

File I

Implementation

1 l3draw implementation

```
1 <*package>
2 <@@=draw>
3 \ProvidesExplPackage{l3draw}{2026-02-18}{}
4 {L3 Experimental core drawing support}
```

1.1 Internal auxiliaries

`\s__draw_mark` Internal scan marks.

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

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

`\q__draw_recursion_tail` Internal recursion quarks.

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

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

`__draw_if_recursion_tail_stop_do:Nn` Functions to query recursion quarks.

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

(End of definition for __draw_if_recursion_tail_stop_do:Nn.)

Everything else is in the sub-files!

```
10 </package>
```

2 l3draw-boxes implementation

```
11 <*package>
```

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

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

`\l__draw_tmp_box`

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

(End of definition for \l__draw_tmp_box.)

`\draw_box_use:N`

`\draw_box_use:Nn`

`__draw_box_use:nNnnnnn`

`__draw_box_use:Nnnnn`

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

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

```
15 {
```

```

16     \_draw_box_use:Nnnnnn #1
17     { Opt } { -\box_dp:N #1 } { \box_wd:N #1 } { \box_ht:N #1 }
18 }
19 \cs_new_protected:Npn \draw_box_use:Nn #1#2
20 {
21     \_draw_box_use:nNnnnn {#2} #1
22     { Opt } { -\box_dp:N #1 } { \box_wd:N #1 } { \box_ht:N #1 }
23 }
24 \cs_new_protected:Npn \_draw_box_use:nNnnnn #1#2#3#4#5#6
25 {
26     \draw_scope_begin:
27     \draw_transform_shift:n {#1}
28     \_draw_box_use:Nnnnnn #2 {#3} {#4} {#5} {#6}
29     \draw_scope_end:
30 }
31 \cs_new_protected:Npn \_draw_box_use:Nnnnnn #1#2#3#4#5
32 {
33     \bool_if:NT \l_draw_bb_update_bool
34     {
35         \_draw_point_process:nn
36         { \_draw_path_update_limits:nn }
37         { \draw_point_transform:n { #2 , #3 } }
38         \_draw_point_process:nn
39         { \_draw_path_update_limits:nn }
40         { \draw_point_transform:n { #4 , #3 } }
41         \_draw_point_process:nn
42         { \_draw_path_update_limits:nn }
43         { \draw_point_transform:n { #4 , #5 } }
44         \_draw_point_process:nn
45         { \_draw_path_update_limits:nn }
46         { \draw_point_transform:n { #2 , #5 } }
47     }
48     \group_begin:
49     \hbox_set:Nn \l__draw_tmp_box
50     {
51         \use:e
52         {
53             \_draw_backend_box_use:Nnnnn #1
54             { \fp_use:N \l__draw_matrix_a_fp }
55             { \fp_use:N \l__draw_matrix_b_fp }
56             { \fp_use:N \l__draw_matrix_c_fp }
57             { \fp_use:N \l__draw_matrix_d_fp }
58         }
59     }
60     \hbox_set:Nn \l__draw_tmp_box
61     {
62         \dim_horizontal:N \l__draw_xshift_dim
63         \box_move_up:nn { \l__draw_yshift_dim }
64         { \box_use_drop:N \l__draw_tmp_box }
65     }
66     \box_set_ht:Nn \l__draw_tmp_box { Opt }
67     \box_set_dp:Nn \l__draw_tmp_box { Opt }
68     \box_set_wd:Nn \l__draw_tmp_box { Opt }
69     \box_use_drop:N \l__draw_tmp_box

```

```

70     \group_end:
71   }

```

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

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

```

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

```

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

```

94 </package>

```

3 I3draw-layers implementation

```

95 <*package>

```

```

96 <@@=draw>

```

3.1 User interface

```

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

```

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

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

```

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

```

(End of definition for `\l__draw_layer_tl`.)

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

```

108 \bool_new:N \l__draw_layer_close_bool

```

(End of definition for `\l__draw_layer_close_bool`.)

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

```

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

```

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

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

```

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

```

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

3.2 Internal cross-links

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

```

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

```

(End of definition for __draw_layers_insert:.)

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

```

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

```

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

```

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

```

```

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

```

4 l3draw-paths implementation

```

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

```

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

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

```

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

```

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

4.1 Tracking paths

```

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

```

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

4.2 Corner arcs

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

`\l__draw_corner_xarc_dim` The two arcs in use.

```

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

```

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

\l__draw_corner_arc_bool A flag to speed up the repeated checks.
242 \bool_new:N \l__draw_corner_arc_bool

(End of definition for \l__draw_corner_arc_bool.)

\draw_path_corner_arc:nn Calculate the arcs, check they are non-zero.
243 \cs_new_protected:Npn \draw_path_corner_arc:nn #1#2
244 {
245 \dim_set:Nn \l__draw_corner_xarc_dim { \fp_to_dim:n {#1} }
246 \dim_set:Nn \l__draw_corner_yarc_dim { \fp_to_dim:n {#2} }
247 \bool_lazy_and:nnTF
248 { \dim_compare_p:nNn \l__draw_corner_xarc_dim = { Opt } }
249 { \dim_compare_p:nNn \l__draw_corner_yarc_dim = { Opt } }
250 { \bool_set_false:N \l__draw_corner_arc_bool }
251 { \bool_set_true:N \l__draw_corner_arc_bool }
252 }

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

__draw_path_mark_corner: Mark up corners for arc post-processing.
253 \cs_new_protected:Npn __draw_path_mark_corner:
254 {
255 \bool_if:NT \l__draw_corner_arc_bool
256 {
257 __draw_softpath_roundpoint:VV
258 \l__draw_corner_xarc_dim
259 \l__draw_corner_yarc_dim
260 }
261 }

(End of definition for __draw_path_mark_corner:.)

4.3 Basic path constructions

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

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

```

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

```

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

`\draw_path_close:` A simple wrapper.

```

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

```

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

4.4 Canvas path constructions

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

```

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

```

```

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

```

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

4.5 Computed curves

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

```

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

```

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

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

and

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

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

```

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

```

```

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

```

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

```

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

```

```

407         #3 {#4} {#5}
408     }
409 }
410 \__draw_path_mark_corner:
411 \__draw_path_arc_auxi:enenNnn
412 { \fp_to_decimal:N \l__draw_path_arc_start_fp }
413 {#2}
414 { \fp_eval:n { abs( \l__draw_path_arc_start_fp - #2 ) } }
415 {#2}
416 #3 {#4} {#5}
417 }

```

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.

```

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

```

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

```

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

```

```

450     {
451       \__draw_point_transform_noshift:n
452       { \draw_point_polar:nnn {#7} {#8} { #1 #4 90 } }
453     }
454   \__draw_point_process:nnn
455   { \__draw_path_arc_auxiv:nnnn }
456   {
457     \draw_point_transform:n
458     { \draw_point_polar:nnn {#5} {#6} {#1} }
459   }
460   {
461     \draw_point_transform:n
462     { \draw_point_polar:nnn {#5} {#6} {#2} }
463   }
464   \__draw_point_process:nn
465   { \__draw_path_arc_auxv:nn }
466   {
467     \__draw_point_transform_noshift:n
468     { \draw_point_polar:nnn {#7} {#8} { #2 #4 -90 } }
469   }
470   \exp_after:wN \__draw_path_curveto:nnnnnn \l__draw_path_tmp_tl
471   \fp_set:Nn \l__draw_path_arc_delta_fp { abs ( #2 - #3 ) }
472   \fp_set:Nn \l__draw_path_arc_start_fp {#2}
473 }

```

The first control point.

```

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

```

The end point: simple arithmetic.

```

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

```

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

```

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

```

```

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

```

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

```

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

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

```

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

```

```

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

```

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

```

```

585     {
586     \__draw_point_process:nnn \__draw_path_rectangle_rounded:nnnn
587     { \draw_point:n {#1} }
588     { \draw_point:n {#2} }
589     }
590     {
591     \__draw_point_process:nnn \__draw_path_rectangle:nnnn
592     { \draw_point:n { (#1) + ( \l__draw_xshift_dim , \l__draw_yshift_dim ) } }
593     { \draw_point:n {#2} }
594     }
595     }
596 \cs_new_protected:Npn \__draw_path_rectangle:nnnn #1#2#3#4
597 {
598   \__draw_path_update_limits:nn {#1} {#2}
599   \__draw_path_update_limits:nn { #1 + #3 } { #2 + #4 }
600   \__draw_softpath_rectangle:nnnn {#1} {#2} {#3} {#4}
601   \__draw_path_update_last:nn {#1} {#2}
602 }
603 \cs_new_protected:Npn \__draw_path_rectangle_rounded:nnnn #1#2#3#4
604 {
605   \draw_path_moveto:n { #1 + #3 , #2 + #4 }
606   \draw_path_lineto:n { #1 , #2 + #4 }
607   \draw_path_lineto:n { #1 , #2 }
608   \draw_path_lineto:n { #1 + #3 , #2 }
609   \draw_path_close:
610   \draw_path_moveto:n { #1 , #2 }
611 }

```

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

`\draw_path_rectangle_corners:nn`
`__draw_path_rectangle_corners:nnnn`

Another shortcut wrapper.

```

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

```

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

4.7 Grids

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

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

```

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

```

```

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

```

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

4.8 Using paths

```

\l__draw_path_use_clip_bool Actions to pass to the driver.
\l__draw_path_use_fill_bool 672 \bool_new:N \l__draw_path_use_clip_bool
\l__draw_path_use_stroke_bool 673 \bool_new:N \l__draw_path_use_fill_bool

```

```
674 \bool_new:N \l__draw_path_use_stroke_bool
```

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

`\l__draw_path_use_clear_bool` Actions handled at the macro layer.

```
675 \bool_new:N \l__draw_path_use_clear_bool
```

(End of definition for `\l__draw_path_use_clear_bool`.)

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

```

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

```

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).

```

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

```

```

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

```

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.

```

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

```

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

4.9 Scoping paths

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

```

\l__draw_path_lastx_dim 787 \dim_new:N \l__draw_path_lastx_dim
\l__draw_path_lasty_dim 788 \dim_new:N \l__draw_path_lasty_dim
\l__draw_path_xmax_dim 789 \dim_new:N \l__draw_path_xmax_dim
\l__draw_path_xmin_dim 790 \dim_new:N \l__draw_path_xmin_dim
\l__draw_path_ymax_dim 791 \dim_new:N \l__draw_path_ymax_dim
\l__draw_path_ymin_dim 792 \dim_new:N \l__draw_path_ymin_dim
\l__draw_softpath_corners_bool 793 \dim_new:N \l__draw_softpath_lastx_dim
794 \dim_new:N \l__draw_softpath_lasty_dim
795 \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:
796 \cs_new_protected:Npn \draw_path_scope_begin:
797 {
798   \group_begin:
799   \dim_set_eq:NN \l__draw_path_lastx_dim \g__draw_path_lastx_dim
800   \dim_set_eq:NN \l__draw_path_lasty_dim \g__draw_path_lasty_dim
801   \dim_set_eq:NN \l__draw_path_xmax_dim \g__draw_path_xmax_dim
802   \dim_set_eq:NN \l__draw_path_xmin_dim \g__draw_path_xmin_dim
803   \dim_set_eq:NN \l__draw_path_ymax_dim \g__draw_path_ymax_dim
804   \dim_set_eq:NN \l__draw_path_ymin_dim \g__draw_path_ymin_dim
805   \dim_set_eq:NN \l__draw_softpath_lastx_dim \g__draw_softpath_lastx_dim

```

```

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

```

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

4.10 Messages

```

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

```

5 I3draw-points implementation

```

830 <*package>
831 <@@=draw>

```

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

- `\pgfpointorigin`: Can be given explicitly as `0pt,0pt`.
- `\pgfpointadd`, `\pgfpointdiff`, `\pgfpointscale`: Can be given explicitly.
- `\pgfextractx`, `\pgfextracty`: Available by applying `\use_i:nn/\use_ii:nn` or similar to the `e`-type expansion of a point expression.
- `\pgfgetlastxy`: Unused in the entire `pgf` core, may be emulated by `e`-type expansion of a point expression, then using the result.

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

- `\pgfpointcylindrical`, `\pgfpointspherical`: The usefulness of these commands is not currently clear.

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

5.1 Support functions

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

```

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

```

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

5.2 Basic points

`\draw_point:n` Coordinates are always returned as two dimensions.

```

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

```

```

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

```

5.3 Polar coordinates

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

```

\draw_point_polar:nn
\draw_point_polar:nnn
__draw_draw_polar:nnn
__draw_draw_polar:enn
871 \cs_new:Npn \draw_point_polar:nn #1#2
872   { \draw_point_polar:nnn {#1} {#1} {#2} }
873 \cs_new:Npn \draw_point_polar:nnn #1#2#3
874   { \__draw_draw_polar:enn { \fp_eval:n {#3} } {#1} {#2} }
875 \cs_new:Npn \__draw_draw_polar:nnn #1#2#3
876   { \draw_point:n { cosd(#1) * (#2) , sind(#1) * (#3) } }
877 \cs_generate_variant:Nn \__draw_draw_polar:nnn { e }

```

5.4 Point expression arithmetic

These functions all take point expressions as arguments.

