 , #6 } { #2 , #3 }
1660   }
1661   \tl_put_right:Ne \l__draw_softpath_part_tl
1662   {
1663     \exp_not:N #4

```

```

1664     \__draw_softpath_round_calc:fVnnnn
1665     {
1666         \draw_point_interpolate_distance:nnn
1667         \l__draw_softpath_corneri_dim
1668         { #5 , #6 }
1669         {
1670             \l__draw_softpath_lastx_fp ,
1671             \l__draw_softpath_lasty_fp
1672         }
1673     }
1674     \l__draw_softpath_curve_end_t1
1675     {#5} {#6} {#2} {#3}
1676 }
1677 \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1678 \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1679 #1
1680 }

```

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).

```

1681 \cs_new:Npn \__draw_softpath_round_calc:nnnnnn #1#2#3#4#5#6
1682 {
1683     \__draw_softpath_round_calc:nnnw {#3} {#4} {#5} {#6}
1684     #1 \s__draw_mark #2 \s__draw_stop
1685 }
1686 \cs_generate_variant:Nn \__draw_softpath_round_calc:nnnnnn { fV }

```

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

```

1687 \cs_new:Npn \__draw_softpath_round_calc:nnnw
1688 #1#2#3#4 #5 , #6 \s__draw_mark #7 , #8 \s__draw_stop
1689 {
1690     {#5} {#6}
1691     \exp_not:N \__draw_softpath_curveto_opi:nn
1692     {
1693         \fp_to_dim:n
1694         { #5 + \c__draw_softpath_arc_fp * ( #1 - #5 ) }
1695     }
1696     {
1697         \fp_to_dim:n
1698         { #6 + \c__draw_softpath_arc_fp * ( #2 - #6 ) }
1699     }
1700     \exp_not:N \__draw_softpath_curveto_opii:nn
1701     {
1702         \fp_to_dim:n
1703         { #7 + \c__draw_softpath_arc_fp * ( #1 - #7 ) }
1704     }
1705     {
1706         \fp_to_dim:n
1707         { #8 + \c__draw_softpath_arc_fp * ( #2 - #8 ) }
1708     }
1709     \exp_not:N \__draw_softpath_curveto_opiii:nn
1710     {#7} {#8}
1711 }

```

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.

```

1712 \cs_new_protected:Npn \__draw_softpath_round_close:nn #1#2
1713 {
1714   \use:e
1715   {
1716     \__draw_softpath_round_calc:NnnNnn
1717     {
1718       \tl_set:Nc \exp_not:N \l__draw_softpath_move_tl
1719       {
1720         \__draw_softpath_moveto_op:nn
1721         \exp_not:N \exp_after:wN
1722         \exp_not:N \__draw_softpath_round_close:w
1723         \exp_not:N \l__draw_softpath_curve_end_tl
1724         \s__draw_stop
1725       }
1726       \use:e
1727       {
1728         \exp_not:N \exp_not:N \exp_not:N \use_i:nnnn
1729         {
1730           \__draw_softpath_round_loop:Nnn
1731           \__draw_softpath_close_op:nn
1732           \exp_not:N \exp_after:wN
1733           \exp_not:N \__draw_softpath_round_close:w
1734           \exp_not:N \l__draw_softpath_curve_end_tl
1735           \s__draw_stop
1736         }
1737       }
1738     }
1739     {#1} {#2}
1740     \__draw_softpath_lineto_op:nn
1741     \exp_after:wN \use_none:n \l__draw_softpath_move_tl
1742   }
1743 }
1744 \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.

```

1745 \cs_new_protected:Npn \__draw_softpath_round_end:
1746 {
1747   \tl_put_right:No \l__draw_softpath_main_tl
1748   \l__draw_softpath_move_tl
1749   \tl_put_right:No \l__draw_softpath_main_tl
1750   \l__draw_softpath_part_tl
1751   \tl_build_gbegin:N \g__draw_softpath_main_tl
1752   \__draw_softpath_add:o \l__draw_softpath_main_tl
1753 }

```

(End of definition for __draw_softpath_round_corners: and others.)

```

1754 \</package>

```


8 I3draw-state implementation

1755 `<*package>`

1756 `<@@=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.

1757 `\dim_new:N \g__draw_linewidth_dim`

(End of definition for `\g__draw_linewidth_dim`.)

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

1758 `\dim_new:N \l_draw_default_linewidth_dim`

1759 `\dim_set:Nn \l_draw_default_linewidth_dim { 0.4pt }`

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

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

1760 `\cs_new_protected:Npn \draw_linewidth:n #1`

1761 `{`

1762 `\dim_gset:Nn \g__draw_linewidth_dim { \fp_to_dim:n {#1} }`

1763 `__draw_backend_linewidth:n \g__draw_linewidth_dim`

1764 `}`

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

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

`\l__draw_tmp_seq` 1765 `\cs_new_protected:Npn \draw_dash_pattern:nn #1#2`

1766 `{`

1767 `\group_begin:`

1768 `\seq_set_from_clist:Nn \l__draw_tmp_seq {#1}`

1769 `\seq_set_map:NNn \l__draw_tmp_seq \l__draw_tmp_seq`

1770 `{ \fp_to_dim:n {##1} }`

1771 `\use:e`

1772 `{`

1773 `__draw_backend_dash_pattern:nn`

1774 `{ \seq_use:Nn \l__draw_tmp_seq { , } }`

1775 `{ \fp_to_dim:n {#2} }`

1776 `}`

1777 `\group_end:`

1778 `}`

1779 `\seq_new:N \l__draw_tmp_seq`

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

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

1780 `\cs_new_protected:Npn \draw_miterlimit:n #1`

1781 `{ \exp_args:Ne __draw_backend_miterlimit:n { \fp_eval:n {#1} } }`

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

```
\draw_cap_but: All straight wrappers.
\draw_cap_rectangle: 1782 \cs_new_protected:Npn \draw_cap_but: { \__draw_backend_cap_but: }
\draw_cap_round: 1783 \cs_new_protected:Npn \draw_cap_rectangle: { \__draw_backend_cap_rectangle: }
\draw_evenodd_rule: 1784 \cs_new_protected:Npn \draw_cap_round: { \__draw_backend_cap_round: }
\draw_nonzero_rule: 1785 \cs_new_protected:Npn \draw_evenodd_rule: { \__draw_backend_evenodd_rule: }
\draw_join_bevel: 1786 \cs_new_protected:Npn \draw_nonzero_rule: { \__draw_backend_nonzero_rule: }
\draw_join_miter: 1787 \cs_new_protected:Npn \draw_join_bevel: { \__draw_backend_join_bevel: }
\draw_join_round: 1788 \cs_new_protected:Npn \draw_join_miter: { \__draw_backend_join_miter: }
1789 \cs_new_protected:Npn \draw_join_round: { \__draw_backend_join_round: }
```

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

```
1790 </package>
```

9 I3draw-transforms implementation

```
1791 <*package>
```

```
1792 <@@=draw>
```

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

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

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

```
1793 \bool_new:N \l__draw_matrix_active_bool
```

