

File I

Implementation

1 l3draw implementation

```

1 <*initex | package>
2 <@@=draw>
3 <*package>
4 \ProvidesExplPackage{l3draw}{2019-05-03}{}
5 {L3 Experimental core drawing support}
6 </package>
7 \RequirePackage { l3color }
8
9 Everything else is in the sub-files!
10 </initex | package>

```

2 l3draw-boxes implementation

```

9 <*initex | package>
10 <@@=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`

```

11 \box_new:N \l__draw_tmp_box

```

(End definition for \l__draw_tmp_box.)

`\draw_box_use:N`
`__draw_box_use:Nnnnn`

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

```

12 \cs_new_protected:Npn \draw_box_use:N #1
13 {
14   \__draw_box_use:Nnnnn #1
15   { Opt } { -\box_dp:N #1 } { \box_wd:N #1 } { \box_ht:N #1 }
16 }
17 \cs_new_protected:Npn \__draw_box_use:Nnnnn #1#2#3#4#5
18 {
19   \bool_if:NT \l_draw_bb_update_bool
20   {
21     \__draw_point_process:nn
22     { \__draw_path_update_limits:nn }
23     { \draw_point_transform:n { #2 , #3 } }
24     \__draw_point_process:nn
25     { \__draw_path_update_limits:nn }
26     { \draw_point_transform:n { #4 , #3 } }
27     \__draw_point_process:nn
28     { \__draw_path_update_limits:nn }
29     { \draw_point_transform:n { #4 , #5 } }

```

```

30     \__draw_point_process:nn
31     { \__draw_path_update_limits:nn }
32     { \draw_point_transform:n { #2 , #5 } }
33 }
34 \group_begin:
35 \hbox_set:Nn \l__draw_tmp_box
36 {
37     \use:x
38     {
39         \driver_draw_box_use:Nnnnn #1
40         { \fp_use:N \l__draw_matrix_a_fp }
41         { \fp_use:N \l__draw_matrix_b_fp }
42         { \fp_use:N \l__draw_matrix_c_fp }
43         { \fp_use:N \l__draw_matrix_d_fp }
44     }
45 }
46 \hbox_set:Nn \l__draw_tmp_box
47 {
48     \tex_kern:D \l__draw_xshift_dim
49     \box_move_up:nn { \l__draw_yshift_dim }
50     { \box_use_drop:N \l__draw_tmp_box }
51 }
52 \box_set_ht:Nn \l__draw_tmp_box { Opt }
53 \box_set_dp:Nn \l__draw_tmp_box { Opt }
54 \box_set_wd:Nn \l__draw_tmp_box { Opt }
55 \box_use_drop:N \l__draw_tmp_box
56 \group_end:
57 }

```

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

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

```

58 \cs_new_protected:Npn \draw_coffin_use:Nnn #1#2#3
59 {
60     \group_begin:
61     \hbox_set:Nn \l__draw_tmp_box
62     { \coffin_typeset:Nnnnn #1 {#2} {#3} { Opt } { Opt } }
63     \__draw_box_use:Nnnnn \l__draw_tmp_box
64     { \box_wd:N \l__draw_tmp_box - \coffin_wd:N #1 }
65     { -\box_dp:N \l__draw_tmp_box }
66     { \box_wd:N \l__draw_tmp_box }
67     { \box_ht:N \l__draw_tmp_box }
68     \group_end:
69 }

```

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

70 `</initex | package>`

3 l3draw-layers implementation

71 `<*initex | package>`

72 <@@=draw>

3.1 User interface

`\draw_layer_new:n`

```
73 \cs_new_protected:Npn \draw_layer_new:n #1
74 {
75   \str_if_eq:nnTF {#1} { main }
76   { \msg_error:nnn { draw } { main-reserved } }
77   {
78     \box_new:c { g__draw_layer_ #1 _box }
79     \box_new:c { l__draw_layer_ #1 _box }
80   }
81 }
```

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

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

```
82 \tl_new:N \l__draw_layer_tl
83 \tl_set:Nn \l__draw_layer_tl { main }
```

(End definition for `\l__draw_layer_tl`.)

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

```
84 \bool_new:N \l__draw_layer_close_bool
```

(End definition for `\l__draw_layer_close_bool`.)

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

```
\g__draw_layers_clist
85 \clist_new:N \l_draw_layers_clist
86 \clist_set:Nn \l_draw_layers_clist { main }
87 \clist_new:N \g__draw_layers_clist
```

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

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

```
88 \cs_new_protected:Npn \draw_layer_begin:n #1
89 {
90   \group_begin:
91   \box_if_exist:cTF { g__draw_layer_ #1 _box }
92   {
93     \str_if_eq:VnTF \l__draw_layer_tl {#1}
94     { \bool_set_false:N \l__draw_layer_close_bool }
95     {
96       \bool_set_true:N \l__draw_layer_close_bool
97       \tl_set:Nn \l__draw_layer_tl {#1}
98       \box_gset:wd:cn { g__draw_layer_ #1 _box } { Opt }
99       \hbox_gset:cw { g__draw_layer_ #1 _box }
100       \box_use_drop:c { g__draw_layer_ #1 _box }
101       \group_begin:
102     }
103     \draw_linewidth:n { \l_draw_default_linewidth_dim }
```

```

104     }
105     {
106         \str_if_eq:nnTF {#1} { main }
107         { \msg_error:nnn { draw } { unknown-layer } {#1} }
108         { \msg_error:nnn { draw } { main-layer } }
109     }
110 }
111 \cs_new_protected:Npn \draw_layer_end:
112 {
113     \bool_if:NT \l__draw_layer_close_bool
114     {
115         \group_end:
116         \hbox_gset_end:
117     }
118     \group_end:
119 }

```

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

3.2 Internal cross-links

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

```

120 \cs_new_protected:Npn \__draw_layers_insert:
121 {
122     \clist_map_inline:Nn \l_draw_layers_clist
123     {
124         \str_if_eq:nnTF {##1} { main }
125         {
126             \box_set_wd:Nn \l__draw_layer_main_box { Opt }
127             \box_use_drop:N \l__draw_layer_main_box
128         }
129         {
130             \driver_draw_scope_begin:
131             \box_gset_wd:cn { g__draw_layer_ ##1 _box } { Opt }
132             \box_use_drop:c { g__draw_layer_ ##1 _box }
133             \driver_draw_scope_end:
134         }
135     }
136 }

```

(End definition for `__draw_layers_insert:.`)

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

```

137 \cs_new_protected:Npn \__draw_layers_save:
138 {
139     \clist_map_inline:Nn \l_draw_layers_clist
140     {
141         \str_if_eq:nnF {##1} { main }
142         {
143             \box_set_eq:cc { l__draw_layer_ ##1 _box }
144             { g__draw_layer_ ##1 _box }
145         }
146     }

```

```

147 }
148 \cs_new_protected:Npn \__draw_layers_restore:
149 {
150   \clist_map_inline:Nn \l_draw_layers_clist
151   {
152     \str_if_eq:nnF {##1} { main }
153     {
154       \box_gset_eq:cc { g__draw_layer_ ##1 _box }
155       { l__draw_layer_ ##1 _box }
156     }
157   }
158 }

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

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

```

4 l3draw-paths implementation

```

171 <*initex | package>
172 <@@=draw>

```

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

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

```

\l__draw_path_tmp_tl Scratch space.
\l__draw_path_tmpa_fp
\l__draw_path_tmpb_fp
173 \tl_new:N \l__draw_path_tmp_tl
174 \fp_new:N \l__draw_path_tmpa_fp
175 \fp_new:N \l__draw_path_tmpb_fp

```

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

4.1 Tracking paths

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

`\g__draw_path_lasty_dim` 176 \dim_new:N \g__draw_path_lastx_dim
177 \dim_new:N \g__draw_path_lasty_dim

(End definition for \g__draw_path_lastx_dim and \g__draw_path_lasty_dim.)

`\g__draw_path_xmax_dim` The limiting size of a path.

`\g__draw_path_xmin_dim` 178 \dim_new:N \g__draw_path_xmax_dim
179 \dim_new:N \g__draw_path_xmin_dim
180 \dim_new:N \g__draw_path_ymax_dim
181 \dim_new:N \g__draw_path_ymin_dim

(End definition for \g__draw_path_xmax_dim and others.)

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

`__draw_path_reset_limits:` 182 \cs_new_protected:Npn __draw_path_update_limits:nn #1#2
183 {
184 \dim_gset:Nn \g__draw_path_xmax_dim
185 { \dim_max:nn \g__draw_path_xmax_dim {#1} }
186 \dim_gset:Nn \g__draw_path_xmin_dim
187 { \dim_min:nn \g__draw_path_xmin_dim {#1} }
188 \dim_gset:Nn \g__draw_path_ymax_dim
189 { \dim_max:nn \g__draw_path_ymax_dim {#2} }
190 \dim_gset:Nn \g__draw_path_ymin_dim
191 { \dim_min:nn \g__draw_path_ymin_dim {#2} }
192 \bool_if:NT \l_draw_bb_update_bool
193 {
194 \dim_gset:Nn \g__draw_xmax_dim
195 { \dim_max:nn \g__draw_xmax_dim {#1} }
196 \dim_gset:Nn \g__draw_xmin_dim
197 { \dim_min:nn \g__draw_xmin_dim {#1} }
198 \dim_gset:Nn \g__draw_ymax_dim
199 { \dim_max:nn \g__draw_ymax_dim {#2} }
200 \dim_gset:Nn \g__draw_ymin_dim
201 { \dim_min:nn \g__draw_ymin_dim {#2} }
202 }
203 }
204 \cs_new_protected:Npn __draw_path_reset_limits:
205 {
206 \dim_gset:Nn \g__draw_path_xmax_dim { -\c_max_dim }
207 \dim_gset:Nn \g__draw_path_xmin_dim { \c_max_dim }
208 \dim_gset:Nn \g__draw_path_ymax_dim { -\c_max_dim }
209 \dim_gset:Nn \g__draw_path_ymin_dim { \c_max_dim }
210 }

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

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

211 \cs_new_protected:Npn __draw_path_update_last:nn #1#2
212 {
213 \dim_gset:Nn \g__draw_path_lastx_dim {#1}
214 \dim_gset:Nn \g__draw_path_lasty_dim {#2}
215 }

(End definition for _draw_path_update_last:nn.)

4.2 Corner arcs

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

\l__draw_corner_xarc_dim The two arcs in use.

```

\l__draw_corner_yarc_dim 216 \dim_new:N \l__draw_corner_xarc_dim
217 \dim_new:N \l__draw_corner_yarc_dim

```

(End definition for \l__draw_corner_xarc_dim and \l__draw_corner_yarc_dim.)

\l__draw_corner_arc_bool A flag to speed up the repeated checks.

```
218 \bool_new:N \l__draw_corner_arc_bool
```

(End definition for \l__draw_corner_arc_bool.)

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

```

219 \cs_new_protected:Npn \draw_path_corner_arc:nn #1#2
220 {
221   \dim_set:Nn \l__draw_corner_xarc_dim {#1}
222   \dim_set:Nn \l__draw_corner_yarc_dim {#2}
223   \bool_lazy_and:nnTF
224     { \dim_compare_p:nNn \l__draw_corner_xarc_dim = { 0pt } }
225     { \dim_compare_p:nNn \l__draw_corner_yarc_dim = { 0pt } }
226     { \bool_set_false:N \l__draw_corner_arc_bool }
227     { \bool_set_true:N \l__draw_corner_arc_bool }
228 }

```

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

__draw_path_mark_corner: Mark up corners for arc post-processing.