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

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

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

```

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

```

5.5 Intersection calculations

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

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

and

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

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

```

896 \cs_new:Npn \draw_point_intersect_lines:nmnn #1#2#3#4
897 {
898   \__draw_point_process:nmnnn
899   { \__draw_point_intersect_lines:nmnnnnnn }
900   { \draw_point:n {#1} }
901   { \draw_point:n {#2} }
902   { \draw_point:n {#3} }
903   { \draw_point:n {#4} }
904 }

```

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.

```

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

```

```

918     {
919         ( #2 * #3 - #1 * #4 , #2 * #5 - #1 * #6 )
920         / ( #4 * #5 - #6 * #3 )
921     }
922 }
923 \cs_generate_variant:Nn \__draw_point_intersect_lines_aux:nnnnnn { eeeee }

```

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

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

in either

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

or

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

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

```

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

```

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

```

#1 r
#2 s
#3 a
#4 b

```

#5 c

#6 d

#7 n

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

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

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

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

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

#1 k

#2 $1/p$

#3 r^2

#4 e

#5 f

#6 a

#7 b

#8 n

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

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

The intersection points P_1 and P_2 between a line joining points (x_1, y_1) and (x_2, y_2) and a circle with center (x_3, y_3) and radius r . We use the intermediate values

```
\draw_point_intersect_line_circle:nnnnn
\draw_point_intersect_line_circle_auxi:nnnnnnnn
\draw_point_intersect_line_circle_auxii:nnnnnnnn
\draw_point_intersect_line_circle_auxiii:ennnnnnnn
\draw_point_intersect_line_circle_auxiiii:nnnnnnnn
\draw_point_intersect_line_circle_auxv:eeennnnnn
\draw_point_intersect_line_circle_auxvi:nnnnnnnn
\draw_point_intersect_line_circle_auxvii:eeennnnnn
\draw_point_intersect_line_circle_auxviii:nnnnn
\draw_point_intersect_line_circle_auxviiii:ennnn
```

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

in either

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

or

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

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

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

```

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

```

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

```

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

```

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

```

1003 \cs_new:Npn \__draw_point_intersect_line_circle_auxiii:nnnnnnnn #1#2#3#4#5#6#7#8
1004 {
1005   \__draw_point_intersect_line_circle_auxiv:eeennnnnn
1006   { \fp_eval:n { #2 * #2 - 4 * #1 * #3 } }
1007   { \int_if_odd:nTF {#8} { 1 } { -1 } }
1008   {#1} {#2} {#4} {#5} {#6} {#7}
1009 }
1010 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxiii:nnnnnnnn { eee }

```

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

```

#1 a
#2 b
#3 c

```

#4 d
#5 \pm (the usage of n)
#6 x_1
#7 y_1
#8 x_2
#9 y_2

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

```

1011 \cs_new:Npn \__draw_point_intersect_line_circle_auxv:nnnnnnn #1#2#3#4#5#6#7#8
1012 {
1013   \__draw_point_intersect_line_circle_auxv:ennnn
1014   { \fp_eval:n { (-1 * #4 + #2 * sqrt(#1)) / (2 * #3) } }
1015   {#5} {#6} {#7} {#8}
1016 }
1017 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxv:nnnnnnn { ee }
1018 \cs_new:Npn \__draw_point_intersect_line_circle_auxv:nnnnn #1#2#3#4#5
1019 {
1020   \draw_point:n
1021   { #2 + #1 * (#4 - #2), #3 + #1 * (#5 - #3) }
1022 }
1023 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxv:nnnnn { e }

```

5.6 Interpolation on a line (vector) or arc

Simple maths after expansion.

```

\draw_point_interpolate_line:nnn 1024 \cs_new:Npn \draw_point_interpolate_line:nnn #1#2#3
\_draw_point_interpolate_line_aux:nnnnn 1025 {
\_draw_point_interpolate_line_aux:ennnn 1026   \__draw_point_process:nnn
\_draw_point_interpolate_line_aux:nnnnnn 1027   { \__draw_point_interpolate_line_aux:ennnn { \fp_eval:n {#1} } }
\_draw_point_interpolate_line_aux:ennnnn 1028   { \draw_point:n {#2} }
1029   { \draw_point:n {#3} }
1030 }
1031 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnn #1#2#3#4#5
1032 {
1033   \__draw_point_interpolate_line_aux:ennnnn { \fp_eval:n { 1 - #1 } }
1034   {#1} {#2} {#3} {#4} {#5}
1035 }
1036 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnn { e }
1037 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnnn #1#2#3#4#5#6
1038 { \draw_point:n { #2 * #3 + #1 * #5 , #2 * #4 + #1 * #6 } }
1039 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnnn { e }

```

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

```

\draw_point_interpolate_distance:nnn 1040 \cs_new:Npn \draw_point_interpolate_distance:nnn #1#2#3
\_draw_point_interpolate_distance:nnnnn 1041 {
\_draw_point_interpolate_distance:nnnnnn 1042   \__draw_point_process:nn
\_draw_point_interpolate_distance:ennnnn 1043   { \__draw_point_interpolate_distance:nnnn {#1} {#3} }
1044   {#2}

```

```

1045 }
1046 \cs_new:Npn \__draw_point_interpolate_distance:nnnn #1#2#3#4
1047 {
1048   \__draw_point_process:nn
1049   {
1050     \__draw_point_interpolate_distance:ennnn
1051     { \fp_eval:n {#1} } {#3} {#4}
1052   }
1053   { \draw_point_unit_vector:n { ( #2 ) - ( #3 , #4 ) } }
1054 }
1055 \cs_new:Npn \__draw_point_interpolate_distance:nnnnn #1#2#3#4#5
1056 { \draw_point:n { #2 + #1 * #4 , #3 + #1 * #5 } }
1057 \cs_generate_variant:Nn \__draw_point_interpolate_distance:nnnnn { e }

```

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

```

\draw_point_interpolate_arcaxes:nnnnnn
\draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn
\draw_point_interpolate_arcaxes_auxii:nnnnnnnnnn
\draw_point_interpolate_arcaxes_auxiii:nnnnnnnnnn
\draw_point_interpolate_arcaxes_auxiiii:nnnnnnnnnn
\draw_point_interpolate_arcaxes_auxiv:nnnnnnnnnn
\draw_point_interpolate_arcaxes_auxiv:eennnnnnnn

```

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

```

1058 \cs_new:Npn \draw_point_interpolate_arcaxes:nnnnnn #1#2#3#4#5#6
1059 {
1060   \__draw_point_process:nnnn
1061   { \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn {#1} {#5} {#6} }
1062   { \draw_point:n {#2} }
1063   { \draw_point:n {#3} }
1064   { \draw_point:n {#4} }
1065 }
1066 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
1067 {
1068   \__draw_point_interpolate_arcaxes_auxii:ennnnnnnnnn
1069   { \fp_eval:n {#1} } {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1070 }

```

At this stage, the three coordinate 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.

```

1071 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
1072 {

```

```

1073   \_draw_point_interpolate_arcaxes_auxiii:ennnnnn
1074   { \fp_eval:n { #1 * (#3) + ( 1 - #1 ) * (#2) } }
1075   {#4} {#5} {#6} {#7} {#8} {#9}
1076 }
1077 \cs_generate_variant:Nn \_draw_point_interpolate_arcaxes_auxii:nnnnnnnnn { e }
1078 \cs_new:Npn \_draw_point_interpolate_arcaxes_auxiii:nnnnnnn #1#2#3#4#5#6#7
1079 {
1080   \_draw_point_interpolate_arcaxes_auxiv:ennnnnn
1081   { \fp_eval:n { cosd (#1) } }
1082   { \fp_eval:n { sind (#1) } }
1083   {#2} {#3} {#4} {#5} {#6} {#7}
1084 }
1085 \cs_generate_variant:Nn \_draw_point_interpolate_arcaxes_auxiii:nnnnnnn { e }
1086 \cs_new:Npn \_draw_point_interpolate_arcaxes_auxiv:nnnnnnn #1#2#3#4#5#6#7#8
1087 {
1088   \draw_point:n
1089   { #3 + #1 * #5 + #2 * #7 , #4 + #1 * #6 + #2 * #8 }
1090 }
1091 \cs_generate_variant:Nn \_draw_point_interpolate_arcaxes_auxiv:nnnnnnn { ee }

```

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

```

\_draw_point_interpolate_curve:nnnnn
draw_point_interpolate_curve_auxi:nnnnnnnnn
draw_point_interpolate_curve_auxii:nnnnnnnnn
draw_point_interpolate_curve_auxiii:ennnnnnnn
\_draw_point_interpolate_curve_auxiiii:nnnnnnn
\_draw_point_interpolate_curve_auxiii:ennnnnn
\_draw_point_interpolate_curve_auxiv:nnnnnnn
\_draw_point_interpolate_curve_auxv:nnv
\_draw_point_interpolate_curve_auxv:eev
\_draw_point_interpolate_curve_auxvi:n
draw_point_interpolate_curve_auxvii:nnnnnnnnn
draw_point_interpolate_curve_auxviii:nnnnnnn
draw_point_interpolate_curve_auxviii:ennnnnn

```

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.

```

1092 \cs_new:Npn \draw_point_interpolate_curve:nnnnnn #1#2#3#4#5
1093 {
1094   \_draw_point_process:nnnnn
1095   { \_draw_point_interpolate_curve_auxi:nnnnnnnnn {#1} }
1096   { \draw_point:n {#2} }
1097   { \draw_point:n {#3} }
1098   { \draw_point:n {#4} }
1099   { \draw_point:n {#5} }
1100 }
1101 \cs_new:Npn \_draw_point_interpolate_curve_auxi:nnnnnnnnn #1#2#3#4#5#6#7#8#9
1102 {
1103   \_draw_point_interpolate_curve_auxii:ennnnnnnn
1104   { \fp_eval:n {#1} }
1105   {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1106 }

```

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

need all of the input coordinates

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

In the second stage, we can drop the final point

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

and for the final stage only need one set of calculations

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

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

```

1107 \cs_new:Npn \__draw_point_interpolate_curve_auxii:nnnnnnnn
1108   #1#2#3#4#5#6#7#8#9
1109   {
1110     \__draw_point_interpolate_curve_auxiii:ennnnn
1111     { \fp_eval:n { 1 - #1 } }
1112     {#1}
1113     { {#2} {#3} } { {#4} {#5} } { {#6} {#7} } { {#8} {#9} }
1114   }
1115 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxii:nnnnnnnn { e }
1116 % \begin{macrocode}
1117 % We need to do the first cycle, but haven't got enough arguments to keep
1118 % everything in play at once. So here we use a bit of argument re-ordering
1119 % and a single auxiliary to get the job done.
1120 % \begin{macrocode}
1121 \cs_new:Npn \__draw_point_interpolate_curve_auxiii:nnnnnn #1#2#3#4#5#6
1122   {
1123     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #3 #4
1124     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #4 #5
1125     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #5 #6
1126     \prg_do_nothing:
1127     \__draw_point_interpolate_curve_auxvi:n { {#1} {#2} }
1128   }
1129 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxiii:nnnnnn { e }
1130 \cs_new:Npn \__draw_point_interpolate_curve_auxiv:nnnnnn #1#2#3#4#5#6
1131   {
1132     \__draw_point_interpolate_curve_auxv:eew
1133     { \fp_eval:n { #1 * #3 + #2 * #5 } }
1134     { \fp_eval:n { #1 * #4 + #2 * #6 } }
1135   }

```

```

1136 \cs_new:Npn \__draw_point_interpolate_curve_auxv:nnw
1137   #1#2#3 \prg_do_nothing: #4#5
1138   {
1139     #3
1140     \prg_do_nothing:
1141     #4 { #5 {#1} {#2} }
1142   }
1143 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxv:nnw { ee }
1144 % \begin{macrocode}
1145 % Get the arguments back into the right places and to the second and
1146 % third cycles directly.
1147 % \begin{macrocode}
1148 \cs_new:Npn \__draw_point_interpolate_curve_auxvi:n #1
1149   { \__draw_point_interpolate_curve_auxvii:nnnnnnn #1 }
1150 \cs_new:Npn \__draw_point_interpolate_curve_auxvii:nnnnnnn #1#2#3#4#5#6#7#8
1151   {
1152     \__draw_point_interpolate_curve_auxviii:eeeenn
1153     { \fp_eval:n { #1 * #5 + #2 * #3 } }
1154     { \fp_eval:n { #1 * #6 + #2 * #4 } }
1155     { \fp_eval:n { #1 * #7 + #2 * #5 } }
1156     { \fp_eval:n { #1 * #8 + #2 * #6 } }
1157     {#1} {#2}
1158   }
1159 \cs_new:Npn \__draw_point_interpolate_curve_auxviii:nnnnnn #1#2#3#4#5#6
1160   {
1161     \draw_point:n
1162     { #5 * #3 + #6 * #1 , #5 * #4 + #6 * #2 }
1163   }
1164 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxviii:nnnnnn { eeee }

```

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

5.7 Vector support

As well as coordinates relative to the drawing

```

\l__draw_xvec_x_dim Base vectors to map to the underlying two-dimensional drawing space.
\l__draw_xvec_y_dim
\l__draw_yvec_x_dim
\l__draw_yvec_y_dim
\l__draw_zvec_x_dim
\l__draw_zvec_y_dim
1165 \dim_new:N \l__draw_xvec_x_dim
1166 \dim_new:N \l__draw_xvec_y_dim
1167 \dim_new:N \l__draw_yvec_x_dim
1168 \dim_new:N \l__draw_yvec_y_dim
1169 \dim_new:N \l__draw_zvec_x_dim
1170 \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
\draw_zvec:n
1171 \cs_new_protected:Npn \draw_xvec:n #1
1172   { \__draw_vec:nn { x } {#1} }
1173 \cs_new_protected:Npn \draw_yvec:n #1
1174   { \__draw_vec:nn { y } {#1} }
1175 \cs_new_protected:Npn \draw_zvec:n #1
1176   { \__draw_vec:nn { z } {#1} }

```

```

1177 \cs_new_protected:Npn \__draw_vec:nn #1#2
1178 { \__draw_point_process:nn { \__draw_vec:nnn {#1} } { \draw_point:n {#2} } }
1179 \cs_new_protected:Npn \__draw_vec:nnn #1#2#3
1180 {
1181   \dim_set:cn { l__draw_ #1 vec_x_dim } {#2}
1182   \dim_set:cn { l__draw_ #1 vec_y_dim } {#3}
1183 }

```

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

Initialize the vectors.

```

1184 \draw_xvec:n { 1cm , 0cm }
1185 \draw_yvec:n { 0cm , 1cm }
1186 \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 1187 \cs_new:Npn \draw_point_vec:nn #1#2
\__draw_point_vec:ee 1188 { \__draw_point_vec:ee { \fp_eval:n {#1} } { \fp_eval:n {#2} } }
\draw_point_vec:nnn 1189 \cs_new:Npn \__draw_point_vec:nn #1#2
\__draw_point_vec:nnn 1190 {
\__draw_point_vec:eee 1191   \draw_point:n
1192   {
1193     #1 * \l__draw_xvec_x_dim + #2 * \l__draw_yvec_x_dim ,
1194     #1 * \l__draw_xvec_y_dim + #2 * \l__draw_yvec_y_dim
1195   }
1196 }
1197 \cs_generate_variant:Nn \__draw_point_vec:nn { ee }
1198 \cs_new:Npn \draw_point_vec:nnn #1#2#3
1199 {
1200   \__draw_point_vec:eee
1201   { \fp_eval:n {#1} } { \fp_eval:n {#2} } { \fp_eval:n {#3} }
1202 }
1203 \cs_new:Npn \__draw_point_vec:nnn #1#2#3
1204 {
1205   \draw_point:n
1206   {
1207     #1 * \l__draw_xvec_x_dim
1208     + #2 * \l__draw_yvec_x_dim
1209     + #3 * \l__draw_zvec_x_dim
1210     ,
1211     #1 * \l__draw_xvec_y_dim
1212     + #2 * \l__draw_yvec_y_dim
1213     + #3 * \l__draw_zvec_y_dim
1214   }
1215 }
1216 \cs_generate_variant:Nn \__draw_point_vec:nnn { eee }

```

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

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

```

\draw_point_vec_polar:nnn 1217 \cs_new:Npn \draw_point_vec_polar:nn #1#2
\__draw_point_vec_polar:nnn 1218 { \draw_point_vec_polar:nnn {#1} {#1} {#2} }
\__draw_point_vec_polar:enn 1219 \cs_new:Npn \draw_point_vec_polar:nnn #1#2#3
1220 { \__draw_draw_vec_polar:enn { \fp_eval:n {#3} } {#1} {#2} }
1221 \cs_new:Npn \__draw_draw_vec_polar:nnn #1#2#3