(End of definition for `\l__draw_matrix_active_bool`.)

```
\l__draw_matrix_a_fp The active matrix and shifts.
```

```
\l__draw_matrix_b_fp 1794 \fp_new:N \l__draw_matrix_a_fp
```

```
\l__draw_matrix_c_fp 1795 \fp_new:N \l__draw_matrix_b_fp
```

```
\l__draw_xshift_dim 1796 \fp_new:N \l__draw_matrix_c_fp
```

```
\l__draw_yshift_dim 1797 \fp_new:N \l__draw_matrix_d_fp
```

```
1798 \dim_new:N \l__draw_xshift_dim
```

```
1799 \dim_new:N \l__draw_yshift_dim
```

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

`\draw_transform_matrix_reset:` Fast resetting.

```
\draw_transform_shift_reset: 1800 \cs_new_protected:Npn \draw_transform_matrix_reset:
1801 {
1802   \fp_set:Nn \l__draw_matrix_a_fp { 1 }
1803   \fp_zero:N \l__draw_matrix_b_fp
1804   \fp_zero:N \l__draw_matrix_c_fp
1805   \fp_set:Nn \l__draw_matrix_d_fp { 1 }
1806 }
1807 \cs_new_protected:Npn \draw_transform_shift_reset:
1808 {
1809   \dim_zero:N \l__draw_xshift_dim
1810   \dim_zero:N \l__draw_yshift_dim
1811 }
1812 \draw_transform_matrix_reset:
1813 \draw_transform_shift_reset:
```

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

`\draw_transform_matrix_absolute:nmmn` Setting the transform matrix is straight-forward, with just a bit of expansion to sort out.

`\draw_transform_shift_absolute:n` With the mechanism active, the identity matrix is set.

```
\_draw_transform_shift_absolute:nn 1814 \cs_new_protected:Npn \draw_transform_matrix_absolute:nmmn #1#2#3#4
1815 {
1816   \fp_set:Nn \l__draw_matrix_a_fp {#1}
1817   \fp_set:Nn \l__draw_matrix_b_fp {#2}
1818   \fp_set:Nn \l__draw_matrix_c_fp {#3}
1819   \fp_set:Nn \l__draw_matrix_d_fp {#4}
1820   \bool_lazy_all:nTF
1821     {
1822       { \fp_compare_p:nNn \l__draw_matrix_a_fp = \c_one_fp }
1823       { \fp_compare_p:nNn \l__draw_matrix_b_fp = \c_zero_fp }
1824       { \fp_compare_p:nNn \l__draw_matrix_c_fp = \c_zero_fp }
1825       { \fp_compare_p:nNn \l__draw_matrix_d_fp = \c_one_fp }
1826     }
1827     { \bool_set_false:N \l__draw_matrix_active_bool }
1828     { \bool_set_true:N \l__draw_matrix_active_bool }
1829 }
1830 \cs_new_protected:Npn \draw_transform_shift_absolute:n #1
1831 {
1832   \__draw_point_process:nn
1833   { \__draw_transform_shift_absolute:nn } {#1}
1834 }
1835 \cs_new_protected:Npn \__draw_transform_shift_absolute:nn #1#2
1836 {
1837   \dim_set:Nn \l__draw_xshift_dim {#1}
1838   \dim_set:Nn \l__draw_yshift_dim {#2}
1839 }
```

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

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

`__draw_transform:nmmn`

`\draw_transform_shift:n`

```
1840 \cs_new_protected:Npn \draw_transform_matrix:nmmn #1#2#3#4
```

`__draw_transform_shift:nn`

```

1841 {
1842   \use:e
1843   {
1844     \__draw_transform:nnnn
1845     { \fp_eval:n {#1} }
1846     { \fp_eval:n {#2} }
1847     { \fp_eval:n {#3} }
1848     { \fp_eval:n {#4} }
1849   }
1850 }
1851 \cs_new_protected:Npn \__draw_transform:nnnn #1#2#3#4
1852 {
1853   \use:e
1854   {
1855     \draw_transform_matrix_absolute:nnnn
1856     { #1 * \l__draw_matrix_a_fp + #2 * \l__draw_matrix_c_fp }
1857     { #1 * \l__draw_matrix_b_fp + #2 * \l__draw_matrix_d_fp }
1858     { #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_c_fp }
1859     { #3 * \l__draw_matrix_b_fp + #4 * \l__draw_matrix_d_fp }
1860   }
1861 }
1862 \cs_new_protected:Npn \draw_transform_shift:n #1
1863 {
1864   \__draw_point_process:nn
1865   { \__draw_transform_shift:nn } {#1}
1866 }
1867 \cs_new_protected:Npn \__draw_transform_shift:nn #1#2
1868 {
1869   \dim_set:Nn \l__draw_xshift_dim { \l__draw_xshift_dim + #1 }
1870   \dim_set:Nn \l__draw_yshift_dim { \l__draw_yshift_dim + #2 }
1871 }

```

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

```

\draw_transform_matrix_invert: Standard mathematics: calculate the inverse matrix and use that, then undo the shifts.
\__draw_transform_invert:n 1872 \cs_new_protected:Npn \draw_transform_matrix_invert:
\__draw_transform_invert:f 1873 {
\draw_transform_shift_invert: 1874   \bool_if:NT \l__draw_matrix_active_bool
1875   {
1876     \__draw_transform_invert:f
1877     {
1878       \fp_eval:n
1879       {
1880         1 /
1881         (
1882           \l__draw_matrix_a_fp * \l__draw_matrix_d_fp
1883           - \l__draw_matrix_b_fp * \l__draw_matrix_c_fp
1884         )
1885       }
1886     }
1887   }
1888 }
1889 \cs_new_protected:Npn \__draw_transform_invert:n #1

```

```

1890 {
1891   \fp_set:Nn \l__draw_matrix_a_fp
1892     { \l__draw_matrix_d_fp * #1 }
1893   \fp_set:Nn \l__draw_matrix_b_fp
1894     { -\l__draw_matrix_b_fp * #1 }
1895   \fp_set:Nn \l__draw_matrix_c_fp
1896     { -\l__draw_matrix_c_fp * #1 }
1897   \fp_set:Nn \l__draw_matrix_d_fp
1898     { \l__draw_matrix_a_fp * #1 }
1899 }
1900 \cs_generate_variant:Nn \__draw_transform_invert:n { f }
1901 \cs_new_protected:Npn \draw_transform_shift_invert:
1902 {
1903   \dim_set:Nn \l__draw_xshift_dim { -\l__draw_xshift_dim }
1904   \dim_set:Nn \l__draw_yshift_dim { -\l__draw_yshift_dim }
1905 }

```

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

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

```

1906 \cs_new_protected:Npn \draw_transform_triangle:nnn #1#2#3
1907 {
1908   \__draw_point_process:nnn
1909   {
1910     \__draw_point_process:nn
1911     { \__draw_transform_triangle:nnnnnn }
1912     {#1}
1913   }
1914   {#2} {#3}
1915 }
1916 \cs_new_protected:Npn \__draw_transform_triangle:nnnnnn #1#2#3#4#5#6
1917 {
1918   \use:e
1919   {
1920     \draw_transform_matrix_absolute:nnnn
1921     { #3 - #1 }
1922     { #4 - #2 }
1923     { #5 - #1 }
1924     { #6 - #2 }
1925     \draw_transform_shift_absolute:n { #1 , #2 }
1926   }
1927 }