```

229 \cs_new_protected:Npn \__draw_path_mark_corner:
230 {
231   \bool_if:NT \l__draw_corner_arc_bool
232   {
233     \__draw_softpath_roundpoint:VV
234       \l__draw_corner_xarc_dim
235       \l__draw_corner_yarc_dim
236   }
237 }

```

(End definition for __draw_path_mark_corner:.)

4.3 Basic path constructions

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

```

\draw_path_moveto:n
\draw_path_lineto:n
__draw_path_moveto:nn
__draw_path_lineto:nn
\draw_path_curveto:nnn
__draw_path_curveto:nnnnnn
238 \cs_new_protected:Npn \draw_path_moveto:n #1
239 {
240   \__draw_point_process:nn
241   { \__draw_path_moveto:nn }
242   { \draw_point_transform:n {#1} }
243 }
244 \cs_new_protected:Npn \__draw_path_moveto:nn #1#2
245 {
246   \__draw_path_update_limits:nn {#1} {#2}
247   \__draw_softpath_moveto:nn {#1} {#2}
248   \__draw_path_update_last:nn {#1} {#2}
249 }
250 \cs_new_protected:Npn \draw_path_lineto:n #1
251 {
252   \__draw_point_process:nn
253   { \__draw_path_lineto:nn }
254   { \draw_point_transform:n {#1} }
255 }
256 \cs_new_protected:Npn \__draw_path_lineto:nn #1#2
257 {
258   \__draw_path_mark_corner:
259   \__draw_path_update_limits:nn {#1} {#2}
260   \__draw_softpath_lineto:nn {#1} {#2}
261   \__draw_path_update_last:nn {#1} {#2}
262 }
263 \cs_new_protected:Npn \draw_path_curveto:nnn #1#2#3
264 {
265   \__draw_point_process:nnnn
266   {
267     \__draw_path_mark_corner:
268     \__draw_path_curveto:nnnnnn
269   }
270   { \draw_point_transform:n {#1} }
271   { \draw_point_transform:n {#2} }
272   { \draw_point_transform:n {#3} }
273 }
274 \cs_new_protected:Npn \__draw_path_curveto:nnnnnn #1#2#3#4#5#6
275 {
276   \__draw_path_update_limits:nn {#1} {#2}
277   \__draw_path_update_limits:nn {#3} {#4}
278   \__draw_path_update_limits:nn {#5} {#6}
279   \__draw_softpath_curveto:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
280   \__draw_path_update_last:nn {#5} {#6}
281 }

```

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

\draw_path_close: A simple wrapper.

```

282 \cs_new_protected:Npn \draw_path_close:
283 {

```



```

284     \__draw_path_mark_corner:
285     \__draw_softpath_closepath:
286 }

```

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

4.4 Canvas path constructions

```

\draw_path_canvas_moveto:n Operations with no application of the transformation matrix.
\draw_path_canvas_lineto:n
    \draw_path_canvas_curveto:nnn
287 \cs_new_protected:Npn \draw_path_canvas_moveto:n #1
288 { \__draw_point_process:nn { \__draw_path_moveto:nn } {#1} }
289 \cs_new_protected:Npn \draw_path_canvas_lineto:n #1
290 { \__draw_point_process:nn { \__draw_path_lineto:nn } {#1} }
291 \cs_new_protected:Npn \draw_path_canvas_curveto:nnn #1#2#3
292 {
293     \__draw_point_process:nnnn
294     {
295         \__draw_path_mark_corner:
296         \__draw_path_curveto:nnnnnn
297     }
298     {#1} {#2} {#3}
299 }

```

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

4.5 Computed curves

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

```

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

```

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

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

and

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

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

```

300 \cs_new_protected:Npn \draw_path_curveto:nn #1#2
301 {
302     \__draw_point_process:nnn
303     { \__draw_path_curveto:nnnn }
304     { \draw_point_transform:n {#1} }
305     { \draw_point_transform:n {#2} }
306 }
307 \cs_new_protected:Npn \__draw_path_curveto:nnnn #1#2#3#4
308 {
309     \fp_set:Nn \l__draw_path_tmpa_fp { \c__draw_path_curveto_b_fp * #1 }
310     \fp_set:Nn \l__draw_path_tmpb_fp { \c__draw_path_curveto_b_fp * #2 }
311     \use:x
312     {
313         \__draw_path_mark_corner:
314         \__draw_path_curveto:nnnnnn

```

```

315         {
316             \fp_to_dim:n
317             {
318                 \c__draw_path_curveto_a_fp * \g__draw_path_lastx_dim
319                 + \l__draw_path_tmpa_fp
320             }
321         }
322         {
323             \fp_to_dim:n
324             {
325                 \c__draw_path_curveto_a_fp * \g__draw_path_lasty_dim
326                 + \l__draw_path_tmpb_fp
327             }
328         }
329         {
330             \fp_to_dim:n
331             { \c__draw_path_curveto_a_fp * #3 + \l__draw_path_tmpa_fp }
332         }
333         {
334             \fp_to_dim:n
335             { \c__draw_path_curveto_a_fp * #4 + \l__draw_path_tmpb_fp }
336         }
337         {#3}
338         {#4}
339     }
340 }
341 \fp_const:Nn \c__draw_path_curveto_a_fp { 1 / 3 }
342 \fp_const:Nn \c__draw_path_curveto_b_fp { 2 / 3 }

```

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

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

`\draw_path_arc:nnnn`

`__draw_path_arc:nnnn`

`__draw_path_arc:nnNnn`

`__draw_path_arc_auxi:nnnnNnn`

`__draw_path_arc_auxi:fnnnNnn`

`__draw_path_arc_auxi:fnfnNnn`

`__draw_path_arc_auxii:nnnNnnnn`

`__draw_path_arc_auxiii:nn`

`__draw_path_arc_auxiv:nnnn`

`__draw_path_arc_auxv:nn`

`__draw_path_arc_auxvi:nn`

`__draw_path_arc_add:nnnn`

`\l__draw_path_arc_delta_fp`

`\l__draw_path_arc_start_fp`

`\c__draw_path_arc_90_fp`

`\c__draw_path_arc_60_fp`

```

343 \cs_new_protected:Npn \draw_path_arc:nnn #1#2#3
344 { \draw_path_arc:nnnn {#1} {#2} {#3} {#3} }
345 \cs_new_protected:Npn \draw_path_arc:nnnn #1#2#3#4
346 {
347     \use:x
348     {
349         \__draw_path_arc:nnnn
350         { \fp_eval:n {#1} }
351         { \fp_eval:n {#2} }
352         { \fp_to_dim:n {#3} }
353         { \fp_to_dim:n {#4} }
354     }
355 }
356 \cs_new_protected:Npn \__draw_path_arc:nnnn #1#2#3#4
357 {
358     \fp_compare:nNnTF {#1} > {#2}
359     { \__draw_path_arc:nnNnn {#1} {#2} - {#3} {#4} }
360     { \__draw_path_arc:nnNnn {#1} {#2} + {#3} {#4} }
361 }
362 \cs_new_protected:Npn \__draw_path_arc:nnNnn #1#2#3#4#5

```

```

363 {
364   \fp_set:Nn \l__draw_path_arc_start_fp {#1}
365   \fp_set:Nn \l__draw_path_arc_delta_fp { abs( #1 - #2 ) }
366   \fp_while_do:nNnn { \l__draw_path_arc_delta_fp } > { 90 }
367   {
368     \fp_compare:nNnTF \l__draw_path_arc_delta_fp > { 115 }
369     {
370       \__draw_path_arc_auxi:ffnnNnn
371       { \fp_to_decimal:N \l__draw_path_arc_start_fp }
372       { \fp_eval:n { \l__draw_path_arc_start_fp #3 90 } }
373       { 90 } {#2}
374       #3 {#4} {#5}
375     }
376     {
377       \__draw_path_arc_auxi:ffnnNnn
378       { \fp_to_decimal:N \l__draw_path_arc_start_fp }
379       { \fp_eval:n { \l__draw_path_arc_start_fp #3 60 } }
380       { 60 } {#2}
381       #3 {#4} {#5}
382     }
383   }
384   \__draw_path_mark_corner:
385   \__draw_path_arc_auxi:fnfnNnn
386   { \fp_to_decimal:N \l__draw_path_arc_start_fp }
387   {#2}
388   { \fp_eval:n { abs( \l__draw_path_arc_start_fp - #2 ) } }
389   {#2}
390   #3 {#4} {#5}
391 }

```

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.

```

392 \cs_new_protected:Npn \__draw_path_arc_auxi:nnnnNnn #1#2#3#4#5#6#7
393 {
394   \use:x
395   {
396     \__draw_path_arc_auxii:nnnNnnnn
397     {#1} {#2} {#4} #5 {#6} {#7}
398     {
399       \fp_to_dim:n
400       {
401         \cs_if_exist_use:cF
402         { c__draw_path_arc_ #3 _fp }
403         { 4/3 * tand( 0.25 * #3 ) }
404         * #6
405       }
406     }
407     {
408       \fp_to_dim:n
409       {
410         \cs_if_exist_use:cF
411         { c__draw_path_arc_ #3 _fp }

```

```

412         { 4/3 * tand( 0.25 * #3 ) }
413         * #7
414     }
415 }
416 }
417 }
418 \cs_generate_variant:Nn \__draw_path_arc_auxi:nnnnNnn { fnf , ff }

```

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

```

419 \cs_new_protected:Npn \__draw_path_arc_auxii:nnnnnnn #1#2#3#4#5#6#7#8
420 {
421   \tl_clear:N \l__draw_path_tmp_tl
422   \__draw_point_process:nn
423   { \__draw_path_arc_auxiii:nn }
424   {
425     \__draw_point_transform_noshift:n
426     { \draw_point_polar:nnn { #1 #4 90 } {#7} {#8} }
427   }
428   \__draw_point_process:nnn
429   { \__draw_path_arc_auxiv:nnnn }
430   {
431     \draw_point_transform:n
432     { \draw_point_polar:nnn {#1} {#5} {#6} }
433   }
434   {
435     \draw_point_transform:n
436     { \draw_point_polar:nnn {#2} {#5} {#6} }
437   }
438   \__draw_point_process:nn
439   { \__draw_path_arc_auxv:nn }
440   {
441     \__draw_point_transform_noshift:n
442     { \draw_point_polar:nnn { #2 #4 -90 } {#7} {#8} }
443   }
444   \exp_after:wN \__draw_path_curveto:nnnnnnn \l__draw_path_tmp_tl
445   \fp_set:Nn \l__draw_path_arc_delta_fp { abs ( #2 - #3 ) }
446   \fp_set:Nn \l__draw_path_arc_start_fp {#2}
447 }

```

The first control point.

```

448 \cs_new_protected:Npn \__draw_path_arc_auxiii:nn #1#2
449 {
450   \__draw_path_arc_aux_add:nn
451   { \g__draw_path_lastx_dim + #1 }
452   { \g__draw_path_lasty_dim + #2 }
453 }

```

The end point: simple arithmetic.