```

```

1222 {
1223   \draw_point:n
1224   {
1225     cosd(#1) * (#2) * \l__draw_xvec_x_dim ,
1226     sind(#1) * (#3) * \l__draw_yvec_y_dim
1227   }
1228 }
1229 \cs_generate_variant:Nn \__draw_draw_vec_polar:nnn { e }

```

(End of definition for `\draw_point_vec_polar:n`, `\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.

```

1230 \cs_new:Npn \draw_point_transform:n #1
1231 {
1232   \__draw_point_process:nn
1233   { \__draw_point_transform:nn } { \draw_point:n {#1} }
1234 }
1235 \cs_new:Npn \__draw_point_transform:nn #1#2
1236 {
1237   \bool_if:NTF \l__draw_matrix_active_bool
1238   {
1239     \draw_point:n
1240     {
1241       (
1242         \l__draw_matrix_a_fp * #1
1243         + \l__draw_matrix_c_fp * #2
1244         + \l__draw_xshift_dim
1245       )
1246       ,
1247       (
1248         \l__draw_matrix_b_fp * #1
1249         + \l__draw_matrix_d_fp * #2
1250         + \l__draw_yshift_dim
1251       )
1252     }
1253   }
1254   {
1255     \draw_point:n
1256     {
1257       (#1, #2)
1258       + ( \l__draw_xshift_dim , \l__draw_yshift_dim )
1259     }
1260   }
1261 }

```

(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 1262 \cs_new:Npn \__draw_point_transform_noshift:n #1

```

```

1263 {
1264   \__draw_point_process:nn
1265   { \__draw_point_transform_noshift:nn }
1266   { \draw_point:n {#1} }
1267 }
1268 \cs_new:Npn \__draw_point_transform_noshift:nn #1#2
1269 {
1270   \bool_if:NTF \l__draw_matrix_active_bool
1271   {
1272     \draw_point:n
1273     {
1274       (
1275         \l__draw_matrix_a_fp * #1
1276         + \l__draw_matrix_c_fp * #2
1277       )
1278       ,
1279       (
1280         \l__draw_matrix_b_fp * #1
1281         + \l__draw_matrix_d_fp * #2
1282       )
1283     }
1284   }
1285   { \draw_point:n { (#1, #2) } }
1286 }

```

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

```
1287 \endpackage
```

6 I3draw-scopes implementation

```
1288 \beginpackage
```

```
1289 \begin@draw
```

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

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

6.1 Drawing environment

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

```

\g_draw_bb_xmin_dim
\g_draw_bb_ymax_dim
\g_draw_bb_ymin_dim
1290 \dim_new:N \g_draw_bb_xmax_dim
1291 \dim_new:N \g_draw_bb_xmin_dim
1292 \dim_new:N \g_draw_bb_ymax_dim
1293 \dim_new:N \g_draw_bb_ymin_dim

```

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

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

```
1294 \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.

```
1295 \box_new:N \l__draw_main_box
1296 \box_new:N \l__draw_layer_main_box
```

(End of definition for \l__draw_layer_main_box.)

`\g_draw_id_int` The drawing number.

```
1297 \int_new:N \g_draw_id_int
```

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

`__draw_reset_bb:` A simple auxiliary.

```
1298 \cs_new_protected:Npn \__draw_reset_bb:
1299 {
1300   \dim_gset:Nn \g_draw_bb_xmax_dim { -\c_max_dim }
1301   \dim_gset:Nn \g_draw_bb_xmin_dim { \c_max_dim }
1302   \dim_gset:Nn \g_draw_bb_ymax_dim { -\c_max_dim }
1303   \dim_gset:Nn \g_draw_bb_ymin_dim { \c_max_dim }
1304 }
```

(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

```
1305 \cs_new_protected:Npn \draw_begin:
1306 {
1307   \group_begin:
1308     \int_gincr:N \g_draw_id_int
1309     \hbox_set:Nw \l__draw_main_box
1310     \__draw_backend_begin:
1311     \__draw_reset_bb:
1312     \__draw_path_reset_limits:
1313     \bool_set_true:N \l_draw_bb_update_bool
1314     \draw_transform_matrix_reset:
1315     \draw_transform_shift_reset:
1316     \__draw_softpath_clear:
1317     \draw_set_linewidth:n { \l_draw_default_linewidth_dim }
1318     \color_ensure_current:
1319     \draw_set_nonzero_rule:
1320     \draw_set_cap_but:
1321     \draw_set_join_miter:
1322     \draw_set_miterlimit:n { 10 }
1323     \draw_set_dash_pattern:nn { } { 0cm }
1324     \hbox_set:Nw \l__draw_layer_main_box
1325     \__draw_record_origin:
1326   }
1327 \cs_new_protected:Npn \draw_end:
1328 {
1329   \__draw_baseline_finalize:w
1330   \exp_args:NNNV \hbox_set_end:
```

```

1331         \clist_set:Nn \l_draw_layers_clist \l_draw_layers_clist
1332         \__draw_layers_insert:
1333         \__draw_backend_end:
1334     \hbox_set_end:
1335     \dim_compare:nNnT \g_draw_bb_xmin_dim = \c_max_dim
1336     {
1337         \dim_gzero:N \g_draw_bb_xmax_dim
1338         \dim_gzero:N \g_draw_bb_xmin_dim
1339         \dim_gzero:N \g_draw_bb_ymax_dim
1340         \dim_gzero:N \g_draw_bb_ymin_dim
1341     }
1342     \__draw_finalize:
1343     \box_set_wd:Nn \l__draw_main_box
1344     { \g_draw_bb_xmax_dim - \g_draw_bb_xmin_dim }
1345     \mode_leave_vertical:
1346     \box_use_drop:N \l__draw_main_box
1347 \group_end:
1348 }

```

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

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

```

1349 \cs_new_protected:Npe \__draw_record_origin:
1350 {
1351     \hbox_to_wd:nn { Opt }
1352     {
1353         \tex_savepos:D
1354         \cs_if_exist:NT \@expl@finalise@setup@@
1355         {
1356             \exp_not:N \property_record:en
1357             { draw . \exp_not:N \int_use:N \exp_not:N \g_draw_id_int }
1358             { xpos , ypos , abspage }
1359         }
1360         \tex_savepos:D
1361     }
1362 }
1363 \cs_generate_variant:Nn \property_record:nn { e }

```

(End of definition for `__draw_record_origin:`.)

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

```

1364 \cs_new_protected:Npn \__draw_finalize:
1365 {
1366     \hbox_set:Nn \l__draw_main_box
1367     {
1368         \skip_horizontal:n { -\g_draw_bb_xmin_dim }
1369         \box_move_down:nn
1370         { \g_draw_bb_ymin_dim }

```

```

1371         { \box_use_drop:N \l__draw_main_box }
1372     }
1373     \box_set_dp:Nn \l__draw_main_box { Opt }
1374     \box_set_ht:Nn \l__draw_main_box
1375     { \g_draw_bb_ymax_dim - \g_draw_bb_ymin_dim }
1376 }
1377 \cs_new_protected:Npn \__draw_finalize_baseline:n #1
1378 {
1379     \hbox_set:Nn \l__draw_main_box
1380     {
1381         \skip_horizontal:n { -\g_draw_bb_xmin_dim }
1382         \box_move_down:nn
1383         {#1}
1384         { \box_use_drop:N \l__draw_main_box }
1385     }
1386     \box_set_dp:Nn \l__draw_main_box
1387     {
1388         \dim_max:nn
1389         { #1 - \g_draw_bb_ymin_dim }
1390         { Opt }
1391     }
1392     \box_set_ht:Nn \l__draw_main_box
1393     { \g_draw_bb_ymax_dim - #1 }
1394 }

```

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

6.2 Baseline position

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

```

\l__draw_baseline_dim 1395 \bool_new:N \l__draw_baseline_bool
1396 \dim_new:N \l__draw_baseline_dim

```

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

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

```

1397 \cs_new_protected:Npn \draw_set_baseline:n #1
1398 {
1399     \bool_set_true:N \l__draw_baseline_bool
1400     \dim_set:Nn \l__draw_baseline_dim { \fp_to_dim:n {#1} }
1401 }

```

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

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

```

1402 \cs_new_protected:Npn \__draw_baseline_finalize:w #1 \__draw_finalize:
1403 {
1404     \bool_if:NTF \l__draw_baseline_bool
1405     {
1406         \use:e
1407         {
1408             \exp_not:n {#1}
1409             \__draw_finalize_baseline:n { \dim_use:N \l__draw_baseline_dim }

```

```

1410     }
1411   }
1412   { #1 \_draw_finalize: }
1413 }

```

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

6.3 Scopes

```

\l__draw_linewidth_dim Storage for local variables.
\l__draw_fill_color_tl 1414 \dim_new:N \l__draw_linewidth_dim
\l__draw_stroke_color_tl 1415 \tl_new:N \l__draw_fill_color_tl
1416 \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: 1417 \cs_new_protected:Npn \draw_scope_begin:
1418 {
1419   \_draw_backend_scope_begin:
1420   \group_begin:
1421   \dim_set_eq:NN \l__draw_linewidth_dim \g__draw_linewidth_dim
1422   \draw_path_scope_begin:
1423 }
1424 \cs_new_protected:Npn \draw_scope_end:
1425 {
1426   \draw_path_scope_end:
1427   \dim_gset_eq:NN \g__draw_linewidth_dim \l__draw_linewidth_dim
1428   \group_end:
1429   \_draw_backend_scope_end:
1430 }

```

(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 1431 \dim_new:N \l__draw_xmax_dim
\l__draw_ymax_dim 1432 \dim_new:N \l__draw_xmin_dim
\l__draw_ymin_dim 1433 \dim_new:N \l__draw_ymax_dim
1434 \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: 1435 \cs_new_protected:Npn \_draw_scope_bb_begin:
1436 {
1437   \group_begin:
1438   \dim_set_eq:NN \l__draw_xmax_dim \g_draw_bb_xmax_dim
1439   \dim_set_eq:NN \l__draw_xmin_dim \g_draw_bb_xmin_dim
1440   \dim_set_eq:NN \l__draw_ymax_dim \g_draw_bb_ymax_dim
1441   \dim_set_eq:NN \l__draw_ymin_dim \g_draw_bb_ymin_dim
1442   \_draw_reset_bb:
1443 }
1444 \cs_new_protected:Npn \_draw_scope_bb_end:
1445 {

```

```

1446     \dim_gset_eq:NN \g_draw_bb_xmax_dim \l__draw_xmax_dim
1447     \dim_gset_eq:NN \g_draw_bb_xmin_dim \l__draw_xmin_dim
1448     \dim_gset_eq:NN \g_draw_bb_ymax_dim \l__draw_ymax_dim
1449     \dim_gset_eq:NN \g_draw_bb_ymin_dim \l__draw_ymin_dim
1450   \group_end:
1451 }

```

(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:
1452 \cs_new_protected:Npn \draw_suspend_begin:
1453 {
1454   \__draw_scope_bb_begin:
1455   \draw_path_scope_begin:
1456   \draw_transform_matrix_reset:
1457   \draw_transform_shift_reset:
1458   \__draw_layers_save:
1459 }
1460 \cs_new_protected:Npn \draw_suspend_end:
1461 {
1462   \__draw_layers_restore:
1463   \draw_path_scope_end:
1464   \__draw_scope_bb_end:
1465 }

```

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

```

1466 </package>

```

7 l3draw-softpath implementation

```

1467 <*package>
1468 <@@=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.

```

1469 \tl_new:N \g__draw_softpath_main_tl

```

(End of definition for \g__draw_softpath_main_tl.)

```

\l__draw_softpath_tmp_tl Scratch space.
1470 \tl_new:N \l__draw_softpath_tmp_tl
(End of definition for \l__draw_softpath_tmp_tl.)

\g__draw_softpath_corners_bool Allow for optimized path use.
1471 \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 1472 \cs_new_protected:Npn \__draw_softpath_add:n
\__draw_softpath_add:e 1473 { \tl_build_gput_right:Nn \g__draw_softpath_main_tl }
1474 \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: 1475 \cs_new_protected:Npn \__draw_softpath_use:
1476 {
1477   \tl_build_get_intermediate:NN
1478   \g__draw_softpath_main_tl
1479   \l__draw_softpath_tmp_tl
1480   \l__draw_softpath_tmp_tl
1481 }
1482 \cs_new_protected:Npn \__draw_softpath_clear:
1483 {
1484   \tl_build_gbegin:N \g__draw_softpath_main_tl
1485   \bool_gset_false:N \g__draw_softpath_corners_bool
1486 }
(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: 1487 \cs_new_protected:Npn \__draw_softpath_save:
1488 {
1489   \tl_build_gend:N \g__draw_softpath_main_tl
1490   \tl_set_eq:NN
1491   \l__draw_softpath_main_tl
1492   \g__draw_softpath_main_tl
1493   \bool_set_eq:NN
1494   \l__draw_softpath_corners_bool
1495   \g__draw_softpath_corners_bool
1496   \__draw_softpath_clear:
1497 }
1498 \cs_new_protected:Npn \__draw_softpath_restore:
1499 {
1500   \__draw_softpath_clear:
1501   \__draw_softpath_add:o \l__draw_softpath_main_tl
1502   \bool_gset_eq:NN
1503   \g__draw_softpath_corners_bool
1504   \l__draw_softpath_corners_bool
1505 }
(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 1506 \dim_new:N \g__draw_softpath_lastx_dim
                             1507 \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.
                             1508 \bool_new:N \g__draw_softpath_move_bool
                             1509 \bool_gset_true:N \g__draw_softpath_move_bool

                             (End of definition for \g__draw_softpath_move_bool.)