```

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

`\draw_transform_scale:n` Lots of shortcuts.

```

\draw_transform_xscale:n 1928 \cs_new_protected:Npn \draw_transform_scale:n #1
\draw_transform_yscale:n 1929 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { #1 } }
\draw_transform_xshift:n 1930 \cs_new_protected:Npn \draw_transform_xscale:n #1
\draw_transform_yshift:n 1931 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { 1 } }
\draw_transform_xslant:n 1932 \cs_new_protected:Npn \draw_transform_yscale:n #1
\draw_transform_yslant:n 1933 { \draw_transform_matrix:nnnn { 1 } { 0 } { 0 } { #1 } }
1934 \cs_new_protected:Npn \draw_transform_xshift:n #1
1935 { \draw_transform_shift:n { #1 , Opt } }

```

```

1936 \cs_new_protected:Npn \draw_transform_yshift:n #1
1937   { \draw_transform_shift:n { Opt , #1 } }
1938 \cs_new_protected:Npn \draw_transform_xslant:n #1
1939   { \draw_transform_matrix:nmmm { 1 } { 0 } { #1 } { 1 } }
1940 \cs_new_protected:Npn \draw_transform_yslant:n #1
1941   { \draw_transform_matrix:nmmm { 1 } { #1 } { 0 } { 1 } }

```

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

```

\draw_transform_rotate:n Slightly more involved: evaluate the angle only once, and the sine and cosine only once.
\__draw_transform_rotate:n 1942 \cs_new_protected:Npn \draw_transform_rotate:n #1
\__draw_transform_rotate:f 1943   { \__draw_transform_rotate:f { \fp_eval:n {#1} } }
\__draw_transform_rotate:nn 1944 \cs_new_protected:Npn \__draw_transform_rotate:n #1
\__draw_transform_rotate:ff 1945   {
1946     \__draw_transform_rotate:ff
1947     { \fp_eval:n { cosd(#1) } }
1948     { \fp_eval:n { sind(#1) } }
1949   }
1950 \cs_generate_variant:Nn \__draw_transform_rotate:n { f }
1951 \cs_new_protected:Npn \__draw_transform_rotate:nn #1#2
1952   { \draw_transform_matrix:nmmm {#1} {#2} { -#2 } { #1 } }
1953 \cs_generate_variant:Nn \__draw_transform_rotate:nn { ff }