```

454 \cs_new_protected:Npn \__draw_path_arc_auxiv:nnnn #1#2#3#4
455 {
456   \__draw_path_arc_aux_add:nn

```

```

457     { \g__draw_path_lastx_dim - #1 + #3 }
458     { \g__draw_path_lasty_dim - #2 + #4 }
459 }

```

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

```

460 \cs_new_protected:Npn \__draw_path_arc_auxv:nn #1#2
461 {
462     \exp_after:wN \__draw_path_arc_auxvi:nn
463     \l__draw_path_tmp_tl {#1} {#2}
464 }
465 \cs_new_protected:Npn \__draw_path_arc_auxvi:nn #1#2#3#4#5#6
466 {
467     \tl_set:Nn \l__draw_path_tmp_tl { {#1} {#2} }
468     \__draw_path_arc_aux_add:nn
469     { #5 + #3 }
470     { #6 + #4 }
471     \tl_put_right:Nn \l__draw_path_tmp_tl { {#3} {#4} }
472 }
473 \cs_new_protected:Npn \__draw_path_arc_aux_add:nn #1#2
474 {
475     \tl_put_right:Nx \l__draw_path_tmp_tl
476     { { \fp_to_dim:n {#1} } { \fp_to_dim:n {#2} } }
477 }
478 \fp_new:N \l__draw_path_arc_delta_fp
479 \fp_new:N \l__draw_path_arc_start_fp
480 \fp_const:cn { c__draw_path_arc_90_fp } { 4/3 * (sqrt(2) - 1) }
481 \fp_const:cn { c__draw_path_arc_60_fp } { 4/3 * tand(15) }

```

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

`\draw_path_arc_axes:nnnn` A simple wrapper.

```

482 \cs_new_protected:Npn \draw_path_arc_axes:nnnn #1#2#3#4
483 {
484     \draw_transform_triangle:nnn { 0cm , 0cm } {#3} {#4}
485     \draw_path_arc:nnn {#1} {#2} { 1pt }
486 }

```

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

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

```

\__draw_path_ellipse:nnnnnnn
  \_draw_path_ellipse_arci:nnnnnnn
  \_draw_path_ellipse_arcii:nnnnnnn
  \_draw_path_ellipse_arciiii:nnnnnnn
  \_draw_path_ellipse_arciv:nnnnnnn
\c__draw_path_ellipse_fp
487 \cs_new_protected:Npn \draw_path_ellipse:nnn #1#2#3
488 {
489     \__draw_point_process:nnnn
490     { \__draw_path_ellipse:nnnnnnn }
491     { \draw_point_transform:n {#1} }
492     { \__draw_point_transform_noshift:n {#2} }
493     { \__draw_point_transform_noshift:n {#3} }
494 }
495 \cs_new_protected:Npn \__draw_path_ellipse:nnnnnnn #1#2#3#4#5#6
496 {
497     \use:x
498     {

```

```

499     \__draw_path_moveto:nn
500     { \fp_to_dim:n { #1 + #3 } } { \fp_to_dim:n { #2 + #4 } }
501     \__draw_path_ellipse_arci:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
502     \__draw_path_ellipse_arci:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
503     \__draw_path_ellipse_arci:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
504     \__draw_path_ellipse_arci:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
505 }
506 \__draw_softp_path_closepath:
507 \__draw_path_moveto:nn {#1} {#2}
508 }
509 \cs_new:Npn \__draw_path_ellipse_arci:nnnnnn #1#2#3#4#5#6
510 {
511     \__draw_path_curveto:nnnnnn
512     { \fp_to_dim:n { #1 + #3 + #5 * \c__draw_path_ellipse_fp } }
513     { \fp_to_dim:n { #2 + #4 + #6 * \c__draw_path_ellipse_fp } }
514     { \fp_to_dim:n { #1 + #3 * \c__draw_path_ellipse_fp + #5 } }
515     { \fp_to_dim:n { #2 + #4 * \c__draw_path_ellipse_fp + #6 } }
516     { \fp_to_dim:n { #1 + #5 } }
517     { \fp_to_dim:n { #2 + #6 } }
518 }
519 \cs_new:Npn \__draw_path_ellipse_arci:nnnnnn #1#2#3#4#5#6
520 {
521     \__draw_path_curveto:nnnnnn
522     { \fp_to_dim:n { #1 - #3 * \c__draw_path_ellipse_fp + #5 } }
523     { \fp_to_dim:n { #2 - #4 * \c__draw_path_ellipse_fp + #6 } }
524     { \fp_to_dim:n { #1 - #3 + #5 * \c__draw_path_ellipse_fp } }
525     { \fp_to_dim:n { #2 - #4 + #6 * \c__draw_path_ellipse_fp } }
526     { \fp_to_dim:n { #1 - #3 } }
527     { \fp_to_dim:n { #2 - #4 } }
528 }
529 \cs_new:Npn \__draw_path_ellipse_arci:nnnnnn #1#2#3#4#5#6
530 {
531     \__draw_path_curveto:nnnnnn
532     { \fp_to_dim:n { #1 - #3 - #5 * \c__draw_path_ellipse_fp } }
533     { \fp_to_dim:n { #2 - #4 - #6 * \c__draw_path_ellipse_fp } }
534     { \fp_to_dim:n { #1 - #3 * \c__draw_path_ellipse_fp - #5 } }
535     { \fp_to_dim:n { #2 - #4 * \c__draw_path_ellipse_fp - #6 } }
536     { \fp_to_dim:n { #1 - #5 } }
537     { \fp_to_dim:n { #2 - #6 } }
538 }
539 \cs_new:Npn \__draw_path_ellipse_arci:nnnnnn #1#2#3#4#5#6
540 {
541     \__draw_path_curveto:nnnnnn
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 + #3 - #5 * \c__draw_path_ellipse_fp } }
545     { \fp_to_dim:n { #2 + #4 - #6 * \c__draw_path_ellipse_fp } }
546     { \fp_to_dim:n { #1 + #3 } }
547     { \fp_to_dim:n { #2 + #4 } }
548 }
549 \fp_const:Nn \c__draw_path_ellipse_fp { \fp_use:c { c__draw_path_arc_90_fp } }

```

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

`\draw_path_circle:nn` A shortcut.

```
550 \cs_new_protected:Npn \draw_path_circle:nn #1#2
551 { \draw_path_ellipse:nnn {#1} { #2 , 0pt } { 0pt , #2 } }
```

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

4.6 Rectangles

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

`__draw_path_rectangle:nnnn`

`__draw_path_rectangle_rounded:nnnn`

```
552 \cs_new_protected:Npn \draw_path_rectangle:nn #1#2
553 {
554   \__draw_point_process:nnn
555   {
556     \bool_lazy_or:nnTF
557     { \l__draw_corner_arc_bool }
558     { \l__draw_matrix_active_bool }
559     { \__draw_path_rectangle_rounded:nnnn }
560     { \__draw_path_rectangle:nnnn }
561   }
562   { \draw_point_transform:n {#1} }
563   {#2}
564 }
565 \cs_new_protected:Npn \__draw_path_rectangle:nnnn #1#2#3#4
566 {
567   \__draw_path_update_limits:nn {#1} {#2}
568   \__draw_path_update_limits:nn { #1 + #3 } { #2 + #4 }
569   \__draw_softpath_rectangle:nnnn {#1} {#2} {#3} {#4}
570   \__draw_path_update_last:nn {#1} {#2}
571 }
572 \cs_new_protected:Npn \__draw_path_rectangle_rounded:nnnn #1#2#3#4
573 {
574   \draw_path_moveto:n { #1 + #3 , #2 + #4 }
575   \draw_path_lineto:n { #1 , #2 + #4 }
576   \draw_path_lineto:n { #1 , #2 }
577   \draw_path_lineto:n { #1 + #3 , #2 }
578   \draw_path_close:
579   \draw_path_moveto:n { #1 , #2 }
580 }
```

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

`\draw_path_rectangle_corners:nn` Another shortcut wrapper.

`__draw_path_rectangle_corners:nnnn`

```
581 \cs_new_protected:Npn \draw_path_rectangle_corners:nn #1#2
582 {
583   \__draw_point_process:nnn
584   { \__draw_path_rectangle_corners:nnnn {#1} }
585   {#1} {#2}
586 }
587 \cs_new_protected:Npn \__draw_path_rectangle_corners:nnnn #1#2#3#4#5
588 { \draw_path_rectangle:nn {#1} { #4 - #2 , #5 - #3 } }
```

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

4.7 Grids

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

```

\draw_path_grid:nnnn
  \_draw_path_grid_auxi:nnnnnn
  \_draw_path_grid_auxii:ffnnnn
  \_draw_path_grid_auxiii:nnnnnn
  \_draw_path_grid_auxiiii:ffnnnn
  \_draw_path_grid_auxiv:nnnnnnnn
  \_draw_path_grid_auxiv:ffnnnnnn

589 \cs_new_protected:Npn \draw_path_grid:nnnn #1#2#3#4
590 {
591   \__draw_point_process:nnn
592   {
593     \_draw_path_grid_auxi:ffnnnn
594     { \dim_eval:n { \dim_abs:n {#1} } }
595     { \dim_eval:n { \dim_abs:n {#2} } }
596   }
597   {#3} {#4}
598 }
599 \cs_new_protected:Npn \__draw_path_grid_auxi:nnnnnn #1#2#3#4#5#6
600 {
601   \dim_compare:nNnTF {#3} > {#5}
602   { \__draw_path_grid_auxii:nnnnnn {#1} {#2} {#5} {#4} {#3} {#6} }
603   { \__draw_path_grid_auxii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6} }
604 }
605 \cs_generate_variant:Nn \__draw_path_grid_auxi:nnnnnn { ff }
606 \cs_new_protected:Npn \__draw_path_grid_auxii:nnnnnn #1#2#3#4#5#6
607 {
608   \dim_compare:nNnTF {#4} > {#6}
609   { \__draw_path_grid_auxiii:nnnnnn {#1} {#2} {#3} {#6} {#5} {#4} }
610   { \__draw_path_grid_auxiii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6} }
611 }
612 \cs_new_protected:Npn \__draw_path_grid_auxiii:nnnnnn #1#2#3#4#5#6
613 {
614   \__draw_path_grid_auxiv:ffnnnnnn
615   { \fp_to_dim:n { #1 * trunc(#3/(#1)) } }
616   { \fp_to_dim:n { #2 * trunc(#4/(#2)) } }
617   {#1} {#2} {#3} {#4} {#5} {#6}
618 }
619 \cs_new_protected:Npn \__draw_path_grid_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
620 {
621   \dim_step_inline:nnnn
622   {#1}
623   {#3}
624   {#7}
625   {
626     \draw_path_moveto:n { ##1 , #6 }
627     \draw_path_lineto:n { ##1 , #8 }
628   }
629   \dim_step_inline:nnnn
630   {#2}
631   {#4}
632   {#8}
633   {
634     \draw_path_moveto:n { #5 , ##1 }
635     \draw_path_lineto:n { #7 , ##1 }
636   }
637 }
638 \cs_generate_variant:Nn \__draw_path_grid_auxiv:nnnnnnnn { ff }

```


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

4.8 Using paths

Actions to pass to the driver.

```
639 \bool_new:N \l__draw_path_use_clip_bool
640 \bool_new:N \l__draw_path_use_fill_bool
641 \bool_new:N \l__draw_path_use_stroke_bool
```

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

Actions handled at the macro layer.

```
642 \bool_new:N \l__draw_path_use_bb_bool
643 \bool_new:N \l__draw_path_use_clear_bool
```

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

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

```
644 \cs_new_protected:Npn \draw_path_use:n #1
645 {
646   \tl_if_blank:nF {#1}
647     { \__draw_path_use:n {#1} }
648   }
649 \cs_new_protected:Npn \draw_path_use_clear:n #1
650 {
651   \bool_lazy_or:nnTF
652     { \tl_if_blank_p:n {#1} }
653     { \str_if_eq_p:nn {#1} { clear } }
654     {
655       \__draw_softpath_clear:
656       \__draw_path_reset_limits:
657     }
658   { \__draw_path_use:n { #1 , clear } }
659 }
```

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).