\_draw_softpath_closepath: The various parts of a path expressed as the appropriate soft path functions.
  \_draw_softpath_curveto:nnmnn 1510 \cs_new_protected:Npn \_draw_softpath_closepath:
  \_draw_softpath_lineto:nn      1511 {
  \_draw_softpath_moveto:nn      1512   \_draw_softpath_add:e
  \_draw_softpath_rectangle:nnnn 1513   {
  \_draw_softpath_roundpoint:nn 1514     \_draw_softpath_close_op:nn
  \_draw_softpath_roundpoint:VV 1515     { \dim_use:N \g__draw_softpath_lastx_dim }
                             1516     { \dim_use:N \g__draw_softpath_lasty_dim }
                             1517   }
                             1518 }
  \cs_new_protected:Npn \_draw_softpath_curveto:nnnnnn #1#2#3#4#5#6
                             1519 {
                             1520   \_draw_softpath_add:n
                             1521   {
                             1522     \_draw_softpath_curveto_opi:nn {#1} {#2}
                             1523     \_draw_softpath_curveto_opii:nn {#3} {#4}
                             1524     \_draw_softpath_curveto_opiii:nn {#5} {#6}
                             1525   }
                             1526 }
                             1527 }
  \cs_new_protected:Npn \_draw_softpath_lineto:nn #1#2
                             1528 {
                             1529   \_draw_softpath_add:n
                             1530   { \_draw_softpath_lineto_op:nn {#1} {#2} }
                             1531 }
                             1532 }
  \cs_new_protected:Npn \_draw_softpath_moveto:nn #1#2
                             1533 {
                             1534   \_draw_softpath_add:n
                             1535   { \_draw_softpath_moveto_op:nn {#1} {#2} }
                             1536   \bool_if:NT \g__draw_softpath_move_bool
                             1537   {
                             1538     \dim_gset:Nn \g__draw_softpath_lastx_dim {#1}
                             1539     \dim_gset:Nn \g__draw_softpath_lasty_dim {#2}
                             1540   }
                             1541 }
                             1542 }
  \cs_new_protected:Npn \_draw_softpath_rectangle:nnnn #1#2#3#4
                             1543 {
                             1544   \_draw_softpath_add:n
                             1545   {
                             1546     \_draw_softpath_rectangle_opi:nn {#1} {#2}
                             1547     \_draw_softpath_rectangle_opii:nn {#3} {#4}
                             1548   }
                             1549 }
                             1550 }

```

```

1551 \cs_new_protected:Npn \__draw_softpath_roundpoint:nn #1#2
1552 {
1553   \__draw_softpath_add:n
1554   { \__draw_softpath_roundpoint_op:nn {#1} {#2} }
1555   \bool_gset_true:N \g__draw_softpath_corners_bool
1556 }
1557 \cs_generate_variant:Nn \__draw_softpath_roundpoint:nn { VV }

```

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

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

```

1558 \cs_new_protected:Npn \__draw_softpath_close_op:nn #1#2
1559 { \__draw_backend_closepath: }
1560 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nn #1#2
1561 { \__draw_softpath_curveto_opi:nnNnnNnn {#1} {#2} }
1562 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nnNnnNnn #1#2#3#4#5#6#7#8
1563 { \__draw_backend_curveto:nnnnnn {#1} {#2} {#4} {#5} {#7} {#8} }
1564 \cs_new_protected:Npn \__draw_softpath_curveto_opii:nn #1#2
1565 { \__draw_softpath_curveto_opii:nn }
1566 \cs_new_protected:Npn \__draw_softpath_curveto_opiii:nn #1#2
1567 { \__draw_softpath_curveto_opiii:nn }
1568 \cs_new_protected:Npn \__draw_softpath_lineto_op:nn #1#2
1569 { \__draw_backend_lineto:nn {#1} {#2} }
1570 \cs_new_protected:Npn \__draw_softpath_moveto_op:nn #1#2
1571 { \__draw_backend_moveto:nn {#1} {#2} }
1572 \cs_new_protected:Npn \__draw_softpath_roundpoint_op:nn #1#2
1573 { \__draw_softpath_roundpoint_op:nn }
1574 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nn #1#2
1575 { \__draw_softpath_rectangle_opi:nnNnn {#1} {#2} }
1576 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nnNnn #1#2#3#4#5
1577 { \__draw_backend_rectangle:nnnn {#1} {#2} {#4} {#5} }
1578 \cs_new_protected:Npn \__draw_softpath_rectangle_opii:nn #1#2
1579 { \__draw_softpath_rectangle_opii:nn }

```

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

7.2 Rounding soft path corners

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

`\l__draw_softpath_main_tl` For constructing the updated path.

```
1580 \tl_new:N \l__draw_softpath_main_tl
```

(End of definition for `\l__draw_softpath_main_tl`.)

`\l__draw_softpath_part_tl` Data structures.

```
1581 \tl_new:N \l__draw_softpath_part_tl
```

```
1582 \tl_new:N \l__draw_softpath_curve_end_tl
```

(End of definition for `\l__draw_softpath_part_tl`.)

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

```

1583 \fp_new:N \l__draw_softpath_lastx_fp
1584 \fp_new:N \l__draw_softpath_lasty_fp
1585 \dim_new:N \l__draw_softpath_corneri_dim
1586 \dim_new:N \l__draw_softpath_cornerii_dim
1587 \tl_new:N \l__draw_softpath_first_tl
1588 \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.

```

1589 \fp_const:Nn \c__draw_softpath_arc_fp { 4/3 * (sqrt(2) - 1) }

```

(End of definition for `\c__draw_softpath_arc_fp`.)

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

```

1590 \cs_new_protected:Npn \__draw_softpath_round_corners:
1591 {
1592   \bool_if:NT \g__draw_softpath_corners_bool
1593   {
1594     \group_begin:
1595     \tl_clear:N \l__draw_softpath_main_tl
1596     \tl_clear:N \l__draw_softpath_part_tl
1597     \fp_zero:N \l__draw_softpath_lastx_fp
1598     \fp_zero:N \l__draw_softpath_lasty_fp
1599     \tl_clear:N \l__draw_softpath_first_tl
1600     \tl_clear:N \l__draw_softpath_move_tl
1601     \tl_build_gend:N \g__draw_softpath_main_tl
1602     \exp_after:wN \__draw_softpath_round_loop:Nnn
1603     \g__draw_softpath_main_tl
1604     \q__draw_recursion_tail ? ?
1605     \q__draw_recursion_stop
1606   \group_end:
1607   }
1608   \bool_gset_false:N \g__draw_softpath_corners_bool
1609 }

```

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.

```

1610 \cs_new_protected:Npn \__draw_softpath_round_loop:Nnn #1#2#3
1611 {
1612   \__draw_if_recursion_tail_stop_do:Nn #1 { \__draw_softpath_round_end: }
1613   \token_if_eq_meaning:NNTF #1 \__draw_softpath_roundpoint_op:nn
1614   { \__draw_softpath_round_action:nn {#2} {#3} }
1615   {
1616     \tl_if_empty:NT \l__draw_softpath_first_tl
1617     { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1618     \fp_set:Nn \l__draw_softpath_lastx_fp {#2}
1619     \fp_set:Nn \l__draw_softpath_lasty_fp {#3}

```

```

1620     \token_if_eq_meaning:NNTF #1 \l__draw_softpath_moveto_op:nn
1621     {
1622         \tl_put_right:No \l__draw_softpath_main_tl
1623         \l__draw_softpath_move_tl
1624         \tl_put_right:No \l__draw_softpath_main_tl
1625         \l__draw_softpath_part_tl
1626         \tl_set:Nn \l__draw_softpath_move_tl { #1 {#2} {#3} }
1627         \tl_clear:N \l__draw_softpath_first_tl
1628         \tl_clear:N \l__draw_softpath_part_tl
1629     }
1630     { \tl_put_right:Nn \l__draw_softpath_part_tl { #1 {#2} {#3} } }
1631     \l__draw_softpath_round_loop:Nnn
1632 }
1633 }
1634 \cs_new_protected:Npn \l__draw_softpath_round_action:nn #1#2
1635 {
1636     \dim_set:Nn \l__draw_softpath_corneri_dim {#1}
1637     \dim_set:Nn \l__draw_softpath_cornerii_dim {#2}
1638     \bool_lazy_and:nnTF
1639     { \dim_compare_p:nNn \l__draw_softpath_corneri_dim = { Opt } }
1640     { \dim_compare_p:nNn \l__draw_softpath_cornerii_dim = { Opt } }
1641     { \l__draw_softpath_round_loop:Nnn }
1642     { \l__draw_softpath_round_action:Nnn }
1643 }

```

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.

```

1644 \cs_new_protected:Npn \l__draw_softpath_round_action:Nnn #1#2#3
1645 {
1646     \tl_if_empty:NT \l__draw_softpath_first_tl
1647     { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1648     \token_if_eq_meaning:NNTF #1 \l__draw_softpath_curveto_opi:nn
1649     { \l__draw_softpath_round_action_curveto:NnnNnn }
1650     {
1651         \token_if_eq_meaning:NNTF #1 \l__draw_softpath_close_op:nn
1652         { \l__draw_softpath_round_action_close: }
1653         {
1654             \token_if_eq_meaning:NNTF #1 \l__draw_softpath_lineto_op:nn
1655             { \l__draw_softpath_round_lookahead:NnnNnn }
1656             { \l__draw_softpath_round_loop:Nnn }
1657         }
1658     }
1659     #1 {#2} {#3}
1660 }

```

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.

```

1661 \cs_new_protected:Npn \l__draw_softpath_round_action_curveto:NnnNnn
1662 #1#2#3#4#5#6
1663 {
1664     \tl_put_right:Nn \l__draw_softpath_part_tl
1665     { #1 {#2} {#3} #4 {#5} {#6} }
1666     \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1667     \fp_set:Nn \l__draw_softpath_lasty_fp {#6}

```

```

1668     \__draw_softpath_round_lookahead:NnnNnn
1669   }
1670 \cs_new_protected:Npn \__draw_softpath_round_action_close:
1671 {
1672   \bool_lazy_and:nnTF
1673     { ! \tl_if_empty_p:N \l__draw_softpath_first_tl }
1674     { ! \tl_if_empty_p:N \l__draw_softpath_move_tl }
1675   {
1676     \exp_after:wN \__draw_softpath_round_close:nn
1677       \l__draw_softpath_first_tl
1678   }
1679   { \__draw_softpath_round_loop:Nnn }
1680 }

```

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.

```

1681 \cs_new_protected:Npn \__draw_softpath_round_lookahead:NnnNnn #1#2#3#4#5#6
1682 {
1683   \bool_lazy_any:nTF
1684     {
1685       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_lineto_op:nn }
1686       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_curveto_opi:nn }
1687       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_close_op:nn }
1688     }
1689     {
1690       \__draw_softpath_round_calc:NnnNnn
1691       \__draw_softpath_round_loop:Nnn
1692       {#5} {#6}
1693     }
1694     {
1695       \token_if_eq_meaning:NNTF #4 \__draw_softpath_roundpoint_op:nn
1696       { \__draw_softpath_round_roundpoint:NnnNnnNnn }
1697       { \__draw_softpath_round_loop:Nnn }
1698     }
1699     #1 {#2} {#3}
1700     #4 {#5} {#6}
1701 }
1702 \cs_new_protected:Npn \__draw_softpath_round_roundpoint:NnnNnnNnn
1703 #1#2#3#4#5#6#7#8#9
1704 {
1705   \__draw_softpath_round_calc:NnnNnn
1706   \__draw_softpath_round_loop:Nnn
1707   {#8} {#9}
1708   #1 {#2} {#3}
1709   #4 {#5} {#6} #7 {#8} {#9}
1710 }

```

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.

```

1711 \cs_new_protected:Npn \__draw_softpath_round_calc:NnnNnn #1#2#3#4#5#6
1712 {
1713   \tl_set:Ne \l__draw_softpath_curve_end_tl
1714   {
1715     \draw_point_interpolate_distance:nnn
1716     \l__draw_softpath_cornerii_dim
1717     { #5 , #6 } { #2 , #3 }
1718   }
1719   \tl_put_right:Ne \l__draw_softpath_part_tl
1720   {
1721     \exp_not:N #4
1722     \__draw_softpath_round_calc:eVnnnn
1723     {
1724       \draw_point_interpolate_distance:nnn
1725       \l__draw_softpath_corneri_dim
1726       { #5 , #6 }
1727       {
1728         \l__draw_softpath_lastx_fp ,
1729         \l__draw_softpath_lasty_fp
1730       }
1731     }
1732     \l__draw_softpath_curve_end_tl
1733     {#5} {#6} {#2} {#3}
1734   }
1735   \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1736   \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1737   #1
1738 }

```

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

```

1739 \cs_new:Npn \__draw_softpath_round_calc:nnnnnw #1#2#3#4#5#6
1740 {
1741   \__draw_softpath_round_calc:nnnnw {#3} {#4} {#5} {#6}
1742   #1 \s__draw_mark #2 \s__draw_stop
1743 }
1744 \cs_generate_variant:Nn \__draw_softpath_round_calc:nnnnnw { eV }

```

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

```

1745 \cs_new:Npn \__draw_softpath_round_calc:nnnnw
1746 #1#2#3#4 #5 , #6 \s__draw_mark #7 , #8 \s__draw_stop
1747 {
1748   {#5} {#6}
1749   \exp_not:N \__draw_softpath_curveto_opi:nn
1750   {
1751     \fp_to_dim:n
1752     { #5 + \c__draw_softpath_arc_fp * ( #1 - #5 ) }
1753   }
1754   {
1755     \fp_to_dim:n
1756     { #6 + \c__draw_softpath_arc_fp * ( #2 - #6 ) }

```

```

1757     }
1758     \exp_not:N \__draw_softpath_curveto_opii:nn
1759     {
1760       \fp_to_dim:n
1761       { #7 + \c__draw_softpath_arc_fp * ( #1 - #7 ) }
1762     }
1763     {
1764       \fp_to_dim:n
1765       { #8 + \c__draw_softpath_arc_fp* ( #2 - #8 ) }
1766     }
1767     \exp_not:N \__draw_softpath_curveto_opiii:nn
1768     {#7} {#8}
1769   }

```

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.

```

1770 \cs_new_protected:Npn \__draw_softpath_round_close:nn #1#2
1771 {
1772   \use:e
1773   {
1774     \__draw_softpath_round_calc:NnnNnn
1775     {
1776       \tl_set:Ne \exp_not:N \l__draw_softpath_move_tl
1777       {
1778         \__draw_softpath_moveto_op:nn
1779         \exp_not:N \exp_after:wN
1780         \exp_not:N \__draw_softpath_round_close:w
1781         \exp_not:N \l__draw_softpath_curve_end_tl
1782         \s__draw_stop
1783       }
1784       \use:e
1785       {
1786         \exp_not:N \exp_not:N \exp_not:N \use_i:nnnn
1787         {
1788           \__draw_softpath_round_loop:Nnn
1789           \__draw_softpath_close_op:nn
1790           \exp_not:N \exp_after:wN
1791           \exp_not:N \__draw_softpath_round_close:w
1792           \exp_not:N \l__draw_softpath_curve_end_tl
1793           \s__draw_stop
1794         }
1795       }
1796     }
1797     {#1} {#2}
1798     \__draw_softpath_lineto_op:nn
1799     \exp_after:wN \use_none:n \l__draw_softpath_move_tl
1800   }
1801 }
1802 \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.

```

1803 \cs_new_protected:Npn \__draw_softpath_round_end:
1804 {

```

```

1805     \tl_put_right:No \l__draw_softpath_main_tl
1806     \l__draw_softpath_move_tl
1807     \tl_put_right:No \l__draw_softpath_main_tl
1808     \l__draw_softpath_part_tl
1809     \tl_build_gbegin:N \g__draw_softpath_main_tl
1810     \__draw_softpath_add:o \l__draw_softpath_main_tl
1811   }

```

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

```
1812 </package>
```

8 I3draw-state implementation

```
1813 <*package>
```

```
1814 <@@=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.

```
1815 \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.