```

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

```

1954 </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, 780, 1073, 1077, 1101, 1104
- bool commands:
- `\bool_gset_eq:NN` 1446
 - `\bool_gset_false:N` 1429, 1550
 - `\bool_gset_true:N` 1453, 1499
 - `\bool_if:NTF`
 - . 21, 115, 194, 233, 684, 700, 701,
 - 705, 1196, 1228, 1346, 1481, 1534, 1874
 - `\bool_lazy_all:nTF` 1820
 - `\bool_lazy_and:nnTF`
 - 225, 679, 1580, 1614
 - `\bool_lazy_any:nTF` 1625
 - `\bool_lazy_or:nnTF` . 560, 655, 688, 693
 - `\bool_new:N`
 - 86, 220, 643, 644, 645, 646,
 - 647, 747, 1252, 1337, 1413, 1452, 1793
 - `\bool_set_eq:NN` 1437
 - `\bool_set_false:N`
 - 96, 228, 666, 667, 668, 687, 1827
 - `\bool_set_true:N` 98, 229,
 - 672, 710, 714, 715, 1271, 1341, 1828
- 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` 1311, 1324
 - `\box_move_up:nn` 51
 - `\box_new:N` 13, 80, 81, 1253, 1254
 - `\box_set_dp:Nn` 55, 1315, 1328
 - `\box_set_eq:NN` 145
 - `\box_set_ht:Nn` 54, 1316, 1334
 - `\box_set_wd:Nn` 56, 128, 1300
 - `\box_use_drop:N` 52,
 - 57, 102, 129, 134, 1303, 1313, 1326
 - `\box_wd:N` 17, 66, 68
- C**
- clist commands:
- `\clist_map_inline:Nn` . . 124, 141, 152
 - `\clist_map_inline:nn` 669
 - `\clist_new:N` 87, 89
 - `\clist_set:Nn` 88, 1288
- coffin commands:
- `\coffin_typeset:Nnnnn` 64
 - `\coffin_wd:N` 66
- color commands:
- `\color_select:n` 1276
- cs commands:
- `\cs_generate_variant:Nn`
 - 420, 609, 642,
 - 839, 847, 889, 908, 915, 923, 930,
 - 939, 945, 965, 973, 980, 986, 998,
 - 1001, 1019, 1037, 1045, 1051, 1072,
 - 1086, 1100, 1121, 1156, 1175, 1188,
 - 1416, 1501, 1686, 1900, 1950, 1953
 - `\cs_if_exist:NTF` 671
 - `\cs_if_exist_use:NTF` . . 403, 412, 674
 - `\cs_new:Npn` 513, 523, 533, 543,
 - 784, 790, 792, 794, 801, 803, 805,
 - 813, 815, 818, 827, 832, 835, 837,
 - 840, 841, 843, 845, 848, 850, 856,
 - 865, 871, 881, 890, 896, 901, 909,
 - 916, 924, 931, 940, 946, 952, 957,
 - 966, 974, 981, 987, 993, 999, 1002,
 - 1008, 1017, 1020, 1026, 1031, 1038,
 - 1046, 1052, 1058, 1064, 1078, 1087,
 - 1093, 1105, 1107, 1116, 1146, 1148,
 - 1157, 1162, 1176, 1178, 1180, 1189,
 - 1194, 1221, 1226, 1681, 1687, 1744
 - `\cs_new_protected:Npn`
 - 14, 19, 60, 75, 90, 113,
 - 122, 139, 150, 184, 206, 213, 221,
 - 231, 240, 246, 252, 258, 265, 276,
 - 284, 289, 291, 293, 302, 309, 345,
 - 347, 358, 364, 394, 421, 450, 456,
 - 462, 467, 475, 484, 491, 499, 554,
 - 556, 569, 576, 585, 591, 593, 603,
 - 610, 616, 623, 648, 653, 664, 708,
 - 712, 717, 724, 748, 762, 1128, 1130,
 - 1132, 1134, 1138, 1256, 1263, 1284,
 - 1306, 1319, 1339, 1344, 1359, 1366,
 - 1377, 1386, 1394, 1402, 1414, 1417,
 - 1426, 1431, 1442, 1454, 1463, 1472,
 - 1477, 1487, 1495, 1502, 1504, 1506,
 - 1508, 1510, 1512, 1514, 1516, 1517,
 - 1519, 1521, 1532, 1552, 1576, 1586,
 - 1603, 1612, 1623, 1644, 1653, 1712,
 - 1745, 1760, 1765, 1780, 1782, 1783,
 - 1784, 1785, 1786, 1787, 1788, 1789,
 - 1800, 1807, 1814, 1830, 1835, 1840,
 - 1851, 1862, 1867, 1872, 1889, 1901,
 - 1906, 1916, 1928, 1930, 1932, 1934,
 - 1936, 1938, 1940, 1942, 1944, 1951

D

dim commands:	
<ul style="list-style-type: none"> \dim_abs:n 598, 599 \dim_compare:nNnTF 605, 612, 726, 1292 \dim_compare_p:nNn 226, 227, 1581, 1582 \dim_eval:n 598, 599 \dim_gset:Nn 186, 188, 190, 192, 196, 198, 200, 202, 208, 209, 210, 211, 215, 216, 728, 1258, 1259, 1260, 1261, 1483, 1484, 1762 \dim_gset_eq:NN 765, 766, 767, 768, 769, 770, 771, 772, 1369, 1388, 1389, 1390, 1391 \dim_gzero:N .. 1294, 1295, 1296, 1297 \dim_max:nn .. 187, 191, 197, 201, 1330 \dim_min:nn 189, 193, 199, 203 \dim_new:N 178, 179, 180, 181, 182, 183, 218, 219, 739, 740, 741, 742, 743, 744, 745, 746, 1122, 1123, 1124, 1125, 1126, 1127, 1248, 1249, 1250, 1251, 1338, 1356, 1373, 1374, 1375, 1376, 1450, 1451, 1527, 1528, 1757, 1758, 1798, 1799 \dim_set:Nn 223, 224, 1140, 1141, 1342, 1578, 1579, 1759, 1837, 1838, 1869, 1870, 1903, 1904 \dim_set_eq:NN 751, 752, 753, 754, 755, 756, 757, 758, 1363, 1380, 1381, 1382, 1383 \dim_step_inline:nnnn 625, 633 \dim_use:N 726, 731, 733, 1351, 1459, 1460 \dim_zero:N 1809, 1810 \c_max_dim 208, 209, 210, 211, 726, 1258, 1259, 1260, 1261, 1292 	<ul style="list-style-type: none"> \draw_layer_end: 90, 113 \draw_layer_new:n 75, 75 \l_draw_layers_clist 87, 124, 141, 152, 1288 \draw_linewidth:n 105, 1275, 1760, 1760 \draw_miterlimit:n .. 1280, 1780, 1780 \draw_nonzero_rule: . 1277, 1782, 1786 \draw_path_arc:nnn 345, 345, 488 \draw_path_arc:nnnn ... 345, 346, 347 \draw_path_arc_axes:nnnn ... 484, 484 \draw_path_canvas_curveto:nnn ... 289, 293 \draw_path_canvas_lineto:n . 289, 291 \draw_path_canvas_moveto:n . 289, 289 \draw_path_circle:nn 554, 554 \draw_path_close: 284, 284, 582 \draw_path_corner_arc:nn ... 221, 221 \draw_path_curveto:nn 302, 302 \draw_path_curveto:nnn 240, 265 \draw_path_ellipse:nnn . 491, 491, 555 \draw_path_grid:nnnn 593, 593 \draw_path_lineto:n 240, 252, 579, 580, 581, 631, 639 \draw_path_moveto:n 240, 240, 578, 583, 630, 638 \draw_path_rectangle:nn 556, 556, 592 \draw_path_rectangle_corners:nn . 585, 585 \draw_path_scope_begin: 748, 748, 1364, 1397 \draw_path_scope_end: 748, 762, 1368, 1405 \draw_path_use:n 648, 648 \draw_path_use_clear:n 648, 653 \draw_point:n 787, 797, 798, 808, 809, 810, 821, 822, 823, 824, 835, 835, 846, 861, 883, 942, 983, 1000, 1018, 1048, 1118, 1150, 1164, 1182, 1198, 1214, 1230, 1243 \draw_point_interpolate_arcaxes:nnnnnn 1020, 1020 \draw_point_interpolate_curve:nnnnnn 1052 \draw_point_interpolate_curve:nnnnnn 1052 \draw_point_interpolate_curve_ -auxi:nnnnnnnn 1052 \draw_point_interpolate_curve_ -auxii:nnnnnnnn 1052 \draw_point_interpolate_curve_ -auxiii:nnnnnn 1052 \draw_point_interpolate_curve_ -auxiv:nnnnnn 1052