```
660 \cs_new_protected:Npn \__draw_path_use:n #1
661 {
662   \bool_set_false:N \l__draw_path_use_clip_bool
663   \bool_set_false:N \l__draw_path_use_fill_bool
664   \bool_set_false:N \l__draw_path_use_stroke_bool
665   \clist_map_inline:nn {#1}
666   {
667     \cs_if_exist:CTF { l__draw_path_use_ ##1 _ bool }
668     { \bool_set_true:c { l__draw_path_use_ ##1 _ bool } }
669     {
670       \cs_if_exist_use:cF { __draw_path_use_action_ ##1 : }
671       { \msg_error:nnn { draw } { invalid-path-action } {##1} }
672     }
673   }
```

```

673     }
674     \__draw_softpath_round_corners:
675     \bool_lazy_and:nnT
676       { \l_draw_bb_update_bool }
677       { \l__draw_path_use_stroke_bool }
678       { \__draw_path_use_stroke_bb: }
679     \__draw_softpath_use:
680     \bool_if:NT \l__draw_path_use_clip_bool
681     {
682       \driver_draw_clip:
683       \bool_lazy_or:nnF
684         { \l__draw_path_use_fill_bool }
685         { \l__draw_path_use_stroke_bool }
686         { \driver_draw_discardpath: }
687     }
688     \bool_lazy_or:nnT
689       { \l__draw_path_use_fill_bool }
690       { \l__draw_path_use_stroke_bool }
691     {
692       \use:c
693       {
694         driver_draw_
695         \bool_if:NT \l__draw_path_use_fill_bool { fill }
696         \bool_if:NT \l__draw_path_use_stroke_bool { stroke }
697         :
698       }
699     }
700     \bool_if:NT \l__draw_path_use_clear_bool
701     { \__draw_softpath_clear: }
702   }
703   \cs_new_protected:Npn \__draw_path_use_action_draw:
704   {
705     \bool_set_true:N \l__draw_path_use_stroke_bool
706   }
707   \cs_new_protected:Npn \__draw_path_use_action_fillstroke:
708   {
709     \bool_set_true:N \l__draw_path_use_fill_bool
710     \bool_set_true:N \l__draw_path_use_stroke_bool
711   }

```

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.

```

712   \cs_new_protected:Npn \__draw_path_use_stroke_bb:
713   {
714     \__draw_path_use_stroke_bb_aux:NnN x { max } +
715     \__draw_path_use_stroke_bb_aux:NnN y { max } +
716     \__draw_path_use_stroke_bb_aux:NnN x { min } -
717     \__draw_path_use_stroke_bb_aux:NnN y { min } -
718   }
719   \cs_new_protected:Npn \__draw_path_use_stroke_bb_aux:NnN #1#2#3
720   {
721     \dim_compare:nNnF { \dim_use:c { g__draw_ #1#2 _dim } } = { #3 -\c_max_dim }
722     {
723       \dim_gset:cn { g__draw_ #1#2 _dim }

```

```

724         {
725             \use:c { dim_ #2 :nn }
726             { \dim_use:c { g__draw_ #1#2 _dim } }
727             {
728                 \dim_use:c { g__draw_path_ #1#2 _dim }
729                 #3 0.5 \g__draw_linewidth_dim
730             }
731         }
732     }
733 }

```

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

4.9 Scoping paths

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

```

\l__draw_path_lastx_dim 734 \dim_new:N \l__draw_path_lastx_dim
\l__draw_path_lasty_dim 735 \dim_new:N \l__draw_path_lasty_dim
\l__draw_path_xmax_dim 736 \dim_new:N \l__draw_path_xmax_dim
\l__draw_path_xmin_dim 737 \dim_new:N \l__draw_path_xmin_dim
\l__draw_path_ymax_dim 738 \dim_new:N \l__draw_path_ymax_dim
\l__draw_path_ymin_dim 739 \dim_new:N \l__draw_path_ymin_dim
\l__draw_softpath_corners_bool 740 \dim_new:N \l__draw_softpath_lastx_dim
741 \dim_new:N \l__draw_softpath_lasty_dim
742 \bool_new:N \l__draw_softpath_corners_bool

```

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

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

```

743 \cs_new_protected:Npn \draw_path_scope_begin:
744 {
745     \group_begin:
746     \dim_set_eq:NN \l__draw_path_lastx_dim \g__draw_path_lastx_dim
747     \dim_set_eq:NN \l__draw_path_lasty_dim \g__draw_path_lasty_dim
748     \dim_set_eq:NN \l__draw_path_xmax_dim \g__draw_path_xmax_dim
749     \dim_set_eq:NN \l__draw_path_xmin_dim \g__draw_path_xmin_dim
750     \dim_set_eq:NN \l__draw_path_ymax_dim \g__draw_path_ymax_dim
751     \dim_set_eq:NN \l__draw_path_ymin_dim \g__draw_path_ymin_dim
752     \dim_set_eq:NN \l__draw_softpath_lastx_dim \g__draw_softpath_lastx_dim
753     \dim_set_eq:NN \l__draw_softpath_lasty_dim \g__draw_softpath_lasty_dim
754     \__draw_path_reset_limits:
755     \tl_build_get:NN \g__draw_softpath_main_tl \l__draw_softpath_main_tl
756     \bool_set_eq:NN
757         \l__draw_softpath_corners_bool
758         \g__draw_softpath_corners_bool
759     \__draw_softpath_clear:
760 }
761 \cs_new_protected:Npn \draw_path_scope_end:
762 {
763     \__draw_softpath_clear:
764     \bool_gset_eq:NN

```

```

765         \g__draw_softpath_corners_bool
766         \l__draw_softpath_corners_bool
767         \__draw_softpath_add:o \l__draw_softpath_main_tl
768         \dim_gset_eq:NN \g__draw_softpath_lastx_dim \l__draw_softpath_lastx_dim
769         \dim_gset_eq:NN \g__draw_softpath_lasty_dim \l__draw_softpath_lasty_dim
770         \dim_gset_eq:NN \g__draw_path_xmax_dim \l__draw_path_xmax_dim
771         \dim_gset_eq:NN \g__draw_path_xmin_dim \l__draw_path_xmin_dim
772         \dim_gset_eq:NN \g__draw_path_ymax_dim \l__draw_path_ymax_dim
773         \dim_gset_eq:NN \g__draw_path_ymin_dim \l__draw_path_ymin_dim
774         \dim_gset_eq:NN \g__draw_path_lastx_dim \l__draw_path_lastx_dim
775         \dim_gset_eq:NN \g__draw_path_lasty_dim \l__draw_path_lasty_dim
776     \group_end:
777 }

```

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

```

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

```

5 l3draw-points implementation

```

785 \<*initex | package>
786 \<@@=draw>

```

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

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

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

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

5.1 Support functions

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

```

\__draw_point_process:nn      Execute whatever code is passed to extract the  $x$  and  $y$  co-ordinates. The first argument
    \_draw_point_process_auxi:nn here should itself absorb two arguments. There is also a version to deal with two co-
    \_draw_point_process_auxii:nw ordinates: common enough to justify a separate function.
\__draw_point_process:nnnn
    \_draw_point_process_auxiii:nnnn
    \_draw_point_process_auxiv:nw
\__draw_point_process:nnnnn
    \_draw_point_process_auxv:nnnnn
    \_draw_point_process_auxvi:nw
\__draw_point_process:nnnnnn
    \_draw_point_process_auxvii:nnnnnn
    \_draw_point_process_auxviii:nw
787 \cs_new:Npn \__draw_point_process:nn #1#2
788 {
789     \exp_args:Nf \__draw_point_process_auxi:nn
790     { \__draw_point_to_dim:n {#2} }
791     {#1}
792 }
793 \cs_new:Npn \__draw_point_process_auxi:nn #1#2
794 { \__draw_point_process_auxii:nw {#2} #1 \q_stop }
795 \cs_new:Npn \__draw_point_process_auxii:nw #1 #2 , #3 \q_stop
796 { #1 {#2} {#3} }
797 \cs_new:Npn \__draw_point_process:nnn #1#2#3
798 {
799     \exp_args:Nff \__draw_point_process_auxiii:nnnn
800     { \__draw_point_to_dim:n {#2} }
801     { \__draw_point_to_dim:n {#3} }
802     {#1}
803 }
804 \cs_new:Npn \__draw_point_process_auxiii:nnnn #1#2#3
805 { \__draw_point_process_auxiv:nw {#3} #1 \q_mark #2 \q_stop }
806 \cs_new:Npn \__draw_point_process_auxiv:nw #1 #2 , #3 \q_mark #4 , #5 \q_stop
807 { #1 {#2} {#3} {#4} {#5} }
808 \cs_new:Npn \__draw_point_process:nnnn #1#2#3#4
809 {
810     \exp_args:Nfff \__draw_point_process_auxv:nnnnn
811     { \__draw_point_to_dim:n {#2} }
812     { \__draw_point_to_dim:n {#3} }
813     { \__draw_point_to_dim:n {#4} }
814     {#1}
815 }
816 \cs_new:Npn \__draw_point_process_auxv:nnnnn #1#2#3#4
817 { \__draw_point_process_auxvi:nw {#4} #1 \q_mark #2 \q_mark #3 \q_stop }
818 \cs_new:Npn \__draw_point_process_auxvi:nw
819     #1 #2 , #3 \q_mark #4 , #5 \q_mark #6 , #7 \q_stop
820 { #1 {#2} {#3} {#4} {#5} {#6} {#7} }
821 \cs_new:Npn \__draw_point_process:nnnnn #1#2#3#4#5
822 {
823     \exp_args:Nffff \__draw_point_process_auxvii:nnnnnn
824     { \__draw_point_to_dim:n {#2} }
825     { \__draw_point_to_dim:n {#3} }
826     { \__draw_point_to_dim:n {#4} }
827     { \__draw_point_to_dim:n {#5} }
828     {#1}
829 }
830 \cs_new:Npn \__draw_point_process_auxvii:nnnnnn #1#2#3#4#5
831 {
832     \__draw_point_process_auxviii:nw
833     {#5} #1 \q_mark #2 \q_mark #3 \q_mark #4 \q_stop
834 }
835 \cs_new:Npn \__draw_point_process_auxviii:nw

```

```

836 #1 #2 , #3 \q_mark #4 , #5 \q_mark #6 , #7 \q_mark #8 , #9 \q_stop
837 { #1 {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9} }

```

(End definition for _draw_point_process:nn and others.)

```

\_draw_point_to_dim:n      Co-ordinates are always returned as two dimensions.
\_draw_point_to_dim_aux:n  838 \cs_new:Npn \_draw_point_to_dim:n #1
\_draw_point_to_dim_aux:f  839 { \_draw_point_to_dim_aux:f { \fp_eval:n {#1} } }
\_draw_point_to_dim_aux:w  840 \cs_new:Npn \_draw_point_to_dim_aux:n #1
                           841 { \_draw_point_to_dim_aux:w #1 }
                           842 \cs_generate_variant:Nn \_draw_point_to_dim_aux:n { f }
                           843 \cs_new:Npn \_draw_point_to_dim_aux:w ( #1 , ~ #2 ) { #1pt , #2pt }

```

5.2 Polar co-ordinates

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

```

\draw_point_polar:nn      844 \cs_new:Npn \draw_point_polar:nn #1#2
\draw_point_polar:nnnn    845 { \draw_point_polar:nnnn {#1} {#2} {#2} }
\_draw_draw_polar:nnnn    846 \cs_new:Npn \draw_point_polar:nnnn #1#2#3
\_draw_draw_polar:fnn     847 { \_draw_draw_polar:fnn { \fp_eval:n {#1} } {#2} {#3} }
                           848 \cs_new:Npn \_draw_draw_polar:nnnn #1#2#3
                           849 { \_draw_point_to_dim:n { cosd(#1) * (#2) , sind(#1) * (#3) } }
                           850 \cs_generate_variant:Nn \_draw_draw_polar:nnnn { f }

```

5.3 Point expression arithmetic

These functions all take point expressions as arguments.

Only a single point expression so the expansion is done here. The outcome is the normalised vector from (0,0) in the direction of the point, *i.e.*

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

```

851 \cs_new:Npn \draw_point_unit_vector:n #1
852 { \_draw_point_process:nn { \_draw_point_unit_vector:nn } {#1} }
853 \cs_new:Npn \_draw_point_unit_vector:nn #1#2
854 {
855   \_draw_point_to_dim:n
856   { ( #1 , #2 ) / (sqrt(#1 * #1 + #2 * #2)) }
857 }

```

5.4 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

```

\draw_point_intersect_lines:nnnnn
\_draw_point_intersect_lines:nnnnnnn
\_draw_point_intersect_lines:nnnnnnnn
\_draw_point_intersect_lines_aux:nnnnnn
\_draw_point_intersect_lines_aux:ffffff

```

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

and

$$P_y = \frac{(x_1y_2 - y_1x_2)(y_3 - y_5) - (x_3y_4 - y_3x_4)(y_1 - y_2)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}$$

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.)