```
1816 \dim_new:N \l_draw_default_linewidth_dim
```

```
1817 \dim_set:Nn \l_draw_default_linewidth_dim { 0.4pt }
```

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

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

```
1818 \cs_new_protected:Npn \draw_set_linewidth:n #1
```

```
1819 {
```

```
1820   \dim_gset:Nn \g__draw_linewidth_dim { \fp_to_dim:n {#1} }
```

```
1821   \__draw_backend_linewidth:n \g__draw_linewidth_dim
```

```
1822 }
```

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

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

```
\l__draw_tmp_seq 1823 \cs_new_protected:Npn \draw_set_dash_pattern:nn #1#2
```

```
1824 {
```

```
1825   \group_begin:
```

```
1826     \seq_set_from_clist:Nn \l__draw_tmp_seq {#1}
```

```
1827     \seq_set_map:NNn \l__draw_tmp_seq \l__draw_tmp_seq
```

```
1828     { \fp_to_dim:n {##1} }
```

```
1829     \use:e
```

```
1830     {
```

```
1831       \__draw_backend_dash_pattern:nn
```

```
1832       { \seq_use:Nn \l__draw_tmp_seq { , } }
```

```

1833         { \fp_to_dim:n {#2} }
1834     }
1835     \group_end:
1836 }
1837 \seq_new:N \l__draw_tmp_seq

```

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

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

```

1838 \cs_new_protected:Npn \draw_set_miterlimit:n #1
1839 { \exp_args:Ne \__draw_backend_miterlimit:n { \fp_eval:n {#1} } }

```

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

`\draw_set_cap_but`: All straight wrappers.

```

\draw_set_cap_rectangle: 1840 \cs_new_protected:Npn \draw_set_cap_but: { \__draw_backend_cap_but: }
\draw_set_cap_round:    1841 \cs_new_protected:Npn \draw_set_cap_rectangle: { \__draw_backend_cap_rectangle: }
\draw_set_evenodd_rule: 1842 \cs_new_protected:Npn \draw_set_cap_round: { \__draw_backend_cap_round: }
\draw_set_nonzero_rule: 1843 \cs_new_protected:Npn \draw_set_evenodd_rule: { \__draw_backend_evenodd_rule: }
\draw_set_join_bevel:   1844 \cs_new_protected:Npn \draw_set_nonzero_rule: { \__draw_backend_nonzero_rule: }
\draw_set_join_miter:   1845 \cs_new_protected:Npn \draw_set_join_bevel: { \__draw_backend_join_bevel: }
\draw_set_join_round:   1846 \cs_new_protected:Npn \draw_set_join_miter: { \__draw_backend_join_miter: }
                        1847 \cs_new_protected:Npn \draw_set_join_round: { \__draw_backend_join_round: }

```

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

```
1848 </package>
```

9 l3draw-transforms implementation

```
1849 <*package>
```

```
1850 <@@=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 specialized, need to understand the requirements here.

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

```
1851 \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 1852 \fp_new:N \l__draw_matrix_a_fp
 \l__draw_matrix_c_fp 1853 \fp_new:N \l__draw_matrix_b_fp
 \l__draw_xshift_dim 1854 \fp_new:N \l__draw_matrix_c_fp
 \l__draw_yshift_dim 1855 \fp_new:N \l__draw_matrix_d_fp
 1856 \dim_new:N \l__draw_xshift_dim
 1857 \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: 1858 \cs_new_protected:Npn \draw_transform_matrix_reset:
 1859 {
 1860 \fp_set:Nn \l__draw_matrix_a_fp { 1 }
 1861 \fp_zero:N \l__draw_matrix_b_fp
 1862 \fp_zero:N \l__draw_matrix_c_fp
 1863 \fp_set:Nn \l__draw_matrix_d_fp { 1 }
 1864 \bool_set_false:N \l__draw_matrix_active_bool
 1865 }
 1866 \cs_new_protected:Npn \draw_transform_shift_reset:
 1867 {
 1868 \dim_zero:N \l__draw_xshift_dim
 1869 \dim_zero:N \l__draw_yshift_dim
 1870 }
 1871 \draw_transform_matrix_reset:
 1872 \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 1873 \cs_new_protected:Npn \draw_transform_matrix_absolute:nmmn #1#2#3#4
 1874 {
 1875 \fp_set:Nn \l__draw_matrix_a_fp {#1}
 1876 \fp_set:Nn \l__draw_matrix_b_fp {#2}
 1877 \fp_set:Nn \l__draw_matrix_c_fp {#3}
 1878 \fp_set:Nn \l__draw_matrix_d_fp {#4}
 1879 \bool_lazy_all:nTF
 1880 {
 1881 { \fp_compare_p:nNn \l__draw_matrix_a_fp = \c_one_fp }
 1882 { \fp_compare_p:nNn \l__draw_matrix_b_fp = \c_zero_fp }
 1883 { \fp_compare_p:nNn \l__draw_matrix_c_fp = \c_zero_fp }
 1884 { \fp_compare_p:nNn \l__draw_matrix_d_fp = \c_one_fp }
 1885 }
 1886 { \bool_set_false:N \l__draw_matrix_active_bool }
 1887 { \bool_set_true:N \l__draw_matrix_active_bool }
 1888 }
 1889 \cs_new_protected:Npn \draw_transform_shift_absolute:n #1
 1890 {
 1891 _draw_point_process:nn
 1892 { _draw_transform_shift_absolute:nn } { \draw_point:n {#1} }

```

1893 }
1894 \cs_new_protected:Npn \__draw_transform_shift_absolute:nn #1#2
1895 {
1896   \dim_set:Nn \l__draw_xshift_dim {#1}
1897   \dim_set:Nn \l__draw_yshift_dim {#2}
1898 }

```

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

`\draw_transform_matrix:n` 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:n
\draw_transform_shift:n
\__draw_transform_shift:nn
1899 \cs_new_protected:Npn \draw_transform_matrix:n #1#2#3#4
1900 {
1901   \use:e
1902   {
1903     \__draw_transform:n
1904     { \fp_eval:n {#1} }
1905     { \fp_eval:n {#2} }
1906     { \fp_eval:n {#3} }
1907     { \fp_eval:n {#4} }
1908   }
1909 }
1910 \cs_new_protected:Npn \__draw_transform:n #1#2#3#4
1911 {
1912   \use:e
1913   {
1914     \draw_transform_matrix_absolute:n
1915     { #1 * \l__draw_matrix_a_fp + #2 * \l__draw_matrix_c_fp }
1916     { #1 * \l__draw_matrix_b_fp + #2 * \l__draw_matrix_d_fp }
1917     { #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_c_fp }
1918     { #3 * \l__draw_matrix_b_fp + #4 * \l__draw_matrix_d_fp }
1919   }
1920 }
1921 \cs_new_protected:Npn \draw_transform_shift:n #1
1922 {
1923   \__draw_point_process:nn
1924   { \__draw_transform_shift:nn } { \draw_point:n {#1} }
1925 }
1926 \cs_new_protected:Npn \__draw_transform_shift:nn #1#2
1927 {
1928   \__draw_transform_shift:n
1929   \l__draw_xshift_dim
1930   \l__draw_yshift_dim
1931   {#1} {#2}
1932 }

```

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

`__draw_transform_shift:n` Apply the current transformation matrix to the shift, then store the resulting values: we may or may not have a non-zero starting point here.

```

1933 \cs_new_protected:Npn \__draw_transform_shift:n #1#2#3#4
1934 {
1935   \dim_set:Nn \l__draw_xshift_dim

```

```

1936     {
1937     \fp_to_dim:n
1938     {
1939     #1 +
1940     ( #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_c_fp )
1941     }
1942     }
1943 \dim_set:Nn \l__draw_yshift_dim
1944 {
1945 \fp_to_dim:n
1946 {
1947 #2 +
1948 ( #3 * \l__draw_matrix_b_fp + #4 * \l__draw_matrix_d_fp )
1949 }
1950 }
1951 }

```

(End of definition for __draw_transform_shift:nmmn.)

```

\draw_transform_matrix_invert: Standard mathematics: calculate the inverse matrix and use that, then undo the shifts.
\__draw_transform_invert:n 1952 \cs_new_protected:Npn \draw_transform_matrix_invert:
\__draw_transform_invert:e 1953 {
\draw_transform_shift_invert: 1954 \bool_if:NT \l__draw_matrix_active_bool
1955 {
1956 \__draw_transform_invert:e
1957 {
1958 \fp_eval:n
1959 {
1960 1 /
1961 (
1962 \l__draw_matrix_a_fp * \l__draw_matrix_d_fp
1963 - \l__draw_matrix_b_fp * \l__draw_matrix_c_fp
1964 )
1965 }
1966 }
1967 }
1968 }
1969 \cs_new_protected:Npn \__draw_transform_invert:n #1
1970 {
1971 \fp_set:Nn \l__draw_matrix_a_fp
1972 { \l__draw_matrix_d_fp * #1 }
1973 \fp_set:Nn \l__draw_matrix_b_fp
1974 { -\l__draw_matrix_b_fp * #1 }
1975 \fp_set:Nn \l__draw_matrix_c_fp
1976 { -\l__draw_matrix_c_fp * #1 }
1977 \fp_set:Nn \l__draw_matrix_d_fp
1978 { \l__draw_matrix_a_fp * #1 }
1979 }
1980 \cs_generate_variant:Nn \__draw_transform_invert:n { e }
1981 \cs_new_protected:Npn \draw_transform_shift_invert:
1982 {
1983 \dim_set:Nn \l__draw_xshift_dim { -\l__draw_xshift_dim }
1984 \dim_set:Nn \l__draw_yshift_dim { -\l__draw_yshift_dim }
1985 }

```

(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.

```

1986 \cs_new_protected:Npn \draw_transform_triangle:nnn #1#2#3
1987 {
1988   \_draw_point_process:nnnn
1989   { \_draw_transform_triangle:nnnnnn }
1990   { \draw_point:n {#1} }
1991   { \draw_point:n {#2} }
1992   { \draw_point:n {#3} }
1993 }
1994 \cs_new_protected:Npn \_draw_transform_triangle:nnnnnn #1#2#3#4#5#6
1995 {
1996   \use:e
1997   {
1998     \draw_transform_matrix_absolute:nnnn
1999     { #3 - #1 }
2000     { #4 - #2 }
2001     { #5 - #1 }
2002     { #6 - #2 }
2003     \draw_transform_shift_absolute:n { #1 , #2 }
2004   }
2005 }

```

(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 2006 \cs_new_protected:Npn \draw_transform_scale:n #1
\draw_transform_yscale:n 2007 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { #1 } }
\draw_transform_xshift:n 2008 \cs_new_protected:Npn \draw_transform_xscale:n #1
\draw_transform_yshift:n 2009 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { 1 } }
\draw_transform_xslant:n 2010 \cs_new_protected:Npn \draw_transform_yscale:n #1
\draw_transform_yslant:n 2011 { \draw_transform_matrix:nnnn { 1 } { 0 } { 0 } { #1 } }
2012 \cs_new_protected:Npn \draw_transform_xshift:n #1
2013 { \draw_transform_shift:n { #1 , Opt } }
2014 \cs_new_protected:Npn \draw_transform_yshift:n #1
2015 { \draw_transform_shift:n { Opt , #1 } }
2016 \cs_new_protected:Npn \draw_transform_xslant:n #1
2017 { \draw_transform_matrix:nnnn { 1 } { 0 } { #1 } { 1 } }
2018 \cs_new_protected:Npn \draw_transform_yslant:n #1
2019 { \draw_transform_matrix:nnnn { 1 } { #1 } { 0 } { 1 } }

```

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

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

```

\_draw_transform_rotate:n 2020 \cs_new_protected:Npn \draw_transform_rotate:n #1
\_draw_transform_rotate:e 2021 { \_draw_transform_rotate:e { \fp_eval:n {#1} } }
\_draw_transform_rotate:nn 2022 \cs_new_protected:Npn \_draw_transform_rotate:n #1
\_draw_transform_rotate:ee 2023 {
2024   \_draw_transform_rotate:ee
2025   { \fp_eval:n { cosd(#1) } }
2026   { \fp_eval:n { sind(#1) } }
2027 }

```

```
2028 \cs_generate_variant:Nn \_draw_transform_rotate:n { e }
2029 \cs_new_protected:Npn \_draw_transform_rotate:nn #1#2
2030 { \draw_transform_matrix:nnnn {#1} {#2} { -#2 } { #1 } }
2031 \cs_generate_variant:Nn \_draw_transform_rotate:nn { ee }
```

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