draw commands:	
<ul style="list-style-type: none"> \draw_baseline:n 1339, 1339 \l_draw_bb_update_bool 21, 194, 680, 687, 1252, 1271 \draw_begin: 1263, 1263 \draw_box_use:N 14, 14 \draw_cap_but: 1278, 1782, 1782 \draw_cap_rectangle: 1782, 1783 \draw_cap_round: 1782, 1784 \draw_coffin_use:Nnn 35, 60, 60 \draw_dash_pattern:nn 1281, 1765, 1765 \l_draw_default_linewidth_dim ... 105, 1275, 1758 \draw_end: 1263, 1284 \draw_evenodd_rule: 1782, 1785 \draw_join_bevel: 1782, 1787 \draw_join_miter: ... 1279, 1782, 1788 \draw_join_round: 1782, 1789 \draw_layer_begin:n 90, 90 	

<code>\draw_point_interpolate_curve_</code>	<code>\draw_transform_triangle:nnn</code> . . .
<code>auxv:nnw</code> 1052 487 , 1906 , 1906
<code>\draw_point_interpolate_curve_</code>	<code>\draw_transform_xscale:n</code> . . 1928 , 1930
<code>auxvi:n</code> 1052	<code>\draw_transform_xshift:n</code> . . 1928 , 1934
<code>\draw_point_interpolate_curve_</code>	<code>\draw_transform_xslant:n</code> . . 1928 , 1938
<code>auxvii:nnnnnnn</code> 1052	<code>\draw_transform_yscale:n</code> . . 1928 , 1932
<code>\draw_point_interpolate_curve_</code>	<code>\draw_transform_yshift:n</code> . . 1928 , 1936
<code>auxviii:nnnnn</code> 1052	<code>\draw_transform_yslant:n</code> . . 1928 , 1940
<code>\draw_point_interpolate_distance:nnn</code>	<code>\draw_xvec:n</code> 1128 , 1128 , 1143
. 1002 , 1002 , 1657 , 1666	<code>\draw_yvec:n</code> 1128 , 1130 , 1144
<code>\draw_point_interpolate_line:nnn</code>	<code>\draw_zvec:n</code> 1128 , 1132 , 1145
. 987 , 987	draw internal commands:
<code>\draw_point_intersect_circles:nnnnn</code>	<code>_draw_backend_begin:</code> 1268
. 890 , 890	<code>_draw_backend_box_use:Nnnnn</code> 41
<code>\draw_point_intersect_line_</code>	<code>_draw_backend_cap_but:</code> 1782
<code>circle:nnnnn</code> 946 , 946	<code>_draw_backend_cap_rectangle:</code> 1783
<code>\draw_point_intersect_lines:nnnn</code>	<code>_draw_backend_cap_round:</code> 1784
. 865 , 865	<code>_draw_backend_clip:</code> 686
<code>\draw_point_polar:nn</code> 841 , 841	<code>_draw_backend_closepath:</code> 1503
<code>\draw_point_polar:nnn</code>	<code>_draw_backend_curveto:nnnnnn</code> 1507
. . . . 428 , 434 , 438 , 444 , 841 , 842 , 843	<code>_draw_backend_dash_pattern:nn</code> 1773
<code>\draw_point_transform:n</code> 25 ,	<code>_draw_backend_discardpath:</code> 691
28 , 31 , 34 , 244 , 256 , 272 , 273 , 274 ,	<code>_draw_backend_end:</code> 1290
306 , 307 , 433 , 437 , 495 , 566 , 1189 , 1189	<code>_draw_backend_evenodd_rule:</code> 1785
<code>\draw_point_unit_vector:n</code>	<code>_draw_backend_join_bevel:</code> 1787
. 848 , 848 , 1015	<code>_draw_backend_join_miter:</code> 1788
<code>\draw_point_vec:nn</code> 1146 , 1146	<code>_draw_backend_join_round:</code> 1789
<code>\draw_point_vec:nnn</code> 1146 , 1157	<code>_draw_backend_lineto:nn</code> 1513
<code>\draw_point_vec_polar:nn</code> 1176 , 1176	<code>_draw_backend_linewidth:n</code> 1763
<code>\draw_point_vec_polar:nnn</code>	<code>_draw_backend_miterlimit:n</code> 1781
. 1176 , 1177 , 1178	<code>_draw_backend_moveto:nn</code> 1515
<code>\draw_scope_begin:</code> 1359 , 1359	<code>_draw_backend_nonzero_rule:</code> 1786
<code>\draw_scope_end:</code> 1366	<code>_draw_backend_rectangle:nnnn</code> 1520
<code>\draw_suspend_begin:</code> 1394 , 1394	<code>_draw_backend_scope_begin:</code>
<code>\draw_suspend_end:</code> 1394 , 1402 132 , 1361
<code>\draw_transform_matrix:nnnn</code> 135 , 1371
. 1840 , 1840 ,	<code>\l_draw_baseline_bool</code>
1929 , 1931 , 1933 , 1939 , 1941 , 1952 1337 , 1341 , 1346
<code>\draw_transform_matrix_absolute:nnnn</code>	<code>\l_draw_baseline_dim</code> 1337 , 1342 , 1351
. 1814 , 1814 , 1855 , 1920	<code>_draw_baseline_finalise:w</code>
<code>\draw_transform_matrix_invert:</code> 1286 , 1344 , 1344
. 1872 , 1872	<code>_draw_box_use:Nnnnn</code> 14 , 16 , 19 , 65
<code>\draw_transform_matrix_reset:</code>	<code>\l_draw_corner_arc_bool</code>
. 1272 , 1398 , 1800 , 1800 , 1812 220 , 228 , 229 , 233 , 561
<code>\draw_transform_rotate:n</code> 1942 , 1942	<code>\l_draw_corner_xarc_dim</code>
<code>\draw_transform_scale:n</code> 1928 , 1928 218 , 223 , 226 , 236
<code>\draw_transform_shift:n</code>	<code>\l_draw_corner_yarc_dim</code>
. 1840 , 1862 , 1935 , 1937 218 , 224 , 227 , 237
<code>\draw_transform_shift_absolute:n</code>	<code>_draw_draw_polar:nnn</code>
. 1814 , 1830 , 1925 841 , 844 , 845 , 847
<code>\draw_transform_shift_invert:</code>	<code>_draw_draw_vec_polar:nnn</code>
. 1872 , 1901 1179 , 1180 , 1188
<code>\draw_transform_shift_reset:</code>	<code>\l_draw_fill_color_tl</code> 1356
. 1273 , 1399 , 1800 , 1807 , 1813	

<code>__draw_finalise:</code>	<code>__draw_path_arc_auxvi:nn</code>
..... 1299, 1306, 1306, 1344, 1354 345, 464, 467
<code>_draw_finalise_baseline:n</code>	<code>\l__draw_path_arc_delta_fp</code> 345
..... 1306, 1319, 1351	<code>\l__draw_path_arc_start_fp</code> 345
<code>\g__draw_id_int</code> 1255, 1266	<code>__draw_path_curveto:nnnn</code>
<code>__draw_if_recursion_tail_stop_</code> 302, 305, 309
<code>do:Nn</code> 9, 9, 1554	<code>__draw_path_curveto:nnnnnn</code>
<code>\l__draw_layer_close_bool</code> 240, 270,
..... 86, 96, 98, 115	276, 298, 316, 446, 515, 525, 535, 545