```

858 \cs_new:Npn \draw_point_intersect_lines:nnnn #1#2#3#4
859 {
860   \__draw_point_process:nnnnn
861   { \__draw_point_intersect_lines:nnnnnnnn }
862   {#1} {#2} {#3} {#4}
863 }

```

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

```

#1 x_1
#2 y_1
#3 x_2
#4 y_2
#5 x_3
#6 y_3
#7 x_4
#8 y_4

```

so now just have to do all of the calculation.

```

864 \cs_new:Npn \__draw_point_intersect_lines:nnnnnnnn #1#2#3#4#5#6#7#8
865 {
866   \__draw_point_intersect_lines_aux:ffffff
867   { \fp_eval:n { #1 * #4 - #2 * #3 } }
868   { \fp_eval:n { #5 * #8 - #6 * #7 } }
869   { \fp_eval:n { #1 - #3 } }
870   { \fp_eval:n { #5 - #7 } }
871   { \fp_eval:n { #2 - #4 } }
872   { \fp_eval:n { #6 - #8 } }
873 }
874 \cs_new:Npn \__draw_point_intersect_lines_aux:nnnnnn #1#2#3#4#5#6
875 {
876   \__draw_point_to_dim:n
877   {
878     ( #2 * #3 - #1 * #4 , #2 * #5 - #1 * #6 )
879     / ( #4 * #5 - #6 * #3 )
880   }
881 }
882 \cs_generate_variant:Nn \__draw_point_intersect_lines_aux:nnnnnn { fffffff }

```

```

\draw_point_intersect_circles:nnnnn
\_draw_point_intersect_circles_auxi:nnnnnnnn
\_draw_point_intersect_circles_auxii:nnnnnnnn
\_draw_point_intersect_circles_auxiii:ffnnnnnn
draw_point_intersect_circles_auxiii:nnnnnnnn
draw_point_intersect_circles_auxiii:ffnnnnnn
draw_point_intersect_circles_auxiv:nnnnnnnn
draw_point_intersect_circles_auxiv:ffnnnnnn
draw_point_intersect_circles_auxv:nnnnnnnn
draw_point_intersect_circles_auxv:ffnnnnnn
draw_point_intersect_circles_auxvi:nnnnnnnn
draw_point_intersect_circles_auxvi:ffnnnnnn
draw_point_intersect_circles_auxvii:nnnnnnnn
draw_point_intersect_circles_auxvii:ffnnnnnn

```

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

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

in either

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

or

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

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

```

883 \cs_new:Npn \draw_point_intersect_circles:nnnnn #1#2#3#4#5
884 {
885   \__draw_point_process:nnn
886   { \__draw_point_intersect_circles_auxi:nnnnnnnn {#2} {#4} {#5} }
887   {#1} {#3}
888 }
889 \cs_new:Npn \__draw_point_intersect_circles_auxi:nnnnnnnn #1#2#3#4#5#6#7
890 {
891   \__draw_point_intersect_circles_auxii:ffnnnnnn
892   { \fp_eval:n {#1} } { \fp_eval:n {#2} } {#4} {#5} {#6} {#7} {#3}
893 }

```

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

```

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

```


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

```

894 \cs_new:Npn \__draw_point_intersect_circles_auxii:nnnnnnn #1#2#3#4#5#6#7
895 {
896   \__draw_point_intersect_circles_auxiii:ffnnnnnn
897   { \fp_eval:n { #5 - #3 } }
898   { \fp_eval:n { #6 - #4 } }
899   {#1} {#2} {#3} {#4} {#7}
900 }
901 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxii:nnnnnnn { ff }
902 \cs_new:Npn \__draw_point_intersect_circles_auxiii:nnnnnnn #1#2#3#4#5#6#7
903 {
904   \__draw_point_intersect_circles_auxiv:fnnnnnnn
905   { \fp_eval:n { sqrt( #1 * #1 + #2 * #2 ) } }
906   {#1} {#2} {#3} {#4} {#5} {#6} {#7}
907 }
908 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxiii:nnnnnnn { ff }

```

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

```

909 \cs_new:Npn \__draw_point_intersect_circles_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
910 {
911   \__draw_point_intersect_circles_auxv:ffnnnnnnnn
912   { \fp_eval:n { 1 / #1 } }
913   { \fp_eval:n { #4 * #4 } }
914   {#1} {#2} {#3} {#5} {#6} {#7} {#8}
915 }
916 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxiv:nnnnnnnn { f }
917 \cs_new:Npn \__draw_point_intersect_circles_auxv:nnnnnnnnn #1#2#3#4#5#6#7#8#9
918 {
919   \__draw_point_intersect_circles_auxvi:fnnnnnnnn
920   { \fp_eval:n { 0.5 * #1 * ( #2 + #3 * #3 - #6 * #6 ) } }
921   {#1} {#2} {#4} {#5} {#7} {#8} {#9}
922 }
923 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxv:nnnnnnnnn { ff }

```

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

#1 k

#2 $1/p$

#3 r^2

#4 e

#5 f

#6 a

#7 b

#8 n

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

```

924 \cs_new:Npn \__draw_point_intersect_circles_auxvi:nnnnnnnn #1#2#3#4#5#6#7#8
925 {
926   \__draw_point_intersect_circles_auxvii:fffnnnn
927   { \fp_eval:n { #1 * #2 } }
928   { \int_if_odd:nTF {#8} { 1 } { -1 } }
929   { \fp_eval:n { sqrt ( #3 - #1 * #1 ) * #2 } }
930   {#4} {#5} {#6} {#7}
931 }
932 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxvi:nnnnnnnn { f }
933 \cs_new:Npn \__draw_point_intersect_circles_auxvii:nnnnnnnn #1#2#3#4#5#6#7
934 {
935   \__draw_point_to_dim:n
936   { #6 + #4 * #1 + #2 * #3 * #5 , #7 + #5 * #1 + -1 * #2 * #3 * #4 }
937 }
938 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxvii:nnnnnnnn { fff }

```

5.5 Interpolation on a line (vector) or arc

Simple maths after expansion.

```

\draw_point_interpolate_line:nnn
\__draw_point_interpolate_line_aux:nnnnn
\__draw_point_interpolate_line_aux:fnnnn
\__draw_point_interpolate_line_aux:nnnnnn
\__draw_point_interpolate_line_aux:fnnnnnn
939 \cs_new:Npn \draw_point_interpolate_line:nnn #1#2#3
940 {
941   \__draw_point_process:nnn
942   { \__draw_point_interpolate_line_aux:fnnnn { \fp_eval:n {#1} } }
943   {#2} {#3}
944 }
945 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnn #1#2#3#4#5
946 {
947   \__draw_point_interpolate_line_aux:fnnnnnn { \fp_eval:n { 1 - #1 } }
948   {#1} {#2} {#3} {#4} {#5}
949 }
950 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnnn { f }
951 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnnn #1#2#3#4#5#6
952 { \__draw_point_to_dim:n { #2 * #3 + #1 * #5 , #2 * #4 + #1 * #6 } }
953 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnnn { f }

```

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

```

\draw_point_interpolate_distance:nnn
\__draw_point_interpolate_distance:nnnnn
\__draw_point_interpolate_distance:nnnnnn
\__draw_point_interpolate_distance:fnnnnnn
954 \cs_new:Npn \draw_point_interpolate_distance:nnn #1#2#3
955 {
956   \__draw_point_process:nn
957   { \__draw_point_interpolate_distance:nnnn {#1} {#3} }
958   {#2}
959 }
960 \cs_new:Npn \__draw_point_interpolate_distance:nnnn #1#2#3#4
961 {
962   \__draw_point_process:nn
963   {
964     \__draw_point_interpolate_distance:fnnnnn
965     { \fp_eval:n {#1} } {#3} {#4}
966   }
967   { \draw_point_unit_vector:n { ( #2 ) - ( #3 , #4 ) } }

```

```

968 }
969 \cs_new:Npn \__draw_point_interpolate_distance:nnnnn #1#2#3#4#5
970 { \__draw_point_to_dim:n { #2 + #1 * #4 , #3 + #1 * #5 } }
971 \cs_generate_variant:Nn \__draw_point_interpolate_distance:nnnnn { f }

```

(End definition for __draw_point_to_dim: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:ffnnnnnnnn

```

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

```

972 \cs_new:Npn \draw_point_interpolate_arcaxes:nnnnnn #1#2#3#4#5#6
973 {
974   \__draw_point_process:nnnn
975   { \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn {#1} {#5} {#6} }
976   {#2} {#3} {#4}
977 }
978 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
979 {
980   \__draw_point_interpolate_arcaxes_auxii:ffnnnnnnnn
981   { \fp_eval:n {#1} } {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
982 }

```

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

```

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

```

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

```

983 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
984 {
985   \__draw_point_interpolate_arcaxes_auxiii:ffnnnnnnnn
986   { \fp_eval:n { #1 * (#3) + ( 1 - #1 ) * (#2) } }
987   {#4} {#5} {#6} {#7} {#8} {#9}
988 }
989 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnnn { f }
990 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnnn #1#2#3#4#5#6#7
991 {
992   \__draw_point_interpolate_arcaxes_auxiv:ffnnnnnnnn
993   { \fp_eval:n { cosd (#1) } }
994   { \fp_eval:n { sind (#1) } }
995   {#2} {#3} {#4} {#5} {#6} {#7}

```

```

996 }
997 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnn { f }
998 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
999 {
1000   \__draw_point_to_dim:n
1001     { #3 + #1 * #5 + #2 * #7 , #4 + #1 * #6 + #2 * #8 }
1002 }
1003 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnn { ff }

```

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

```

\draw_point_interpolate_curve:nnnnnn
\draw_point_interpolate_curve_auxi:nnnnnnnnnn
\draw_point_interpolate_curve_auxii:nnnnnnnnnn
\draw_point_interpolate_curve_auxiii:nnnnnnnnnn
\draw_point_interpolate_curve_auxiiii:nnnnnnnnnn
\draw_point_interpolate_curve_auxv:nnnnnnnnnn
\draw_point_interpolate_curve_auxvi:nnnnnnnnnn
\draw_point_interpolate_curve_auxvii:nnnnnnnnnn
\draw_point_interpolate_curve_auxviii:nnnnnnnnnn
\draw_point_interpolate_curve_auxviiii:nnnnnnnnnn

```

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.

```

1004 \cs_new:Npn \draw_point_interpolate_curve:nnnnnn #1#2#3#4#5
1005 {
1006   \__draw_point_process:nnnnnn
1007     { \__draw_point_interpolate_curve_auxi:nnnnnnnnnn {#1} }
1008       {#2} {#3} {#4} {#5}
1009 }
1010 \cs_new:Npn \__draw_point_interpolate_curve_auxi:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
1011 {
1012   \__draw_point_interpolate_curve_auxii:nnnnnnnnnn
1013     { \fp_eval:n {#1} }
1014       {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1015 }