```
2032 </package>
```

Index

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

B	
<code>\begin</code> . . .	193, 828, 1116, 1120, 1144, 1147
bool commands:	
<code>\bool_gset_eq:NN</code>	1502
<code>\bool_gset_false:N</code>	1485, 1608
<code>\bool_gset_true:N</code>	1509, 1555
<code>\bool_if:NTF</code>	33, 137, 216, 255, 729, 745, 746, 750, 1237, 1270, 1404, 1537, 1592, 1954
<code>\bool_lazy_all:nTF</code>	1879
<code>\bool_lazy_and:nnTF</code>	247, 724, 1638, 1672
<code>\bool_lazy_any:nTF</code>	1683
<code>\bool_lazy_or:nnTF</code>	582, 683, 733, 738
<code>\bool_new:N</code> 108, 242, 672, 673, 674, 675, 795, 1294, 1395, 1471, 1508, 1851	
<code>\bool_set_eq:NN</code>	1493
<code>\bool_set_false:N</code>	118, 250, 711, 712, 713, 732, 1864, 1886
<code>\bool_set_true:N</code>	120, 251, 717, 758, 762, 763, 1313, 1399, 1887
box commands:	
<code>\box_dp:N</code>	17, 22, 89
<code>\box_gset_eq:NN</code>	178
<code>\box_gset_wd:Nn</code>	122, 155
<code>\box_ht:N</code>	17, 22, 91
<code>\box_if_exist:NTF</code>	115
<code>\box_move_down:nn</code>	1369, 1382
<code>\box_move_up:nn</code>	63
<code>\box_new:N</code>	13, 102, 103, 1295, 1296
<code>\box_set_dp:Nn</code>	67, 1373, 1386
<code>\box_set_eq:NN</code>	167
<code>\box_set_ht:Nn</code>	66, 1374, 1392
<code>\box_set_wd:Nn</code>	68, 150, 1343
<code>\box_use_drop:N</code>	64, 69, 124, 151, 156, 1346, 1371, 1384
<code>\box_wd:N</code>	17, 22, 88, 90
C	
clist commands:	
<code>\clist_map_inline:Nn</code>	146, 163, 174
<code>\clist_map_inline:nn</code>	714
<code>\clist_new:N</code>	109, 111
<code>\clist_set:Nn</code>	110, 1331
coffin commands:	
<code>\coffin_typeset:Nnnnn</code>	86
<code>\coffin_wd:N</code>	88
color commands:	
<code>\color_ensure_current:</code>	1318
cs commands:	
<code>\cs_generate_variant:Nn</code>	444, 638, 671, 836, 843, 850, 861, 869, 877, 895, 923, 943, 950, 958, 965, 974, 980, 1002, 1010, 1017, 1023, 1036, 1039, 1057, 1077, 1085, 1091, 1115, 1129, 1143, 1164, 1197, 1216, 1229, 1363, 1474, 1557, 1744, 1980, 2028, 2031
<code>\cs_if_exist:NTF</code>	716, 1354
<code>\cs_if_exist_use:NTF</code>	427, 436, 719
<code>\cs_new:Npn</code> 537, 547, 557, 567, 832, 834, 837, 839, 841, 844, 846, 848, 851, 854, 856, 862, 865, 867, 870, 871, 873, 875, 878, 880, 886, 896, 905, 915, 924, 931, 936, 944, 951, 959, 966, 975, 981, 989, 994, 1003, 1011, 1018, 1024, 1031, 1037, 1040, 1046, 1055, 1058, 1066, 1071, 1078, 1086, 1092, 1101, 1107, 1121, 1130, 1136, 1148, 1150, 1159, 1187, 1189, 1198, 1203, 1217, 1219, 1221, 1230, 1235, 1262, 1268, 1739, 1745, 1802	
<code>\cs_new_protected:Npe</code>	1349
<code>\cs_new_protected:Npn</code>	14, 19, 24, 31, 72, 77, 82, 97, 112, 135, 144, 161, 172, 206, 228, 235, 243, 253, 262, 268, 274, 280, 287, 298, 306, 311, 313, 315, 326, 333, 369, 371, 382, 388, 418, 445, 474, 480, 486, 491, 499, 508, 515, 523, 578, 580, 596, 603, 612, 619, 621, 632, 639, 645, 652, 676, 681, 692, 701, 709, 756, 760, 765, 772, 796, 810, 1171, 1173, 1175, 1177, 1179, 1298, 1305, 1327, 1364, 1377, 1397, 1402, 1417, 1424, 1435, 1444, 1452, 1460, 1472, 1475, 1482, 1487, 1498, 1510, 1519, 1528, 1533, 1543, 1551, 1558, 1560, 1562, 1564, 1566, 1568, 1570, 1572, 1574, 1576, 1578, 1590, 1610, 1634, 1644, 1661, 1670, 1681, 1702, 1711, 1770, 1803, 1818, 1823, 1838, 1840, 1841, 1842, 1843, 1844, 1845, 1846, 1847, 1858, 1866, 1873, 1889, 1894, 1899, 1910,

1921, 1926, 1933, 1952, 1969, 1981,
 1986, 1994, 2006, 2008, 2010, 2012,
 2014, 2016, 2018, 2020, 2022, 2029

D

dim commands:

`\dim_abs:n` 626, 627
`\dim_compare:nNnTF` 634, 641, 774, 1335
`\dim_compare_p:nNn` 248, 249, 1639, 1640
`\dim_gset:Nn` 208, 210, 212,
 214, 218, 220, 222, 224, 230, 231,
 232, 233, 237, 238, 703, 776, 1300,
 1301, 1302, 1303, 1539, 1540, 1820
`\dim_gset_eq:NN`
 813, 814, 815, 816, 817, 818,
 819, 820, 1427, 1446, 1447, 1448, 1449
`\dim_gzero:N` .. 1337, 1338, 1339, 1340
`\dim_horizontal:N` 62
`\dim_max:nn` .. 209, 213, 219, 223, 1388
`\dim_min:nn` 211, 215, 221, 225
`\dim_new:N` 200, 201,
 202, 203, 204, 205, 240, 241, 787,
 788, 789, 790, 791, 792, 793, 794,
 1165, 1166, 1167, 1168, 1169, 1170,
 1290, 1291, 1292, 1293, 1396, 1414,
 1431, 1432, 1433, 1434, 1506, 1507,
 1585, 1586, 1815, 1816, 1856, 1857
`\dim_set:Nn` 245, 246,
 1181, 1182, 1400, 1636, 1637, 1817,
 1896, 1897, 1935, 1943, 1983, 1984
`\dim_set_eq:NN`
 799, 800, 801, 802, 803, 804,
 805, 806, 1421, 1438, 1439, 1440, 1441
`\dim_step_inline:nnnn` 654, 662
`\dim_use:N`
 . 705, 774, 779, 781, 1409, 1515, 1516
`\dim_zero:N` 1868, 1869
`\c_max_dim` 230, 231, 232,
 233, 774, 1300, 1301, 1302, 1303, 1335

draw commands:

`\l_draw_bb_update_bool`
 33, 216, 725, 732, 1294, 1313
`\g_draw_bb_xmax_dim` 218,
 219, 1290, 1300, 1337, 1344, 1438, 1446
`\g_draw_bb_xmin_dim`
 220, 221, 1290, 1301, 1335,
 1338, 1344, 1368, 1381, 1439, 1447
`\g_draw_bb_ymax_dim` 222, 223, 1290,
 1302, 1339, 1375, 1393, 1440, 1448
`\g_draw_bb_ymin_dim`
 224, 225, 1290, 1303,
 1340, 1370, 1375, 1389, 1441, 1449
`\draw_begin:` 1305, 1305
`\draw_box_use:N` 14, 14

`\draw_box_use:Nn` 14, 19
`\draw_coffin_use:Nnn` 36, 72, 72
`\draw_coffin_use:Nnnn` 72, 77
`\l_draw_default_linewidth_dim` ...
 127, 1317, 1816
`\draw_end:` 1305, 1327
`\g_draw_id_int` 1297, 1308, 1357
`\draw_layer_begin:n` 112, 112
`\draw_layer_end:` 112, 135
`\draw_layer_new:n` 97, 97
`\l_draw_layers_clist`
 109, 146, 163, 174, 1331
`\draw_path_arc:nnn` 369, 369, 512
`\draw_path_arc:nnnn` ... 369, 370, 371
`\draw_path_arc_axes:nnnn` ... 508, 508
`\draw_path_canvas_curveto:nnn` ...
 311, 315
`\draw_path_canvas_lineto:n` . 311, 313
`\draw_path_canvas_moveto:n` . 311, 311
`\draw_path_circle:nn` 578, 578
`\draw_path_close:` 306, 306, 609
`\draw_path_corner_arc:nn` ... 243, 243
`\draw_path_curveto:mn` 326, 326
`\draw_path_curveto:nnn` 262, 287
`\draw_path_ellipse:nnn` . 515, 515, 579
`\draw_path_grid:nnnn` 621, 621
`\draw_path_lineto:n`
 262, 274, 606, 607, 608, 660, 668
`\draw_path_moveto:n`
 262, 262, 605, 610, 659, 667
`\draw_path_rectangle:nn` 580, 580, 620
`\draw_path_rectangle_corners:nn` .
 612, 612
`\draw_path_replace_bb:` 676, 692
`\draw_path_scope_begin:`
 796, 796, 1422, 1455
`\draw_path_scope_end:`
 796, 810, 1426, 1463
`\draw_path_use:n` 676, 676
`\draw_path_use_clear:n` 676, 681
`\draw_point:n`
 . 312, 314, 322, 323, 324, 587, 588,
 592, 593, 616, 617, 629, 630, 865,
 865, 876, 879, 891, 900, 901, 902,
 903, 917, 928, 929, 977, 985, 986,
 987, 1020, 1028, 1029, 1038, 1056,
 1062, 1063, 1064, 1088, 1096, 1097,
 1098, 1099, 1161, 1178, 1191, 1205,
 1223, 1233, 1239, 1255, 1266, 1272,
 1285, 1892, 1924, 1990, 1991, 1992
`\draw_point_interpolate_arcaxes:nnnnnn`
 1058, 1058
`\draw_point_interpolate_curve:nnnnn`
 1092

<code>\draw_point_interpolate_curve:nnnnn</code>	1092	<code>\draw_set_linewidth:n</code>	127, 1317, 1818, 1818
<code>\draw_point_interpolate_curve_</code> <code>auxi:nnnnnnnn</code>	1092	<code>\draw_set_miterlimit:n</code>	1322, 1838, 1838
<code>\draw_point_interpolate_curve_</code> <code>auxii:nnnnnnnn</code>	1092	<code>\draw_set_nonzero_rule:</code>	1319, 1840, 1844
<code>\draw_point_interpolate_curve_</code> <code>auxiii:nnnnn</code>	1092	<code>\draw_suspend_begin:</code>	1452, 1452
<code>\draw_point_interpolate_curve_</code> <code>auxiv:nnnnn</code>	1092	<code>\draw_suspend_end:</code>	1452, 1460
<code>\draw_point_interpolate_curve_</code> <code>auxv:nnw</code>	1092	<code>\draw_transform_matrix:nnnn</code>	1899, 1899, 2007, 2009, 2011, 2017, 2019, 2030
<code>\draw_point_interpolate_curve_</code> <code>auxvi:n</code>	1092	<code>\draw_transform_matrix_absolute:nnnn</code>	1873, 1873, 1914, 1998
<code>\draw_point_interpolate_curve_</code> <code>auxvii:nnnnnnnn</code>	1092	<code>\draw_transform_matrix_invert:</code>	1952, 1952
<code>\draw_point_interpolate_curve_</code> <code>auxviii:nnnnn</code>	1092	<code>\draw_transform_matrix_reset:</code>	1314, 1456, 1858, 1858, 1871
<code>\draw_point_interpolate_distance:nnn</code>	1040, 1040, 1715, 1724	<code>\draw_transform_rotate:n</code>	2020, 2020
<code>\draw_point_interpolate_line:nnn</code>	1024, 1024	<code>\draw_transform_scale:n</code>	2006, 2006
<code>\draw_point_intersect_circles:nnnnn</code>	924, 924	<code>\draw_transform_shift:n</code>	27, 1899, 1921, 2013, 2015
<code>\draw_point_intersect_line_</code> <code>circle:nnnnn</code>	981, 981	<code>\draw_transform_shift_absolute:n</code>	1873, 1889, 2003
<code>\draw_point_intersect_lines:nnnn</code>	896, 896	<code>\draw_transform_shift_invert:</code>	1952, 1981
<code>\draw_point_polar:nn</code>	871, 871	<code>\draw_transform_shift_reset:</code>	1315, 1457, 1858, 1866, 1872
<code>\draw_point_polar:nnn</code>	452, 458, 462, 468, 871, 872, 873	<code>\draw_transform_triangle:nnn</code>	511, 1986, 1986
<code>\draw_point_transform:n</code>	37, 40, 43, 46, 266, 278, 294, 295, 296, 330, 331, 457, 461, 519, 1230, 1230	<code>\draw_transform_xscale:n</code>	2006, 2008
<code>\draw_point_unit_vector:n</code>	878, 878, 1053	<code>\draw_transform_xshift:n</code>	2006, 2012
<code>\draw_point_vec:nn</code>	1187, 1187	<code>\draw_transform_xslant:n</code>	2006, 2016
<code>\draw_point_vec:nnn</code>	1187, 1198	<code>\draw_transform_yscale:n</code>	2006, 2010
<code>\draw_point_vec_polar:nn</code>	1217, 1217	<code>\draw_transform_yshift:n</code>	2006, 2014
<code>\draw_point_vec_polar:nnn</code>	1217, 1218, 1219	<code>\draw_transform_yslant:n</code>	2006, 2018
<code>\draw_scope_begin:</code>	26, 1417, 1417	<code>\draw_xvec:n</code>	1171, 1171, 1184
<code>\draw_scope_end:</code>	29, 1424	<code>\draw_yvec:n</code>	1171, 1173, 1185
<code>\draw_set_baseline:n</code>	1397, 1397	<code>\draw_zvec:n</code>	1171, 1175, 1186
<code>\draw_set_cap_but:</code>	1320, 1840, 1840	draw internal commands:	
<code>\draw_set_cap_rectangle:</code>	1840, 1841	<code>_draw_backend_begin:</code>	1310
<code>\draw_set_cap_round:</code>	1840, 1842	<code>_draw_backend_box_use:Nnnnn</code>	53
<code>\draw_set_dash_pattern:nn</code>	1323, 1823, 1823	<code>_draw_backend_cap_but:</code>	1840
<code>\draw_set_evenodd_rule:</code>	1840, 1843	<code>_draw_backend_cap_rectangle:</code>	1841
<code>\draw_set_join_bevel:</code>	1840, 1845	<code>_draw_backend_cap_round:</code>	1842
<code>\draw_set_join_miter:</code>	1321, 1840, 1846	<code>_draw_backend_clip:</code>	731
<code>\draw_set_join_round:</code>	1840, 1847	<code>_draw_backend_closepath:</code>	1559
		<code>_draw_backend_curveto:nnnnn</code>	1563
		<code>_draw_backend_dash_pattern:nn</code>	1831
		<code>_draw_backend_discardpath:</code>	736
		<code>_draw_backend_end:</code>	1333
		<code>_draw_backend_evenodd_rule:</code>	1843
		<code>_draw_backend_join_bevel:</code>	1845
		<code>_draw_backend_join_miter:</code>	1846