<code>\l__draw_layer_main_box</code>	<code>\c__draw_path_curveto_a_fp</code> 302
..... 128, 129, 1253, 1282	<code>\c__draw_path_curveto_b_fp</code> 302
<code>\l__draw_layer_tl</code> 84, 95, 99	<code>__draw_path_ellipse:nnnnnn</code>
<code>\g__draw_layers_clist</code> 87 491, 494, 499
<code>__draw_layers_insert:</code> 122, 122, 1289	<code>__draw_path_ellipse_arci:nnnnnn</code>
<code>__draw_layers_restore:</code> 139, 150, 1404 491, 505, 513
<code>__draw_layers_save:</code> . 139, 139, 1400	<code>__draw_path_ellipse_arci:nnnnnn</code>
<code>\g__draw_linewidth_dim</code> 491, 506, 523
... 734, 1363, 1369, 1757, 1762, 1763	<code>__draw_path_ellipse_arci:nnnnnn</code>
<code>\l__draw_linewidth_dim</code> 491, 507, 533
..... 1356, 1363, 1369	<code>__draw_path_ellipse_arci:nnnnnn</code>
<code>\l__draw_main_box</code> 491, 508, 543
1253, 1267, 1300, 1303, 1308, 1313,	<code>\c__draw_path_ellipse_fp</code> 491
1315, 1316, 1321, 1326, 1328, 1334	<code>__draw_path_grid_auxi:nnnnnn</code> ...
<code>\l__draw_matrix_a_fp</code> 593, 597, 603, 609
. 42, 1201, 1233, 1794, 1802, 1816,	<code>__draw_path_grid_auxii:nnnnnn</code> ..
1822, 1856, 1858, 1882, 1891, 1898 593, 606, 607, 610
<code>\l__draw_matrix_active_bool</code>	<code>__draw_path_grid_auxiii:nnnnnn</code> .
562, 1196, 1228, 1793, 1827, 1828, 1874 593, 613, 614, 616
<code>\l__draw_matrix_b_fp</code>	<code>__draw_path_grid_auxiiii:nnnnnn</code> 593
. 43, 1207, 1238, 1794, 1803, 1817,	<code>__draw_path_grid_auxiv:nnnnnnnn</code>
1823, 1857, 1859, 1883, 1893, 1894 593, 618, 623, 642
<code>\l__draw_matrix_c_fp</code>	<code>\g__draw_path_lastx_dim</code>
. 44, 1202, 1234, 1794, 1804, 1818, 178, 215, 320, 453, 459, 751, 771
1824, 1856, 1858, 1883, 1895, 1896	<code>\l__draw_path_lastx_dim</code> 739, 751, 771
<code>\l__draw_matrix_d_fp</code>	<code>\g__draw_path_lasty_dim</code>
. 45, 1208, 1239, 1797, 1805, 1819, 178, 216, 327, 454, 460, 752, 772
1825, 1857, 1859, 1882, 1892, 1897	<code>\l__draw_path_lasty_dim</code> 739, 752, 772
<code>__draw_path_arc:nmnn</code> . 345, 351, 358	<code>__draw_path_lineto:nn</code>
<code>__draw_path_arc:nnNnn</code> 240, 255, 258, 292
..... 345, 361, 362, 364	<code>__draw_path_mark_corner:</code>
<code>\c__draw_path_arc_60_fp</code> 345	231, 231, 260, 269, 286, 297, 315, 386
<code>\c__draw_path_arc_90_fp</code> 345	<code>__draw_path_moveto:nn</code>
<code>__draw_path_arc_add:nnnn</code> 345 240, 243, 246, 290, 503, 511
<code>__draw_path_arc_aux_add:nn</code>	<code>__draw_path_rectangle:nmnn</code>
..... 452, 458, 470, 475 556, 564, 569
<code>__draw_path_arc_auxi:nnnnNnn</code> ...	<code>__draw_path_rectangle_corners:nmnn</code>
..... 345, 372, 379, 387, 394, 420 585
<code>__draw_path_arc_auxii:nnnNnnnn</code> .	<code>__draw_path_rectangle_corners:nnnnn</code>
..... 345, 398, 421 588, 591
<code>__draw_path_arc_auxiii:nn</code>	<code>__draw_path_rectangle_rounded:nmnn</code>
..... 345, 425, 450 556, 563, 576
<code>__draw_path_arc_auxiv:nnnn</code>	<code>__draw_path_reset_limits:</code>
..... 345, 431, 456 184, 206, 660, 759, 1270
<code>__draw_path_arc_auxv:nn</code> 345, 441, 462	

<code>\l__draw_path_tmp_tl</code>	<code>__draw_point_interpolate_curve_-</code>
.... 175 , 423 , 446 , 465 , 469 , 473 , 477	auxii:nnnnnnnnn . 1060 , 1064 , 1072
<code>\l__draw_path_tmpa_fp</code>	<code>__draw_point_interpolate_curve_-</code>
..... 175 , 311 , 321 , 333	auxiii:nnnnnn . 1067 , 1078 , 1086
<code>\l__draw_path_tmpb_fp</code>	<code>__draw_point_interpolate_curve_-</code>
..... 175 , 312 , 328 , 337	auxiv:nnnnnn 1080 , 1081 , 1082 , 1087
<code>__draw_path_update_last:nn</code>	<code>__draw_point_interpolate_curve_-</code>
..... 213 , 213 , 250 , 263 , 282 , 574	auxv:nnw 1089 , 1093 , 1100
<code>__draw_path_update_limits:nn</code> ...	<code>__draw_point_interpolate_curve_-</code>
..... 24 , 27 , 30 , 33 , 184 ,	auxvi:n 1084 , 1105
184 , 248 , 261 , 278 , 279 , 280 , 571 , 572	<code>__draw_point_interpolate_curve_-</code>
<code>__draw_path_use:n</code> . 648 , 651 , 662 , 664	auxvii:nnnnnnnn 1106 , 1107
<code>__draw_path_use_action_draw:</code> ...	<code>__draw_point_interpolate_curve_-</code>
..... 648 , 708	auxviii:nnnnnn . 1109 , 1116 , 1121
<code>__draw_path_use_action_fillstroke:</code>	<code>__draw_point_interpolate_-</code>
..... 648 , 712	distance:nnnn 1005 , 1008
<code>\l__draw_path_use_bb_bool</code> 646	<code>__draw_point_interpolate_-</code>
<code>\l__draw_path_use_clear_bool</code> 646 , 705	distance:nnnn 1002 , 1012 , 1017 , 1019
<code>\l__draw_path_use_clip_bool</code>	<code>__draw_point_interpolate_-</code>
..... 643 , 666 , 684	distance:nnnnnn 1002
<code>\l__draw_path_use_fill_bool</code>	<code>__draw_point_interpolate_line_-</code>
..... 643 , 667 , 689 , 694 , 700 , 714	aux:nnnnnn 987 , 990 , 993 , 998
<code>__draw_path_use_stroke_bb:</code>	<code>__draw_point_interpolate_line_-</code>
..... 648 , 682 , 717	aux:nnnnnn . . . 987 , 995 , 999 , 1001
<code>__draw_path_use_stroke_bb_-</code>	<code>__draw_point_intersect_circles_-</code>
aux:NnN 648 , 719 , 720 , 721 , 722 , 724	auxi:nnnnnnnn 890 , 893 , 896
<code>\l__draw_path_use_stroke_bool</code> ...	<code>__draw_point_intersect_circles_-</code>
643 , 668 , 681 , 690 , 695 , 701 , 710 , 715	auxii:nnnnnnnn . . 890 , 898 , 901 , 908
<code>\g__draw_path_xmax_dim</code>	<code>__draw_point_intersect_circles_-</code>
..... 180 , 186 , 187 , 208 , 753 , 767	auxiii:nnnnnnnn . 890 , 903 , 909 , 915