```

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

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

In the second stage, we can drop the final point

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

and for the final stage only need one set of calculations

$$P_x = (1 - p)x_1'' + px_2''$$

$$P_y = (1 - p)y_1'' + py_2''$$

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

```

1016 \cs_new:Npn \__draw_point_interpolate_curve_auxiii:nnnnnnnnn
1017   #1#2#3#4#5#6#7#8#9
1018   {
1019     \__draw_point_interpolate_curve_auxiii:fnnnnnn
1020     { \fp_eval:n { 1 - #1 } }
1021     {#1}
1022     { {#2} {#3} } { {#4} {#5} } { {#6} {#7} } { {#8} {#9} }
1023   }
1024 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxiii:nnnnnnnnn { f }
1025 % \begin{macrocode}
1026 % We need to do the first cycle, but haven't got enough arguments to keep
1027 % everything in play at once. So here we use a bit of argument re-ordering
1028 % and a single auxiliary to get the job done.
1029 % \begin{macrocode}
1030 \cs_new:Npn \__draw_point_interpolate_curve_auxiii:nnnnnnn #1#2#3#4#5#6
1031   {
1032     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #3 #4
1033     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #4 #5
1034     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #5 #6
1035     \prg_do_nothing:
1036     \__draw_point_interpolate_curve_auxvi:n { {#1} {#2} }
1037   }
1038 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxiii:nnnnnnn { f }
1039 \cs_new:Npn \__draw_point_interpolate_curve_auxiv:nnnnnn #1#2#3#4#5#6
1040   {
1041     \__draw_point_interpolate_curve_auxv:ffw
1042     { \fp_eval:n { #1 * #3 + #2 * #5 } }
1043     { \fp_eval:n { #1 * #4 + #2 * #6 } }
1044   }
1045 \cs_new:Npn \__draw_point_interpolate_curve_auxv:nnw
1046   #1#2#3 \prg_do_nothing: #4#5
1047   {
1048     #3
1049     \prg_do_nothing:
1050     #4 { #5 {#1} {#2} }
1051   }
1052 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxv:nnw { ff }
1053 % \begin{macrocode}
1054 % Get the arguments back into the right places and to the second and
1055 % third cycles directly.
1056 % \begin{macrocode}
1057 \cs_new:Npn \__draw_point_interpolate_curve_auxvi:n #1
1058   { \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1 }
1059 \cs_new:Npn \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1#2#3#4#5#6#7#8
1060   {
1061     \__draw_point_interpolate_curve_auxviii:ffffnn
1062     { \fp_eval:n { #1 * #5 + #2 * #3 } }
1063     { \fp_eval:n { #1 * #6 + #2 * #4 } }

```

```

1064 { \fp_eval:n { #1 * #7 + #2 * #5 } }
1065 { \fp_eval:n { #1 * #8 + #2 * #6 } }
1066 {#1} {#2}
1067 }
1068 \cs_new:Npn \__draw_point_interpolate_curve_auxviii:nnnnnn #1#2#3#4#5#6
1069 {
1070   \__draw_point_to_dim:n
1071     { #5 * #3 + #6 * #1 , #5 * #4 + #6 * #2 }
1072 }
1073 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxviii:nnnnnn { ffff }

```

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

5.6 Vector support

As well as co-ordinates relative to the drawing

```

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

```

(End definition for \l__draw_xvec_x_dim and others.)

```

\draw_xvec:n Calculate the underlying position and store it.
\draw_yvec:n
\draw_zvec:n
\__draw_vec:nn
\__draw_vec:nnn
1080 \cs_new_protected:Npn \draw_xvec:n #1
1081 { \__draw_vec:nn { x } {#1} }
1082 \cs_new_protected:Npn \draw_yvec:n #1
1083 { \__draw_vec:nn { y } {#1} }
1084 \cs_new_protected:Npn \draw_zvec:n #1
1085 { \__draw_vec:nn { z } {#1} }
1086 \cs_new_protected:Npn \__draw_vec:nn #1#2
1087 {
1088   \__draw_point_process:nn { \__draw_vec:nnn {#1} } {#2}
1089 }
1090 \cs_new_protected:Npn \__draw_vec:nnn #1#2#3
1091 {
1092   \dim_set:cn { l__draw_ #1 vec_x_dim } {#2}
1093   \dim_set:cn { l__draw_ #1 vec_y_dim } {#3}
1094 }

```

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

Initialise the vectors.

```

1095 \draw_xvec:n { 1cm , 0cm }
1096 \draw_yvec:n { 0cm , 1cm }
1097 \draw_zvec:n { -0.385cm , -0.385cm }

```

```

\draw_point_vec:nn Force a single evaluation of each factor, then use these to work out the underlying point.
\__draw_point_vec:nn
\__draw_point_vec:ff
\draw_point_vec:nnn
\__draw_point_vec:nnn
\__draw_point_vec:fff

```

```

1100 \cs_new:Npn \__draw_point_vec:nn #1#2
1101 {
1102   \__draw_point_to_dim:n
1103   {
1104     #1 * \l__draw_xvec_x_dim + #2 * \l__draw_yvec_x_dim ,
1105     #1 * \l__draw_xvec_y_dim + #2 * \l__draw_yvec_y_dim
1106   }
1107 }
1108 \cs_generate_variant:Nn \__draw_point_vec:nn { ff }
1109 \cs_new:Npn \draw_point_vec:nnn #1#2#3
1110 {
1111   \__draw_point_vec:fff
1112   { \fp_eval:n {#1} } { \fp_eval:n {#2} } { \fp_eval:n {#3} }
1113 }
1114 \cs_new:Npn \__draw_point_vec:nnn #1#2#3
1115 {
1116   \__draw_point_to_dim:n
1117   {
1118     #1 * \l__draw_xvec_x_dim
1119     + #2 * \l__draw_yvec_x_dim
1120     + #3 * \l__draw_zvec_x_dim
1121     ,
1122     #1 * \l__draw_xvec_y_dim
1123     + #2 * \l__draw_yvec_y_dim
1124     + #3 * \l__draw_zvec_y_dim
1125   }
1126 }
1127 \cs_generate_variant:Nn \__draw_point_vec:nnn { fff }

```

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

```

\draw_point_vec_polar:nn Much the same as the core polar approach.
\draw_point_vec_polar:nnn
\__draw_point_vec_polar:nnn
\__draw_point_vec_polar:fnn
1128 \cs_new:Npn \draw_point_vec_polar:nn #1#2
1129 { \draw_point_vec_polar:nnn {#1} {#2} {#2} }
1130 \cs_new:Npn \draw_point_vec_polar:nnn #1#2#3
1131 { \__draw_draw_vec_polar:fnn { \fp_eval:n {#1} } {#2} {#3} }
1132 \cs_new:Npn \__draw_draw_vec_polar:nnn #1#2#3
1133 {
1134   \__draw_point_to_dim:n
1135   {
1136     cosd(#1) * (#2) * \l__draw_xvec_x_dim ,
1137     sind(#1) * (#3) * \l__draw_yvec_y_dim
1138   }
1139 }
1140 \cs_generate_variant:Nn \__draw_draw_vec_polar:nnn { f }

```

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

5.7 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.

```

1141 \cs_new:Npn \draw_point_transform:n #1

```

```

1142 {
1143   \__draw_point_process:nn
1144   { \__draw_point_transform:nn } {#1}
1145 }
1146 \cs_new:Npn \__draw_point_transform:nn #1#2
1147 {
1148   \bool_if:NTF \l__draw_matrix_active_bool
1149   {
1150     \__draw_point_to_dim:n
1151     {
1152       (
1153         \l__draw_matrix_a_fp * #1
1154         + \l__draw_matrix_c_fp * #2
1155         + \l__draw_xshift_dim
1156       )
1157       ,
1158       (
1159         \l__draw_matrix_b_fp * #1
1160         + \l__draw_matrix_d_fp * #2
1161         + \l__draw_yshift_dim
1162       )
1163     }
1164   }
1165   {
1166     \__draw_point_to_dim:n
1167     {
1168       (#1, #2)
1169       + ( \l__draw_xshift_dim , \l__draw_yshift_dim )
1170     }
1171   }
1172 }

```

(End 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
1173 \cs_new:Npn \__draw_point_transform_noshift:n #1
1174 {
1175   \__draw_point_process:nn
1176   { \__draw_point_transform_noshift:nn } {#1}
1177 }
1178 \cs_new:Npn \__draw_point_transform_noshift:nn #1#2
1179 {
1180   \bool_if:NTF \l__draw_matrix_active_bool
1181   {
1182     \__draw_point_to_dim:n
1183     {
1184       (
1185         \l__draw_matrix_a_fp * #1
1186         + \l__draw_matrix_c_fp * #2
1187       )
1188       ,
1189       (
1190         \l__draw_matrix_b_fp * #1

```



```

1191         + \l__draw_matrix_d_fp * #2
1192     )
1193 }
1194 }
1195 { \__draw_point_to_dim:n { (#1, #2) } }
1196 }

```

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

```
1197 \</initex | package>
```

6 l3draw-scopes implementation

```
1198 \*initex | package>
```

```
1199 \@@=draw>
```

6.1 Drawing environment

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

```

\g__draw_xmin_dim
\g__draw_ymax_dim 1200 \dim_new:N \g__draw_xmax_dim
\g__draw_ymin_dim 1201 \dim_new:N \g__draw_xmin_dim
1202 \dim_new:N \g__draw_ymax_dim
1203 \dim_new:N \g__draw_ymin_dim

```

(End definition for \g__draw_xmax_dim and others.)

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

```
1204 \bool_new:N \l_draw_bb_update_bool
```

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

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

```

1205 \box_new:N \l__draw_main_box
1206 \box_new:N \l__draw_layer_main_box

```

(End definition for \l__draw_layer_main_box.)

\g__draw_id_int The drawing number.

```
1207 \int_new:N \g__draw_id_int
```

(End definition for \g__draw_id_int.)

__draw_reset_bb: A simple auxiliary.