		<code>_draw_backend_join_round:</code>	1847

<code>__draw_backend_lineto:nn</code>	1569	<code>\l__draw_matrix_a_fp</code>	54,
<code>__draw_backend_linewidth:n</code>	..	1821			1242, 1275, 1852, 1860, 1875, 1881,
<code>__draw_backend_miterlimit:n</code>	.	1839			1915, 1917, 1940, 1962, 1971, 1978
<code>__draw_backend_moveto:nn</code>	1571	<code>\l__draw_matrix_active_bool</code>	
<code>__draw_backend_nonzero_rule:</code>	.	1844			584, 1237,
<code>__draw_backend_rectangle:nmnn</code>		1577			1270, 1851, 1864, 1886, 1887, 1954
<code>__draw_backend_scope_begin:</code>	...		<code>\l__draw_matrix_b_fp</code>	55,
	154, 1419			1248, 1280, 1852, 1861, 1876, 1882,
<code>__draw_backend_scope_end:</code>		157, 1429			1916, 1918, 1948, 1963, 1973, 1974
<code>\l__draw_baseline_bool</code>		<code>\l__draw_matrix_c_fp</code>	56,
	1395, 1399, 1404			1243, 1276, 1852, 1862, 1877, 1883,
<code>\l__draw_baseline_dim</code>		1395, 1400, 1409			1915, 1917, 1940, 1963, 1975, 1976
<code>__draw_baseline_finalize:w</code>		<code>\l__draw_matrix_d_fp</code>	57,
	1329, 1402, 1402			1249, 1281, 1855, 1863, 1878, 1884,
<code>__draw_box_use:Nmnnn</code>	14			1916, 1918, 1948, 1962, 1972, 1977
<code>__draw_box_use:nNmnnn</code>	21, 24, 79	<code>__draw_path_arc:nmnn</code>	.	369, 375, 382
<code>__draw_box_use:Nmnnmnn</code>		16, 28, 31, 74	<code>__draw_path_arc:nnNm</code>	
<code>__draw_box_use:nNmnnmnn</code>	14			369, 385, 386, 388
<code>__draw_coffin_use:nNm</code>		72, 74, 79, 82	<code>\c__draw_path_arc_60_fp</code>	369
<code>\l__draw_corner_arc_bool</code>		<code>\c__draw_path_arc_90_fp</code>	369
	242, 250, 251, 255, 583	<code>__draw_path_arc_add:nmnn</code>	369
<code>\l__draw_corner_xarc_dim</code>		<code>__draw_path_arc_aux_add:nn</code>	
	240, 245, 248, 258			476, 482, 494, 499
<code>\l__draw_corner_yarc_dim</code>		<code>__draw_path_arc_auxi:nmnnNm</code>	...	
	240, 246, 249, 259			369, 396, 403, 411, 418, 444
<code>__draw_draw_polar:nm</code>		<code>__draw_path_arc_auxii:nmnnmnn</code>	.	
	871, 874, 875, 877			369, 422, 445
<code>__draw_draw_vec_polar:nm</code>		<code>__draw_path_arc_auxiii:nn</code>	
	1220, 1221, 1229			369, 449, 474
<code>\l__draw_fill_color_tl</code>	1414	<code>__draw_path_arc_auxiv:nmnn</code>	
<code>__draw_finalize:</code>				369, 455, 480
	1342, 1364, 1364, 1402, 1412	<code>__draw_path_arc_auxv:nn</code>		369, 465, 486
<code>__draw_finalize_baseline:n</code>		<code>__draw_path_arc_auxvi:nn</code>	
	1364, 1377, 1409			369, 488, 491
<code>__draw_if_recursion_tail_stop_</code>			<code>\l__draw_path_arc_delta_fp</code>	369
<code>do:Nn</code>	9, 9, 1612	<code>\l__draw_path_arc_start_fp</code>	369
<code>\l__draw_layer_close_bool</code>		<code>__draw_path_curveto:nmnn</code>	
	108, 118, 120, 137			326, 329, 333
<code>\l__draw_layer_main_box</code>		<code>__draw_path_curveto:nmnnmnn</code>	
	150, 151, 1295, 1324			262, 292,
<code>\l__draw_layer_tl</code>	106, 117, 121			298, 320, 340, 470, 539, 549, 559, 569
<code>\g__draw_layers_clist</code>	109	<code>\c__draw_path_curveto_a_fp</code>	326
<code>__draw_layers_insert:</code>		144, 144, 1332	<code>\c__draw_path_curveto_b_fp</code>	326
<code>__draw_layers_restore:</code>		161, 172, 1462	<code>__draw_path_ellipse:nmnnmnn</code>	
<code>__draw_layers_save:</code>	.	161, 161, 1458			515, 518, 523
<code>\g__draw_linewidth_dim</code>		<code>__draw_path_ellipse_arci:nmnnmnn</code>	
	706, 782, 1421, 1427, 1815, 1820, 1821			515, 529, 537
<code>\l__draw_linewidth_dim</code>		<code>__draw_path_ellipse_arci:nnmnnmnn</code>	
	1414, 1421, 1427			515, 530, 547
<code>\l__draw_main_box</code>		<code>__draw_path_ellipse_arci:nnmnnmnn</code>	
	1295, 1309, 1343, 1346, 1366, 1371,			515, 531, 557
	1373, 1374, 1379, 1384, 1386, 1392	<code>__draw_path_ellipse_arci:nnmnnmnn</code>	
				515, 532, 567
		<code>\c__draw_path_ellipse_fp</code>	515

<code>__draw_path_grid_auxi:nnnnnn</code> . . .	<code>\l__draw_path_use_fill_bool</code>
. 621 , 625 , 632 , 638 672 , 712 , 734 , 739 , 745 , 762
<code>__draw_path_grid_auxii:nnnnnn</code> . . .	<code>__draw_path_use_stroke_bb:</code>
. 621 , 635 , 636 , 639 676 , 727 , 765
<code>__draw_path_grid_auxiii:nnnnnn</code> . . .	<code>\l__draw_path_use_stroke_bool</code> . . .
. 621 , 642 , 643 , 645 672 , 713 , 726 , 735 , 740 , 746 , 758 , 763
<code>__draw_path_grid_auxiiii:nnnnnn</code> 621	<code>\g__draw_path_xmax_dim</code>
<code>__draw_path_grid_auxiv:nnnnnnnn</code> 202 , 208 , 209 , 230 , 801 , 815
. 621 , 647 , 652 , 671	<code>\l__draw_path_xmax_dim</code> . 787 , 801 , 815
<code>\g__draw_path_lastx_dim</code>	<code>\g__draw_path_xmin_dim</code>
. 200 , 237 , 344 , 477 , 483 , 799 , 819 202 , 210 , 211 , 231 , 802 , 816
<code>\l__draw_path_lastx_dim</code> 787 , 799 , 819	<code>\l__draw_path_xmin_dim</code> . 787 , 802 , 816
<code>\g__draw_path_lasty_dim</code>	<code>\g__draw_path_ymax_dim</code>
. 200 , 238 , 351 , 478 , 484 , 800 , 820 202 , 212 , 213 , 232 , 803 , 817
<code>\l__draw_path_lasty_dim</code> 787 , 800 , 820	<code>\l__draw_path_ymax_dim</code> . 787 , 803 , 817
<code>__draw_path_lineto:nn</code>	<code>\g__draw_path_ymin_dim</code>
. 262 , 277 , 280 , 314 202 , 214 , 215 , 233 , 804 , 818
<code>__draw_path_mark_corner:</code>	<code>\l__draw_path_ymin_dim</code> . 787 , 804 , 818
. 253 , 253 , 282 , 291 , 308 , 319 , 339 , 410	<code>__draw_point_interpolate_-</code>
<code>__draw_path_moveto:nn</code>	arcaxes_auxi:nnnnnnnnnn
. 262 , 265 , 268 , 312 , 527 , 535 1058 , 1061 , 1066
<code>__draw_path_rectangle:nnnn</code>	<code>__draw_point_interpolate_-</code>
. 580 , 591 , 596	arcaxes_auxii:nnnnnnnnnn
<code>__draw_path_rectangle_corners:nnnn</code> 1058 , 1068 , 1071 , 1077
. 612	<code>__draw_point_interpolate_-</code>
<code>__draw_path_rectangle_corners:nnnnnn</code>	arcaxes_auxiii:nnnnnnnn
. 615 , 619 1058 , 1073 , 1078 , 1085
<code>__draw_path_rectangle_rounded:nnnn</code>	<code>__draw_point_interpolate_-</code>
. 580 , 586 , 603	arcaxes_auxiv:nnnnnnnnnn
<code>__draw_path_replace_bb:NnN</code> 1058 , 1080 , 1086 , 1091
. 676 , 694 , 695 , 696 , 697 , 701	<code>__draw_point_interpolate_curve_-</code>
<code>__draw_path_reset_limits:</code>	auxi:nnnnnnnnnn 1095 , 1101
. 206 , 228 , 688 , 699 , 753 , 807 , 1312	<code>__draw_point_interpolate_curve_-</code>
<code>\l__draw_path_tmp_tl</code>	auxii:nnnnnnnnnn . 1103 , 1107 , 1115
. 197 , 447 , 470 , 489 , 493 , 497 , 501	<code>__draw_point_interpolate_curve_-</code>
<code>\l__draw_path_tmpa_fp</code>	auxiii:nnnnnn 1110 , 1121 , 1129
. 197 , 335 , 345 , 357	<code>__draw_point_interpolate_curve_-</code>
<code>\l__draw_path_tmpb_fp</code>	auxiv:nnnnnn 1123 , 1124 , 1125 , 1130
. 197 , 336 , 352 , 361	<code>__draw_point_interpolate_curve_-</code>
<code>__draw_path_update_last:nn</code>	auxv:nnw 1132 , 1136 , 1143
. 235 , 235 , 272 , 285 , 304 , 601	<code>__draw_point_interpolate_curve_-</code>
<code>__draw_path_update_limits:nn</code> . . .	auxvi:n 1127 , 1148
. 36 , 39 , 42 , 45 , 206 ,	<code>__draw_point_interpolate_curve_-</code>
. 206 , 270 , 283 , 300 , 301 , 302 , 598 , 599	auxvii:nnnnnnnnnn 1149 , 1150
<code>__draw_path_use:n</code> . 676 , 679 , 690 , 709	<code>__draw_point_interpolate_curve_-</code>
<code>__draw_path_use_action_draw:</code> . . .	auxviii:nnnnnn 1152 , 1159 , 1164
. 676 , 756	<code>__draw_point_interpolate_-</code>
<code>__draw_path_use_action_fillstroke:</code>	distance:nnnn 1043 , 1046
. 676 , 760	<code>__draw_point_interpolate_-</code>
<code>__draw_path_use_bb:NnN</code>	distance:nnnnn
. 676 , 767 , 768 , 769 , 770 , 772 1040 , 1050 , 1055 , 1057
<code>\l__draw_path_use_clear_bool</code> 675 , 750	<code>__draw_point_interpolate_-</code>
<code>\l__draw_path_use_clip_bool</code>	distance:nnnnnn 1040
. 672 , 711 , 729	

<code>_draw_point_interpolate_line_-</code>	<code>_draw_point_process_auxiv:nw</code> ..
aux:nnnnn .. 1024 , 1027 , 1031 , 1036 832 , 842 , 844
<code>_draw_point_interpolate_line_-</code>	<code>_draw_point_process_auxv:nnnn</code> ..
aux:nnnnnn .. 1024 , 1033 , 1037 , 1039 832 , 847 , 848 , 850
<code>_draw_point_intersect_circles_-</code>	<code>_draw_point_process_auxvi:nw</code> ..
auxi:nnnnnnn .. 924 , 927 , 931 832 , 849 , 851
<code>_draw_point_intersect_circles_-</code>	<code>_draw_point_process_auxvii:nnnnn</code>
auxii:nnnnnnn .. 924 , 933 , 936 , 943 832 , 855 , 856 , 861
<code>_draw_point_intersect_circles_-</code>	<code>_draw_point_process_auxviii:nw</code>
auxiii:nnnnnnn .. 924 , 938 , 944 , 950 832 , 858 , 862
<code>_draw_point_intersect_circles_-</code>	<code>_draw_point_to_dim:n</code> ..
auxiv:nnnnnnnn .. 924 , 946 , 951 , 958 865 , 866 , 867 , 869
<code>_draw_point_intersect_circles_-</code>	<code>_draw_point_to_dim:w</code> .. 865 , 868 , 870
auxv:nnnnnnnnn .. 924 , 953 , 959 , 965	<code>_draw_point_transform:nn</code> ..
<code>_draw_point_intersect_circles_-</code> 1230 , 1233 , 1235
auxvi:nnnnnnnn .. 924 , 961 , 966 , 974	<code>_draw_point_transform_noshift:n</code>
<code>_draw_point_intersect_circles_-</code> 451 , 467 , 520 , 521 , 1262 , 1262
auxvii:nnnnnnn .. 924 , 968 , 975 , 980	<code>_draw_point_transform_noshift:nn</code>
<code>_draw_point_intersect_line_-</code> 1262 , 1265 , 1268
circle_auxi:nnnnnnnn 981 , 984 , 989	<code>_draw_point_unit_vector:nn</code> ..
<code>_draw_point_intersect_line_-</code> 878 , 879 , 880
circle_auxii:nnnnnnnn ..	<code>_draw_point_unit_vector:nnn</code> ..
..... 981 , 991 , 994 , 1002 878 , 882 , 886 , 895
<code>_draw_point_intersect_line_-</code>	<code>_draw_point_vec:nn</code> ..
circle_auxiii:nnnnnnnn 1187 , 1188 , 1189 , 1197
..... 981 , 996 , 1003 , 1010	<code>_draw_point_vec:nnn</code> ..
<code>_draw_point_intersect_line_-</code> 1187 , 1200 , 1203 , 1216
circle_auxiv:nnnnnnnn ..	<code>_draw_point_vec_polar:nnn</code> .. 1217
..... 981 , 1005 , 1011 , 1017	<code>_draw_record_origin:</code> ..
<code>_draw_point_intersect_line_-</code> 1325 , 1349 , 1349
circle_auxv:nnnnn ..	<code>_draw_reset_bb:</code> ..
..... 981 , 1013 , 1018 , 1023 1298 , 1298 , 1311 , 1442
<code>_draw_point_intersect_lines:nnnnnn</code>	<code>_draw_scope_bb_begin:</code> ..
..... 896 1435 , 1435 , 1454
<code>_draw_point_intersect_lines:nnnnnnnn</code>	<code>_draw_scope_bb_end:</code> 1435 , 1444 , 1464
..... 896 , 899 , 905	<code>_draw_softpath_add:n</code> ..
<code>_draw_point_intersect_lines_-</code>	... 1472 , 1472 , 1474 , 1501 , 1512 ,
aux:nnnnnn .. 896 , 907 , 915 , 923	1521 , 1530 , 1535 , 1545 , 1553 , 1810
<code>_draw_point_process:nn</code> ..	<code>\c_draw_softpath_arc_fp</code> ..
... 35 , 38 , 41 , 44 , 264 , 276 , 312 , 1589 , 1752 , 1756 , 1761 , 1765
314 , 448 , 464 , 832 , 832 , 879 , 1042 ,	<code>_draw_softpath_clear:</code> .. 687 ,