<code>\l__draw_path_xmax_dim</code> . 739 , 753 , 767	<code>__draw_point_intersect_circles_-</code>
<code>\g__draw_path_xmin_dim</code>	auxiv:nnnnnnnn . 890 , 911 , 916 , 923
..... 180 , 188 , 189 , 209 , 754 , 768	<code>__draw_point_intersect_circles_-</code>
<code>\l__draw_path_xmin_dim</code> . 739 , 754 , 768	auxv:nnnnnnnnnn . 890 , 918 , 924 , 930
<code>\g__draw_path_ymax_dim</code>	<code>__draw_point_intersect_circles_-</code>
..... 180 , 190 , 191 , 210 , 755 , 769	auxvi:nnnnnnnn . 890 , 926 , 931 , 939
<code>\l__draw_path_ymax_dim</code> . 739 , 755 , 769	<code>__draw_point_intersect_circles_-</code>
<code>\g__draw_path_ymin_dim</code>	auxvii:nnnnnnnn . 890 , 933 , 940 , 945
..... 180 , 192 , 193 , 211 , 756 , 770	<code>__draw_point_intersect_line_-</code>
<code>\l__draw_path_ymin_dim</code> . 739 , 756 , 770	circle_auxi:nnnnnnnn 946 , 949 , 952
<code>__draw_point_interpolate_-</code>	<code>__draw_point_intersect_line_-</code>
arcaxes_auxi:nnnnnnnnnn 1020 , 1023 , 1026	circle_auxii:nnnnnnnn 946 , 954 , 957 , 965
<code>__draw_point_interpolate_-</code>	<code>__draw_point_intersect_line_-</code>
arcaxes_auxii:nnnnnnnnnn 1020 , 1028 , 1031 , 1037	circle_auxiii:nnnnnnnn 946 , 959 , 966 , 973
<code>__draw_point_interpolate_-</code>	<code>__draw_point_intersect_line_-</code>
arcaxes_auxiii:nnnnnnnn 1020 , 1033 , 1038 , 1045	circle_auxiv:nnnnnnnn 946 , 968 , 974 , 980
<code>__draw_point_interpolate_-</code>	<code>__draw_point_intersect_line_-</code>
arcaxes_auxiv:nnnnnnnn 1020 , 1040 , 1046 , 1051	circle_auxv:nnnnnn 946 , 976 , 981 , 986
<code>__draw_point_interpolate_curve_-</code>	<code>__draw_point_intersect_lines:nnnnnn</code>
auxi:nnnnnnnnnn 1055 , 1058 865

`__draw_point_intersect_lines:nnnnnnnn` 865, 868, 871
`__draw_point_intersect_lines_-aux:nnnnnn` 865, 873, 881, 889
`__draw_point_process:nn` 23, 26, 29, 32, 242, 254, 290, 292, 424, 440, 784, 784, 849, 1004, 1010, 1136, 1191, 1223, 1832, 1864, 1910
`__draw_point_process:nnn` 304, 430, 558, 587, 595, 784, 794, 892, 989, 1908
`__draw_point_process:nnnn` 267, 295, 493, 784, 805, 948, 1022
`__draw_point_process:nnnnn` 784, 818, 867, 1054
`__draw_point_process_auxi:nn` 784, 786, 790
`__draw_point_process_auxii:nw` 784, 791, 792
`__draw_point_process_auxiii:nnn` 784, 796, 801
`__draw_point_process_auxiv:nw` 784, 802, 803
`__draw_point_process_auxv:nnnn` 784, 807, 813
`__draw_point_process_auxvi:nw` 784, 814, 815
`__draw_point_process_auxvii:nnnnn` 784, 820, 827
`__draw_point_process_auxviii:nw` 784, 829, 832
`__draw_point_to_dim:n` 835, 836, 837, 839
`__draw_point_to_dim:w` . 835, 838, 840
`__draw_point_transform:nn` 1189, 1192, 1194
`__draw_point_transform_noshift:n` 427, 443, 496, 497, 1221, 1221
`__draw_point_transform_noshift:nn` 1221, 1224, 1226
`__draw_point_unit_vector:nn` 848, 849, 850
`__draw_point_unit_vector:nnn` 848, 852, 856
`__draw_point_vec:nn` 1146, 1147, 1148, 1156
`__draw_point_vec:nnn` 1146, 1159, 1162, 1175
`__draw_point_vec_polar:nnn` . . 1176
`__draw_reset_bb:` 1256, 1256, 1269, 1384
`__draw_scope_bb_begin:` 1377, 1377, 1396
`__draw_scope_bb_end:` 1377, 1386, 1406
`__draw_softpath_add:n` 1414, 1414, 1416, 1445, 1456, 1465, 1474, 1479, 1489, 1497, 1752
`\c__draw_softpath_arc_fp` 1531, 1694, 1698, 1703, 1707
`__draw_softpath_clear:` 659, 706, 1274, 1417, 1426, 1440, 1444
`__draw_softpath_close_op:nn` . . . 1458, 1502, 1502, 1593, 1629, 1731
`__draw_softpath_closepath:` 287, 510, 1454
`\l__draw_softpath_corneri_dim` 1525, 1578, 1581, 1667
`\l__draw_softpath_cornerii_dim` 1525, 1579, 1582, 1658
`\g__draw_softpath_corners_bool` 1413, 1429, 1439, 1447, 1499, 1534, 1550
`\l__draw_softpath_corners_bool` 739, 1438, 1448
`\l__draw_softpath_curve_end_tl` 1524, 1655, 1674, 1723, 1734
`__draw_softpath_curveto:nnnnnn` 281, 1454, 1463
`__draw_softpath_curveto_opi:nn` . . . 1467, 1502, 1504, 1590, 1628, 1691
`__draw_softpath_curveto_opi:nnNnnNnn` 1502, 1505, 1506
`__draw_softpath_curveto_opiii:nn` 1468, 1502, 1508, 1509, 1700
`__draw_softpath_curveto_opiiii:nn` 1469, 1502, 1510, 1511, 1709
`\l__draw_softpath_first_tl` 1525, 1541, 1558, 1559, 1569, 1588, 1589, 1615, 1619
`\g__draw_softpath_lastx_dim` 757, 765, 1450, 1459, 1483
`\l__draw_softpath_lastx_dim` 745, 757, 765
`\l__draw_softpath_lastx_fp` 1525, 1539, 1560, 1608, 1670, 1677
`\g__draw_softpath_lasty_dim` 758, 766, 1450, 1460, 1484
`\l__draw_softpath_lasty_dim` 746, 758, 766
`\l__draw_softpath_lasty_fp` 1525, 1540, 1561, 1609, 1671, 1678
`__draw_softpath_lineto:nn` 262, 1454, 1472
`__draw_softpath_lineto_op:nn` 1475, 1502, 1512, 1596, 1627, 1740
`\g__draw_softpath_main_tl` 1411, 1415, 1419, 1420, 1422, 1424, 1428, 1433, 1436, 1543, 1545, 1751

\l__draw_softpath_main_tl	__draw_softpath_roundpoint:nn
. 19, 1435, 1445, 1522, 235, 1454, 1495, 1501
1537, 1564, 1566, 1747, 1749, 1752	__draw_softpath_roundpoint_-
\g__draw_softpath_move_bool	op:nn . . . 1498, 1502, 1516, 1555, 1637
. 1452, 1481	__draw_softpath_save: 760, 1431, 1431
\l__draw_softpath_move_tl	\l__draw_softpath_tmp_tl
. 1525, 1542, 1412, 1420, 1421, 1424
1565, 1568, 1616, 1718, 1741, 1748	__draw_softpath_use: 683, 1417, 1417
__draw_softpath_moveto:nn	\l__draw_stroke_color_tl 1356
. 249, 1454, 1477	\l__draw_tmp_box 13, 37, 48,
__draw_softpath_moveto_op:nn	52, 54, 55, 56, 57, 63, 65, 66, 67, 68, 69
. 1480, 1502, 1514, 1562, 1720	\l__draw_tmp_seq 1765
\l__draw_softpath_part_tl	__draw_transform:nnnn
. 1523, 1538, 1840, 1844, 1851
1567, 1570, 1572, 1606, 1661, 1750	__draw_transform_invert:n