```

1208 \cs_new_protected:Npn \__draw_reset_bb:
1209 {
1210   \dim_gset:Nn \g__draw_xmax_dim { -\c_max_dim }
1211   \dim_gset:Nn \g__draw_xmin_dim { \c_max_dim }
1212   \dim_gset:Nn \g__draw_ymax_dim { -\c_max_dim }
1213   \dim_gset:Nn \g__draw_ymin_dim { \c_max_dim }
1214 }

```

(End 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

```

1215 \cs_new_protected:Npn \draw_begin:
1216 {
1217   \group_begin:
1218     \int_gincr:N \g__draw_id_int
1219     \hbox_set:Nw \l__draw_main_box
1220     \driver_draw_begin:
1221     \__draw_reset_bb:
1222     \__draw_path_reset_limits:
1223     \bool_set_true:N \l_draw_bb_update_bool
1224     \draw_transform_matrix_reset:
1225     \draw_transform_shift_reset:
1226     \__draw_softpath_clear:
1227     \draw_linewidth:n { \l_draw_default_linewidth_dim }
1228     \draw_color:n { . }
1229     \draw_nonzero_rule:
1230     \draw_cap_but:
1231     \draw_join_miter:
1232     \draw_miterlimit:n { 10 }
1233     \draw_dash_pattern:nn { } { 0cm }
1234     \hbox_set:Nw \l__draw_layer_main_box
1235   }
1236 \cs_new_protected:Npn \draw_end:
1237 {
1238   \exp_args:NNNV \hbox_set_end:
1239   \clist_set:Nn \l_draw_layers_clist \l_draw_layers_clist
1240   \__draw_layers_insert:
1241   \driver_draw_end:
1242   \hbox_set_end:
1243   \dim_compare:nNnT \g__draw_xmin_dim = \c_max_dim
1244   {
1245     \dim_gzero:N \g__draw_xmax_dim
1246     \dim_gzero:N \g__draw_xmin_dim
1247     \dim_gzero:N \g__draw_ymax_dim
1248     \dim_gzero:N \g__draw_ymin_dim
1249   }
1250   \hbox_set:Nn \l__draw_main_box
1251   {
1252     \skip_horizontal:n { -\g__draw_xmin_dim }
1253     \box_move_down:nn { \g__draw_ymin_dim }
1254     { \box_use_drop:N \l__draw_main_box }
1255   }
1256   \box_set_ht:Nn \l__draw_main_box
1257   { \g__draw_ymax_dim - \g__draw_ymin_dim }
1258   \box_set_dp:Nn \l__draw_main_box { Opt }
1259   \box_set_wd:Nn \l__draw_main_box
1260   { \g__draw_xmax_dim - \g__draw_xmin_dim }
1261   \mode_leave_vertical:

```

```

1262     \box_use_drop:N \l__draw_main_box
1263   \group_end:
1264 }

```

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

6.2 Scopes

`\l__draw_linewidth_dim` Storage for local variables.
`\l__draw_fill_color_tl` 1265 `\dim_new:N \l__draw_linewidth_dim`
`\l__draw_stroke_color_tl` 1266 `\tl_new:N \l__draw_fill_color_tl`
1267 `\tl_new:N \l__draw_stroke_color_tl`

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

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

```

\draw_scope_begin: 1268 \cs_new_protected:Npn \draw_scope_begin:
1269 {
1270   \driver_draw_scope_begin:
1271   \group_begin:
1272     \dim_set_eq:NN \l__draw_linewidth_dim \g__draw_linewidth_dim
1273     \draw_path_scope_begin:
1274   }
1275 \cs_new_protected:Npn \draw_scope_end:
1276 {
1277   \draw_path_scope_end:
1278   \dim_gset_eq:NN \g__draw_linewidth_dim \l__draw_linewidth_dim
1279   \group_end:
1280   \driver_draw_scope_end:
1281 }

```

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

`\l__draw_xmax_dim` Storage for the bounding box.
`\l__draw_xmin_dim` 1282 `\dim_new:N \l__draw_xmax_dim`
`\l__draw_ymax_dim` 1283 `\dim_new:N \l__draw_xmin_dim`
`\l__draw_ymin_dim` 1284 `\dim_new:N \l__draw_ymax_dim`
1285 `\dim_new:N \l__draw_ymin_dim`

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

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

```

\__draw_scope_bb_end: 1286 \cs_new_protected:Npn \__draw_scope_bb_begin:
1287 {
1288   \group_begin:
1289     \dim_set_eq:NN \l__draw_xmax_dim \g__draw_xmax_dim
1290     \dim_set_eq:NN \l__draw_xmin_dim \g__draw_xmin_dim
1291     \dim_set_eq:NN \l__draw_ymax_dim \g__draw_ymax_dim
1292     \dim_set_eq:NN \l__draw_ymin_dim \g__draw_ymin_dim
1293     \__draw_reset_bb:
1294   }
1295 \cs_new_protected:Npn \__draw_scope_bb_end:
1296 {
1297   \dim_gset_eq:NN \g__draw_xmax_dim \l__draw_xmax_dim

```

```

1298     \dim_gset_eq:NN \g__draw_xmin_dim \l__draw_xmin_dim
1299     \dim_gset_eq:NN \g__draw_ymax_dim \l__draw_ymax_dim
1300     \dim_gset_eq:NN \g__draw_ymin_dim \l__draw_ymin_dim
1301   \group_end:
1302 }

```

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

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

```

\draw_suspend_end:
1303 \cs_new_protected:Npn \draw_suspend_begin:
1304 {
1305   \__draw_scope_bb_begin:
1306   \draw_path_scope_begin:
1307   \draw_transform_matrix_reset:
1308   \draw_transform_shift_reset:
1309   \__draw_layers_save:
1310 }
1311 \cs_new_protected:Npn \draw_suspend_end:
1312 {
1313   \__draw_layers_restore:
1314   \draw_path_scope_end:
1315   \__draw_scope_bb_end:
1316 }

```

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

```

1317 </initex | package>

```

7 l3draw-softpath implementation

```

1318 <*initex | package>

```

```

1319 <@@=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.

```

1320 \tl_new:N \g__draw_softpath_main_tl

```

(End definition for `\g__draw_softpath_main_tl.`)

`\l__draw_softpath_internal_tl` The soft path itself.

```

1321 \tl_new:N \l__draw_softpath_internal_tl

```

(End definition for \l__draw_softpath_internal_tl.)

\g__draw_softpath_corners_bool Allow for optimised path use.

1322 \bool_new:N \g__draw_softpath_corners_bool

(End definition for \g__draw_softpath_corners_bool.)

__draw_softpath_add:n

__draw_softpath_add:o

__draw_softpath_add:x

1323 \cs_new_protected:Npn __draw_softpath_add:n

1324 { \tl_build_gput_right:Nn \g__draw_softpath_main_tl }

1325 \cs_generate_variant:Nn __draw_softpath_add:n { o, x }

(End definition for __draw_softpath_add:n.)

__draw_softpath_use: Using and clearing is trivial.

__draw_softpath_clear:

1326 \cs_new_protected:Npn __draw_softpath_use:

1327 {

1328 \tl_build_get:NN \g__draw_softpath_main_tl \l__draw_softpath_internal_tl

1329 \l__draw_softpath_internal_tl

1330 }

1331 \cs_new_protected:Npn __draw_softpath_clear:

1332 {

1333 \tl_build_gclear:N \g__draw_softpath_main_tl

1334 \bool_gset_false:N \g__draw_softpath_corners_bool

1335 }

(End definition for __draw_softpath_use: and __draw_softpath_clear:.)

\g__draw_softpath_lastx_dim

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

\g__draw_softpath_lasty_dim

1336 \dim_new:N \g__draw_softpath_lastx_dim

1337 \dim_new:N \g__draw_softpath_lasty_dim

(End definition for \g__draw_softpath_lastx_dim and \g__draw_softpath_lasty_dim.)

\g__draw_softpath_move_bool

Track if moving a point should update the close position.

1338 \bool_new:N \g__draw_softpath_move_bool

1339 \bool_gset_true:N \g__draw_softpath_move_bool

(End definition for \g__draw_softpath_move_bool.)

_draw_softpath_curveto:nnnnnn

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

__draw_softpath_lineto:nn

1340 \cs_new_protected:Npn __draw_softpath_closepath:

1341 {

__draw_softpath_moveto:nn

_draw_softpath_rectangle:nnnn

1342 __draw_softpath_add:x

_draw_softpath_roundpoint:nn

1343 {

_draw_softpath_roundpoint:VV

1344 __draw_softpath_close_op:nn

1345 { \dim_use:N \g__draw_softpath_lastx_dim }

1346 { \dim_use:N \g__draw_softpath_lasty_dim }

1347 }

1348 }

1349 \cs_new_protected:Npn __draw_softpath_curveto:nnnnnn #1#2#3#4#5#6

1350 {

1351 __draw_softpath_add:n

1352 {

1353 __draw_softpath_curveto_opi:nn {#1} {#2}

```

1354         \_draw_softpath_curveto_opii:nn {#3} {#4}
1355         \_draw_softpath_curveto_opiii:nn {#5} {#6}
1356     }
1357 }
1358 \cs_new_protected:Npn \_draw_softpath_lineto:nn #1#2
1359 {
1360     \_draw_softpath_add:n
1361     { \_draw_softpath_lineto_op:nn {#1} {#2} }
1362 }
1363 \cs_new_protected:Npn \_draw_softpath_moveto:nn #1#2
1364 {
1365     \_draw_softpath_add:n
1366     { \_draw_softpath_moveto_op:nn {#1} {#2} }
1367     \bool_if:NT \g__draw_softpath_move_bool
1368     {
1369         \dim_gset:Nn \g__draw_softpath_lastx_dim {#1}
1370         \dim_gset:Nn \g__draw_softpath_lasty_dim {#2}
1371     }
1372 }
1373 \cs_new_protected:Npn \_draw_softpath_rectangle:nnnn #1#2#3#4
1374 {
1375     \_draw_softpath_add:n
1376     {
1377         \_draw_softpath_rectangle_opi:nn {#1} {#2}
1378         \_draw_softpath_rectangle_opii:nn {#3} {#4}
1379     }
1380 }
1381 \cs_new_protected:Npn \_draw_softpath_roundpoint:nn #1#2
1382 {
1383     \_draw_softpath_add:n
1384     { \_draw_softpath_roundpoint_op:nn {#1} {#2} }
1385     \bool_gset_true:N \g__draw_softpath_corners_bool
1386 }
1387 \cs_generate_variant:Nn \_draw_softpath_roundpoint:nn { VV }

```

(End definition for _draw_softpath_curveto:nnnnnn and others.)

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

```

\_draw_softpath_curveto_opi:nn 1388 \cs_new_protected:Npn \_draw_softpath_close_op:nn #1#2
\_draw_softpath_curveto_opii:nn 1389 { \driver_draw_closepath: }
\_draw_softpath_curveto_opiii:nn 1390 \cs_new_protected:Npn \_draw_softpath_curveto_opi:nn #1#2
\_draw_softpath_lineto_op:nn 1391 { \_draw_softpath_curveto_opi:nnNnnNnn {#1} {#2} }
\_draw_softpath_moveto_op:nn 1392 \cs_new_protected:Npn \_draw_softpath_curveto_opi:nnNnnNnn #1#2#3#4#5#6#7#8
\_draw_softpath_roundpoint_op:nn 1393 { \driver_draw_curveto:nnnnnn {#1} {#2} {#4} {#5} {#7} {#8} }
\_draw_softpath_rectangle_opi:nn 1394 \cs_new_protected:Npn \_draw_softpath_curveto_opii:nn #1#2
\_draw_softpath_rectangle_opii:nn 1395 { \_draw_softpath_curveto_opii:nn }
\_draw_softpath_curveto_opi:nnNnnNnn 1396 \cs_new_protected:Npn \_draw_softpath_curveto_opiii:nn #1#2
\_draw_softpath_rectangle_opi:nnNnn 1397 { \_draw_softpath_curveto_opiii:nn }
1398 \cs_new_protected:Npn \_draw_softpath_lineto_op:nn #1#2
1399 { \driver_draw_lineto:nn {#1} {#2} }
1400 \cs_new_protected:Npn \_draw_softpath_moveto_op:nn #1#2
1401 { \driver_draw_moveto:nn {#1} {#2} }

```

```

1402 \cs_new_protected:Npn \__draw_softpath_roundpoint_op:nn #1#2 { }
1403 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nn #1#2
1404 { \__draw_softpath_rectangle_opi:nnNnn {#1} {#2} }
1405 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nnNnn #1#2#3#4#5
1406 { \driver_draw_rectangle:nnnn {#1} {#2} {#4} {#5} }
1407 \cs_new_protected:Npn \__draw_softpath_rectangle_opii:nn #1#2 { }

```

(End definition for __draw_softpath_close_op:nn and others.)

7.2 Rounding soft path corners

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

\l__draw_softpath_main_tl For constructing the updated path.

```
1408 \tl_new:N \l__draw_softpath_main_tl
```

(End definition for \l__draw_softpath_main_tl.)

\l__draw_softpath_part_tl Data structures.

```
1409 \tl_new:N \l__draw_softpath_part_tl
```

```
1410 \tl_new:N \l__draw_softpath_curve_end_tl
```

(End definition for \l__draw_softpath_part_tl.)