1048 , 1178 , 1232 , 1264 , 1891 , 1923	698 , 752 , 1316 , 1475 , 1482 , 1496 , 1500
<code>_draw_point_process:nnn</code> 328 , 454 ,	<code>_draw_softpath_close_op:nn</code> ..
586 , 591 , 614 , 623 , 832 , 839 , 926 , 1026	.. 1514 , 1558 , 1558 , 1651 , 1687 , 1789
<code>_draw_point_process:nnnn</code> .. 289 ,	<code>_draw_softpath_closepath:</code> ..
317 , 517 , 832 , 846 , 983 , 1060 , 1988 309 , 534 , 1510 , 1510
<code>_draw_point_process:nnnnn</code> ..	<code>\l_draw_softpath_corneri_dim</code> ..
..... 832 , 854 , 898 , 1094 1583 , 1636 , 1639 , 1725
<code>_draw_point_process_auxi:nn</code> ..	<code>\l_draw_softpath_cornerii_dim</code> ..
..... 832 , 833 , 834 , 836 1583 , 1637 , 1640 , 1716
<code>_draw_point_process_auxii:nw</code> ..	<code>\g_draw_softpath_corners_bool</code> ..
..... 832 , 835 , 837 1471 ,
<code>_draw_point_process_auxiii:nnn</code>	1485 , 1495 , 1503 , 1555 , 1592 , 1608
..... 832 , 840 , 841 , 843	

\l__draw_softpath_corners_bool	787, 1494, 1504	__draw_softpath_rectangle_- opi:nn	1547, 1558, 1574
\l__draw_softpath_curve_end_tl	1582, 1713, 1732, 1781, 1792	__draw_softpath_rectangle_- opi:nnNnn	1558, 1575, 1576
__draw_softpath_curveto:nnnnnn	303, 1510, 1519	__draw_softpath_rectangle_- opii:nn	1548, 1558, 1578, 1579
__draw_softpath_curveto_opi:nn	1523, 1558, 1560, 1648, 1686, 1749	__draw_softpath_restore:	812, 1487, 1498
__draw_softpath_curveto_- opi:nnNnnNnn	1558, 1561, 1562	__draw_softpath_round_action:nn	1590, 1614, 1634
__draw_softpath_curveto_opii:nn	1524, 1558, 1564, 1565, 1758	__draw_softpath_round_action:Nnn	1590, 1642, 1644
__draw_softpath_curveto_- opiii:nn	1525, 1558, 1566, 1567, 1767	__draw_softpath_round_action_- close:	1590, 1652, 1670
\l__draw_softpath_first_tl	1583, 1599, 1616, 1617, 1627, 1646, 1647, 1673, 1677	__draw_softpath_round_action_- curveto:NnnNnn	1590, 1649, 1661
\g__draw_softpath_lastx_dim	805, 813, 1506, 1515, 1539	__draw_softpath_round_calc:NnnNnn	1590, 1690, 1705, 1711, 1774
\l__draw_softpath_lastx_dim	793, 805, 813	__draw_softpath_round_calc:nnnnnn	1590, 1722, 1739, 1744
\l__draw_softpath_lastx_fp	1583, 1597, 1618, 1666, 1728, 1735	__draw_softpath_round_calc:nnnnw	1590, 1741, 1745
\g__draw_softpath_lasty_dim	806, 814, 1506, 1516, 1540	__draw_softpath_round_close:nn	1590, 1676, 1770
\l__draw_softpath_lasty_dim	794, 806, 814	__draw_softpath_round_close:w	1590, 1780, 1791, 1802
\l__draw_softpath_lasty_fp	1583, 1598, 1619, 1667, 1729, 1736	__draw_softpath_round_corners:	723, 1590, 1590
__draw_softpath_lineto:nn	284, 1510, 1528	__draw_softpath_round_end:	1590, 1612, 1803
__draw_softpath_lineto_op:nn	1531, 1558, 1568, 1654, 1685, 1798	__draw_softpath_round_lookahead:NnnNnn	1590, 1655, 1668, 1681
\g__draw_softpath_main_tl	1469, 1473, 1478, 1484, 1489, 1492, 1601, 1603, 1809	__draw_softpath_round_loop:Nnn	1590, 1602, 1610, 1631, 1641, 1656, 1679, 1691, 1697, 1706, 1788
\l__draw_softpath_main_tl	20, 1491, 1501, 1580, 1595, 1622, 1624, 1805, 1807, 1810	__draw_softpath_round_roundpoint:NnnNnnNnn	1590, 1696, 1702
\g__draw_softpath_move_bool	1508, 1537	__draw_softpath_roundpoint:nn	257, 1510, 1551, 1557
\l__draw_softpath_move_tl	1583, 1600, 1623, 1626, 1674, 1776, 1799, 1806	__draw_softpath_roundpoint_- op:nn	1554, 1558, 1572, 1573, 1613, 1695
__draw_softpath_moveto:nn	271, 1510, 1533	__draw_softpath_save: 808, 1487, 1487	
__draw_softpath_moveto_op:nn	1536, 1558, 1570, 1620, 1778	\l__draw_softpath_tmp_tl	1470, 1479, 1480
\l__draw_softpath_part_tl	1581, 1596, 1625, 1628, 1630, 1664, 1719, 1808	__draw_softpath_use: 728, 1475, 1475	
__draw_softpath_rectangle:nnnn	600, 1510, 1543	\l__draw_stroke_color_tl	1414
		\l__draw_tmp_box	13, 49, 60, 64, 66, 67, 68, 69, 85, 87, 88, 89, 90, 91
		\l__draw_tmp_seq	1823
		__draw_transform:nnnn	
		__draw_transform_invert:n	1899, 1903, 1910
			1952, 1956, 1969, 1980

<code>__draw_transform_rotate:n</code>	<code>\fp_eval:n</code>
. 2020 , 2021 , 2022 , 2028	376 , 377 , 398 , 405 , 414 ,
<code>__draw_transform_rotate:nn</code>	866 , 874 , 883 , 908 , 909 , 910 , 911 ,
. 2020 , 2024 , 2029 , 2031	912 , 913 , 934 , 939 , 940 , 947 , 954 ,
<code>__draw_transform_shift:nn</code>	955 , 962 , 969 , 971 , 992 , 997 , 998 ,
. 1899 , 1924 , 1926	999 , 1006 , 1014 , 1027 , 1033 , 1051 ,
<code>__draw_transform_shift:nnnn</code>	1069 , 1074 , 1081 , 1082 , 1104 , 1111 ,
. 1928 , 1933 , 1933	1133 , 1134 , 1153 , 1154 , 1155 , 1156 ,
<code>__draw_transform_shift_absolute:nn</code>	1188 , 1201 , 1220 , 1839 , 1904 , 1905 ,
. 1873 , 1892 , 1894	1906 , 1907 , 1958 , 2021 , 2025 , 2026
<code>__draw_transform_triangle:nnnnnn</code>	<code>\fp_new:N</code> 198 , 199 , 504 ,
. 1989 , 1994	505 , 1583 , 1584 , 1852 , 1853 , 1854 , 1855
<code>__draw_vec:nn</code>	<code>\fp_set:Nn</code> 335 , 336 , 390 , 391 ,
. 1171 , 1172 , 1174 , 1176 , 1177	471 , 472 , 1618 , 1619 , 1666 , 1667 ,
<code>__draw_vec:nnn</code> 1171 , 1178 , 1179	1735 , 1736 , 1860 , 1863 , 1875 , 1876 ,
<code>\l__draw_xmax_dim</code> 1431 , 1438 , 1446	1877 , 1878 , 1971 , 1973 , 1975 , 1977
<code>\l__draw_xmin_dim</code> 1431 , 1439 , 1447	<code>\fp_to_decimal:N</code> 397 , 404 , 412
<code>\l__draw_xshift_dim</code>	<code>\fp_to_dim:n</code> 245 ,
. 62 , 592 , 1244 , 1258 ,	246 , 342 , 349 , 356 , 360 , 378 , 379 ,
1852 , 1868 , 1896 , 1929 , 1935 , 1983	425 , 434 , 502 , 528 , 540 , 541 , 542 ,
<code>\l__draw_xvec_x_dim</code>	543 , 544 , 545 , 550 , 551 , 552 , 553 ,
. 1165 , 1193 , 1207 , 1225	554 , 555 , 560 , 561 , 562 , 563 , 564 ,
<code>\l__draw_xvec_y_dim</code> 1165 , 1194 , 1211	565 , 570 , 571 , 572 , 573 , 574 , 575 ,
<code>\l__draw_ymax_dim</code> 1431 , 1440 , 1448	648 , 649 , 1400 , 1751 , 1755 , 1760 ,
<code>\l__draw_ymin_dim</code> 1431 , 1441 , 1449	1764 , 1820 , 1828 , 1833 , 1937 , 1945
<code>\l__draw_yshift_dim</code>	<code>\fp_use:N</code> 54 , 55 , 56 , 57 , 577
. 63 , 592 , 1250 , 1258 ,	<code>\fp_while_do:nNnn</code> 392
1852 , 1869 , 1897 , 1930 , 1943 , 1984	<code>\fp_zero:N</code> 1597 , 1598 , 1861 , 1862
<code>\l__draw_yvec_x_dim</code> 1165 , 1193 , 1208	<code>\c_one_fp</code> 1881 , 1884
<code>\l__draw_yvec_y_dim</code>	<code>\c_zero_fp</code> 888 , 1882 , 1883
. 1165 , 1194 , 1212 , 1226	
<code>\l__draw_zvec_x_dim</code> 1165 , 1209	
<code>\l__draw_zvec_y_dim</code> 1165 , 1213	
	G
E	group commands:
<code>\end</code> 191 , 826	<code>\group_begin:</code> 48 , 84 , 114 , 125 ,
exp commands:	510 , 798 , 1307 , 1420 , 1437 , 1594 , 1825
<code>\exp_after:wN</code>	<code>\group_end:</code> 70 , 92 , 139 , 142 ,
470 , 488 , 1602 , 1676 , 1779 , 1790 , 1799	513 , 821 , 1347 , 1428 , 1450 , 1606 , 1835
<code>\exp_args:Ne</code> 1839	
<code>\exp_args:NNNV</code> 1330	H
<code>\exp_not:N</code> 1356 , 1357 ,	hbox commands:
1721 , 1749 , 1758 , 1767 , 1776 , 1779 ,	<code>\hbox_gset:Nw</code> 123
1780 , 1781 , 1786 , 1790 , 1791 , 1792	<code>\hbox_gset_end:</code> 140
<code>\exp_not:n</code> 1408	<code>\hbox_set:Nn</code> 49 , 60 , 85 , 1366 , 1379
	<code>\hbox_set:Nw</code> 1309 , 1324
F	<code>\hbox_set_end:</code> 1330 , 1334
fp commands:	<code>\hbox_to_wd:nn</code> 1351
<code>\fp_compare:nNnTF</code> 384 , 394 , 888	
<code>\fp_compare_p:nNn</code>	I
. 1881 , 1882 , 1883 , 1884	int commands:
<code>\fp_const:Nn</code>	<code>\int_gincr:N</code> 1308
. 367 , 368 , 506 , 507 , 577 , 1589	<code>\int_if_odd:nTF</code> 970 , 1007
	<code>\int_new:N</code> 1297
	<code>\int_use:N</code> 1357

K	<code>\ProvidesExplPackage</code> 3
kernel internal commands:	
<code>__kernel_quark_new_test:N</code> 9	
M	Q
mode commands:	quark commands:
<code>\mode_leave_vertical:</code> 1345	<code>\quark_new:N</code> 7, 8
msg commands:	quark internal commands:
<code>\msg_error:nnn</code> 100, 131, 132, 720	<code>\q__draw_recursion_stop</code> 7, 1605
<code>\msg_new:nnn</code> 186	<code>\q__draw_recursion_tail</code> 7, 1604
<code>\msg_new:nnnn</code> 183, 188, 823	
P	S
<code>\pgfextractx</code> 21	scan commands:
<code>\pgfextracty</code> 21	<code>\scan_new:N</code> 5, 6
<code>\pgfgetlastxy</code> 21	scan internal commands:
<code>\pgfgettransform</code> 51	<code>\s__draw_mark</code> 5,
<code>\pgfgettransformentries</code> 51	842, 844, 849, 852, 859, 863, 1742, 1746
<code>\pgfinnerlinewidth</code> 50	<code>\s__draw_stop</code>
<code>\pgflowlevel</code> 51	. . 5, 835, 837, 842, 844, 849, 852,
<code>\pgflowlevelsynccm</code> 51	859, 863, 1742, 1746, 1782, 1793, 1802
<code>\pgfpatharcto</code> 6	seq commands:
<code>\pgfpatharctoprecomputed</code> 6	<code>\seq_new:N</code> 1837
<code>\pgfpathcosine</code> 6	<code>\seq_set_from_clist:Nn</code> 1826
<code>\pgfpathcurvebetweentime</code> 6	<code>\seq_set_map:NNn</code> 1827
<code>\pgfpathcurvebetweentimecontinue</code> 6	<code>\seq_use:Nn</code> 1832
<code>\pgfpathparabola</code> 6	skip commands:
<code>\pgfpathsine</code> 6	<code>\skip_horizontal:n</code> 1368, 1381
<code>\pgfpointadd</code> 21	str commands:
<code>\pgfpointborderellipse</code> 22	<code>\str_if_eq:nnTF</code>
<code>\pgfpointborderrectangle</code> 22 99, 117, 130, 148, 165, 176
<code>\pgfpointcylindrical</code> 21	<code>\str_if_eq_p:nn</code> 685
<code>\pgfpointdiff</code> 21	
<code>\pgfpointorigin</code> 21	T
<code>\pgfpointscale</code> 21	TeX and L ^A T _E X 2 _ε commands:
<code>\pgfpointspherical</code> 21	<code>\@expl@finalise@setup@</code> 1354
<code>\pgfqpoint</code> 22	<code>\savepos</code> 38
<code>\pgfqpointpolar</code> 22	tex commands:
<code>\pgfqpointscale</code> 22	<code>\tex_savepos:D</code> 1353, 1360
<code>\pgfqpointxy</code> 22	tl commands:
<code>\pgfqpointxyz</code> 22	<code>\tl_build_gbegin:N</code> 1484, 1809
<code>\pgfsetinnerlinewidth</code> 50	<code>\tl_build_gend:N</code> 1489, 1601
<code>\pgfsetinnerstrokecolor</code> 50	<code>\tl_build_get_intermediate:NN</code> 1477
<code>\pgftext</code> 36	<code>\tl_build_gput_right:Nn</code> 1473
<code>\pgftransformarcaxesattime</code> 51	<code>\tl_clear:N</code>
<code>\pgftransformarrow</code> 51	447, 1595, 1596, 1599, 1600, 1627, 1628
<code>\pgftransformationadjustments</code> 51	<code>\tl_if_blank:nTF</code> 678
<code>\pgftransformcurveattime</code> 51	<code>\tl_if_blank_p:n</code> 684
<code>\pgftransformlineattime</code> 51	<code>\tl_if_empty:NTF</code> 1616, 1646
<code>\pgfviewboxscope</code> 51	<code>\tl_if_empty_p:N</code> 1673, 1674
prg commands:	<code>\tl_new:N</code> 106, 197, 1415, 1416, 1469,
<code>\prg_do_nothing:</code> 1126, 1137, 1140	1470, 1580, 1581, 1582, 1587, 1588
property commands:	<code>\tl_put_right:Nn</code> 497, 501, 1622,
<code>\property_record:nn</code> 1356, 1363	1624, 1630, 1664, 1719, 1805, 1807
	<code>\tl_set:Nn</code> 107,
	121, 493, 1617, 1626, 1647, 1713, 1776
	<code>\tl_set_eq:NN</code> 1490