__draw_softpath_rectangle:nnnn 1872, 1876, 1889, 1900
. 573, 1454, 1487	__draw_transform_rotate:n
__draw_softpath_rectangle_- 1942, 1943, 1944, 1950
opi:nn 1491, 1502, 1517	__draw_transform_rotate:nn
__draw_softpath_rectangle_- 1942, 1946, 1951, 1953
opi:nnNnn 1502, 1518, 1519	__draw_transform_shift:nn
__draw_softpath_rectangle_- 1840, 1865, 1867
opii:nn 1492, 1502, 1521	__draw_transform_shift_absolute:nn
__draw_softpath_restore: 1814, 1833, 1835
. 764, 1431, 1442	__draw_transform_triangle:nnnnnn
__draw_softpath_round_action:nn 1911, 1916
. 1532, 1556, 1576	__draw_vec:nn
__draw_softpath_round_action:Nnn 1128, 1129, 1131, 1133, 1134
. 1532, 1584, 1586	__draw_vec:nnn 1128, 1136, 1138
__draw_softpath_round_action_-	\g__draw_xmax_dim 196,
close: 1532, 1594, 1612	197, 1248, 1258, 1294, 1301, 1380, 1388
__draw_softpath_round_action_-	\l__draw_xmax_dim . . . 1373, 1380, 1388
curveto:NnnNnn . . . 1532, 1591, 1603	\g__draw_xmin_dim
__draw_softpath_round_calc:NnnNnn 198, 199, 1248, 1259, 1292,
. 1532, 1632, 1647, 1653, 1716	1295, 1301, 1310, 1323, 1381, 1389
__draw_softpath_round_calc:nnnnnn	\l__draw_xmin_dim . . . 1373, 1381, 1389
. 1532, 1664, 1681, 1686	\l__draw_xshift_dim 50, 1203,
__draw_softpath_round_calc:nnnnw	1217, 1794, 1809, 1837, 1869, 1903
. 1532, 1683, 1687	\l__draw_xvec_x_dim
__draw_softpath_round_close:nn 1122, 1152, 1166, 1184
. 1532, 1618, 1712	\l__draw_xvec_y_dim . . 1122, 1153, 1170
__draw_softpath_round_close:w	\g__draw_ymax_dim . . . 200, 201, 1248,
. 1532, 1722, 1733, 1744	1260, 1296, 1317, 1335, 1382, 1390
__draw_softpath_round_corners:	\l__draw_ymax_dim . . . 1373, 1382, 1390
. 678, 1532, 1532	\g__draw_ymin_dim
__draw_softpath_round_end: 202, 203, 1248, 1261,
. 1532, 1554, 1745	1297, 1312, 1317, 1331, 1383, 1391
__draw_softpath_round_lookahead:NnnNnn	\l__draw_ymin_dim . . . 1373, 1383, 1391
. 1532, 1597, 1610, 1623	\l__draw_yshift_dim 51, 1209,
__draw_softpath_round_loop:Nnn	1217, 1794, 1810, 1838, 1870, 1904
. . . 1532, 1544, 1552, 1573, 1583,	\l__draw_yvec_x_dim . . 1122, 1152, 1167
1598, 1621, 1633, 1639, 1648, 1730	\l__draw_yvec_y_dim
__draw_softpath_round_roundpoint:NnnNnnNnn 1122, 1153, 1171, 1185
. 1532, 1638, 1644	\l__draw_zvec_x_dim 1122, 1168

<code>\l__draw_zvec_y_dim</code>	1122, 1172		
E		G	
<code>\end</code>	169, 778	group commands:	
exp commands:		<code>\group_begin:</code>	36, 62, 92, 103, 486, 750, 1265, 1362, 1379, 1536, 1767
<code>\exp_after:wN</code>		<code>\group_end:</code>	58, 70, 117, 120, 489, 773, 1304, 1370, 1392, 1548, 1777
446, 464, 1544, 1618, 1721, 1732, 1741		H	
<code>\exp_args:Ne</code>	1781	hbox commands:	
<code>\exp_args:Nf</code>	786, 852	<code>\hbox_gset:Nw</code>	101
<code>\exp_args:Nff</code>	796	<code>\hbox_gset_end:</code>	118
<code>\exp_args:Nfff</code>	807	<code>\hbox_set:Nn</code>	37, 48, 63, 1308, 1321
<code>\exp_args:Nffff</code>	820	<code>\hbox_set:Nw</code>	1267, 1282
<code>\exp_args:NNNV</code>	1287	<code>\hbox_set_end:</code>	1287, 1291
<code>\exp_args:NNV</code>	1423	I	
<code>\exp_not:N</code>		int commands:	
1663, 1691, 1700, 1709, 1718, 1721,		<code>\int_gincr:N</code>	1266
1722, 1723, 1728, 1732, 1733, 1734		<code>\int_if_odd:nTF</code>	935, 970
<code>\exp_not:n</code>	1350	<code>\int_new:N</code>	1255
F		K	
fp commands:		kernel internal commands:	
<code>\fp_compare:nNnTF</code>	360, 370, 858	<code>_kernel_kern:n</code>	50
<code>\fp_compare_p:nNn</code>		<code>_kernel_quark_new_test:N</code>	9
1822, 1823, 1824, 1825		M	
<code>\fp_const:Nn</code>		mode commands:	
343, 344, 482, 483, 553, 1531		<code>\mode_leave_vertical:</code>	1302
<code>\fp_eval:n</code>	352, 353, 374, 381,	msg commands:	
390, 836, 844, 853, 874, 875, 876,		<code>\msg_error:nnn</code>	78, 109, 110, 675
877, 878, 879, 899, 904, 905, 912,		<code>\msg_new:nnn</code>	164
919, 920, 927, 934, 936, 955, 960,		<code>\msg_new:nnnn</code>	161, 166, 775
961, 962, 969, 977, 990, 995, 1013,		P	
1029, 1034, 1041, 1042, 1061, 1068,		<code>\pgfextractx</code>	20
1090, 1091, 1110, 1111, 1112, 1113,		<code>\pgfextracty</code>	20
1147, 1160, 1179, 1781, 1845, 1846,		<code>\pgfgetlastxy</code>	20
1847, 1848, 1878, 1943, 1947, 1948		<code>\pgfgettransform</code>	50
<code>\fp_new:N</code>	176, 177, 480,	<code>\pgfgettransformentries</code>	50
481, 1525, 1526, 1794, 1795, 1796, 1797		<code>\pgfinnerlinewidth</code>	49
<code>\fp_set:Nn</code>	311, 312, 366, 367,	<code>\pgflowlevel</code>	50
447, 448, 1560, 1561, 1608, 1609,		<code>\pgflowlevelsynccm</code>	50
1677, 1678, 1802, 1805, 1816, 1817,		<code>\pgfpatharcto</code>	6
1818, 1819, 1891, 1893, 1895, 1897		<code>\pgfpatharctoprecomputed</code>	6
<code>\fp_to_decimal:N</code>	373, 380, 388	<code>\pgfpathcosine</code>	6
<code>\fp_to_dim:n</code>	318, 325, 332,	<code>\pgfpathcurvebetweentime</code>	6
336, 354, 355, 401, 410, 478, 504,		<code>\pgfpathcurvebetweentimecontinue</code>	6
516, 517, 518, 519, 520, 521, 526,		<code>\pgfpathparabola</code>	6
527, 528, 529, 530, 531, 536, 537,		<code>\pgfpathsine</code>	6
538, 539, 540, 541, 546, 547, 548,		<code>\pgfpointadd</code>	20
549, 550, 551, 619, 620, 1342, 1693,		<code>\pgfpointborderellipse</code>	21
1697, 1702, 1706, 1762, 1770, 1775		<code>\pgfpointborderrectangle</code>	21
<code>\fp_use:N</code>	42, 43, 44, 45, 553	<code>\pgfpointcylindrical</code>	21
<code>\fp_while_do:nNnn</code>	368	<code>\pgfpointdiff</code>	20
<code>\fp_zero:N</code>	1539, 1540, 1803, 1804		
<code>\c_one_fp</code>	1822, 1825		
<code>\c_zero_fp</code>	858, 1823, 1824		