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

```
1411 \fp_new:N \l__draw_softpath_lastx_fp
```

```
1412 \fp_new:N \l__draw_softpath_lasty_fp
```

```
1413 \dim_new:N \l__draw_softpath_corneri_dim
```

```
1414 \dim_new:N \l__draw_softpath_cornerii_dim
```

```
1415 \tl_new:N \l__draw_softpath_first_tl
```

```
1416 \tl_new:N \l__draw_softpath_move_tl
```

(End definition for \l__draw_softpath_lastx_fp and others.)

\c__draw_softpath_arc_fp The magic constant.

```
1417 \fp_const:Nn \c__draw_softpath_arc_fp { 4/3 * (sqrt(2) - 1) }
```

(End definition for \c__draw_softpath_arc_fp.)

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

```
1418 \cs_new_protected:Npn \__draw_softpath_round_corners:
```

```
1419 {
```

```
1420 \bool_if:NT \g__draw_softpath_corners_bool
```

```
1421 {
```

```
1422 \group_begin:
```

```
1423 \tl_clear:N \l__draw_softpath_main_tl
```

```
1424 \tl_clear:N \l__draw_softpath_part_tl
```

```
1425 \fp_zero:N \l__draw_softpath_lastx_fp
```

```
1426 \fp_zero:N \l__draw_softpath_lasty_fp
```

```
1427 \tl_clear:N \l__draw_softpath_first_tl
```

```
1428 \tl_clear:N \l__draw_softpath_move_tl
```

```
1429 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1430 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1431 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1432 \tl_clear:N \l__draw_softpath_move_tl
```

```
1433 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1434 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1435 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1436 \tl_clear:N \l__draw_softpath_move_tl
```

```
1437 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1438 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1439 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1440 \tl_clear:N \l__draw_softpath_move_tl
```

```
1441 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1442 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1443 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1444 \tl_clear:N \l__draw_softpath_move_tl
```

```
1445 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1446 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1447 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1448 \tl_clear:N \l__draw_softpath_move_tl
```

```
1449 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1450 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1451 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1452 \tl_clear:N \l__draw_softpath_move_tl
```

```
1453 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1454 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1455 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1456 \tl_clear:N \l__draw_softpath_move_tl
```

```
1457 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1458 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1459 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1460 \tl_clear:N \l__draw_softpath_move_tl
```

```
1461 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1462 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1463 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1464 \tl_clear:N \l__draw_softpath_move_tl
```

```
1465 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1466 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1467 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1468 \tl_clear:N \l__draw_softpath_move_tl
```

```
1469 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1470 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1471 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1472 \tl_clear:N \l__draw_softpath_move_tl
```

```
1473 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1474 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1475 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1476 \tl_clear:N \l__draw_softpath_move_tl
```

```
1477 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1478 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1479 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1480 \tl_clear:N \l__draw_softpath_move_tl
```

```
1481 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1482 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1483 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1484 \tl_clear:N \l__draw_softpath_move_tl
```

```
1485 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1486 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1487 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1488 \tl_clear:N \l__draw_softpath_move_tl
```

```
1489 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1490 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1491 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1492 \tl_clear:N \l__draw_softpath_move_tl
```

```
1493 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1494 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1495 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1496 \tl_clear:N \l__draw_softpath_move_tl
```

```
1497 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1498 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1499 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1500 \tl_clear:N \l__draw_softpath_move_tl
```

```
1501 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1502 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1503 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1504 \tl_clear:N \l__draw_softpath_move_tl
```

```
1505 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1506 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1507 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1508 \tl_clear:N \l__draw_softpath_move_tl
```

```
1509 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1510 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1511 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1512 \tl_clear:N \l__draw_softpath_move_tl
```

```
1513 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1514 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1515 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1516 \tl_clear:N \l__draw_softpath_move_tl
```

```
1517 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1518 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1519 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1520 \tl_clear:N \l__draw_softpath_move_tl
```

```
1521 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1522 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1523 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1524 \tl_clear:N \l__draw_softpath_move_tl
```

```
1525 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1526 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1527 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1528 \tl_clear:N \l__draw_softpath_move_tl
```

```
1529 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1530 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1531 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1532 \tl_clear:N \l__draw_softpath_move_tl
```

```
1533 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1534 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1535 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1536 \tl_clear:N \l__draw_softpath_move_tl
```

```
1537 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1538 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1539 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1540 \tl_clear:N \l__draw_softpath_move_tl
```

```
1541 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1542 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1543 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1544 \tl_clear:N \l__draw_softpath_move_tl
```

```
1545 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1546 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1547 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1548 \tl_clear:N \l__draw_softpath_move_tl
```

```
1549 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1550 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1551 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1552 \tl_clear:N \l__draw_softpath_move_tl
```

```
1553 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1554 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1555 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1556 \tl_clear:N \l__draw_softpath_move_tl
```

```
1557 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1558 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1559 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1560 \tl_clear:N \l__draw_softpath_move_tl
```

```
1561 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1562 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1563 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1564 \tl_clear:N \l__draw_softpath_move_tl
```

```
1565 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1566 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1567 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1568 \tl_clear:N \l__draw_softpath_move_tl
```

```
1569 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1570 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1571 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1572 \tl_clear:N \l__draw_softpath_move_tl
```

```
1573 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1574 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1575 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1576 \tl_clear:N \l__draw_softpath_move_tl
```

```
1577 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1578 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1579 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1580 \tl_clear:N \l__draw_softpath_move_tl
```

```
1581 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1582 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1583 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1584 \tl_clear:N \l__draw_softpath_move_tl
```

```
1585 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1586 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1587 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1588 \tl_clear:N \l__draw_softpath_move_tl
```

```
1589 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1590 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1591 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1592 \tl_clear:N \l__draw_softpath_move_tl
```

```
1593 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1594 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1595 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1596 \tl_clear:N \l__draw_softpath_move_tl
```

```
1597 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1598 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1599 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1600 \tl_clear:N \l__draw_softpath_move_tl
```

```
1601 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1602 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1603 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1604 \tl_clear:N \l__draw_softpath_move_tl
```

```
1605 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1606 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1607 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1608 \tl_clear:N \l__draw_softpath_move_tl
```

```
1609 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1610 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1611 \tl_clear:N \l__draw_softpath_cornerii_dim
```

```
1612 \tl_clear:N \l__draw_softpath_move_tl
```

```
1613 \tl_clear:N \l__draw_softpath_curve_end_tl
```

```
1614 \tl_clear:N \l__draw_softpath_corneri_dim
```

```
1615
```

```

1429         \tl_build_get:NN \g__draw_softpath_main_tl \l__draw_softpath_internal_tl
1430         \exp_after:wN \__draw_softpath_round_loop:Nnn
1431         \l__draw_softpath_internal_tl
1432         \q_recursion_tail ? ?
1433         \q_recursion_stop
1434     \group_end:
1435 }
1436 \bool_gset_false:N \g__draw_softpath_corners_bool
1437 }

```

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.

```

1438 \cs_new_protected:Npn \__draw_softpath_round_loop:Nnn #1#2#3
1439 {
1440     \quark_if_recursion_tail_stop_do:Nn #1 { \__draw_softpath_round_end: }
1441     \token_if_eq_meaning:NNTF #1 \__draw_softpath_roundpoint_op:nn
1442     { \__draw_softpath_round_action:nn {#2} {#3} }
1443     {
1444         \tl_if_empty:NT \l__draw_softpath_first_tl
1445         { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1446         \fp_set:Nn \l__draw_softpath_lastx_fp {#2}
1447         \fp_set:Nn \l__draw_softpath_lasty_fp {#3}
1448         \token_if_eq_meaning:NNTF #1 \__draw_softpath_moveto_op:nn
1449         {
1450             \tl_put_right:No \l__draw_softpath_main_tl
1451             \l__draw_softpath_move_tl
1452             \tl_put_right:No \l__draw_softpath_main_tl
1453             \l__draw_softpath_part_tl
1454             \tl_set:Nn \l__draw_softpath_move_tl { #1 {#2} {#3} }
1455             \tl_clear:N \l__draw_softpath_first_tl
1456             \tl_clear:N \l__draw_softpath_part_tl
1457         }
1458         { \tl_put_right:Nn \l__draw_softpath_part_tl { #1 {#2} {#3} } }
1459     }
1460 }
1461 }
1462 \cs_new_protected:Npn \__draw_softpath_round_action:nn #1#2
1463 {
1464     \dim_set:Nn \l__draw_softpath_corneri_dim {#1}
1465     \dim_set:Nn \l__draw_softpath_cornerii_dim {#2}
1466     \bool_lazy_and:nnTF
1467     { \dim_compare_p:nNn \l__draw_softpath_corneri_dim = { 0pt } }
1468     { \dim_compare_p:nNn \l__draw_softpath_cornerii_dim = { 0pt } }
1469     { \__draw_softpath_round_loop:Nnn }
1470     { \__draw_softpath_round_action:Nnn }
1471 }

```

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.

```

1472 \cs_new_protected:Npn \__draw_softpath_round_action:Nnn #1#2#3

```



```

1473 {
1474   \tl_if_empty:NT \l__draw_softpath_first_tl
1475   { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1476   \token_if_eq_meaning:NNTF #1 \__draw_softpath_curveto_opi:nn
1477   { \__draw_softpath_round_action_curveto:NnnNnn }
1478   {
1479     \token_if_eq_meaning:NNTF #1 \__draw_softpath_close_op:nn
1480     { \__draw_softpath_round_action_close: }
1481     {
1482       \token_if_eq_meaning:NNTF #1 \__draw_softpath_lineto_op:nn
1483       { \__draw_softpath_round_lookahead:NnnNnn }
1484       { \__draw_softpath_round_loop:Nnn }
1485     }
1486   }
1487   #1 {#2} {#3}
1488 }

```

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.

```

1489 \cs_new_protected:Npn \__draw_softpath_round_action_curveto:NnnNnn
1490   #1#2#3#4#5#6
1491   {
1492     \tl_put_right:Nn \l__draw_softpath_part_tl
1493     { #1 {#2} {#3} #4 {#5} {#6} }
1494     \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1495     \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1496     \__draw_softpath_round_lookahead:NnnNnn
1497   }
1498 \cs_new_protected:Npn \__draw_softpath_round_action_close:
1499   {
1500     \bool_lazy_and:nnTF
1501     { ! \tl_if_empty_p:N \l__draw_softpath_first_tl }
1502     { ! \tl_if_empty_p:N \l__draw_softpath_move_tl }
1503     {
1504       \exp_after:wN \__draw_softpath_round_close:nn
1505       \l__draw_softpath_first_tl
1506     }
1507     { \__draw_softpath_round_loop:Nnn }
1508   }

```

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.

```

1509 \cs_new_protected:Npn \__draw_softpath_round_lookahead:NnnNnn #1#2#3#4#5#6
1510   {
1511     \bool_lazy_any:nTF
1512     {
1513       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_lineto_op:nn }
1514       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_curveto_opi:nn }
1515       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_close_op:nn }
1516     }
1517     {
1518       \__draw_softpath_round_calc:nnnNnn
1519       \__draw_softpath_round_loop:Nnn {#5} {#6}
1520     }

```

```

1521     {
1522         \token_if_eq_meaning:NNTF #4 \__draw_softpath_roundpoint_op:nn
1523         { \__draw_softpath_round_roundpoint:NnnNnnNnn }
1524         { \__draw_softpath_round_loop:Nnn }
1525     }
1526     #1 {#2} {#3}
1527     #4 {#5} {#6}
1528 }
1529 \cs_new_protected:Npn \__draw_softpath_round_roundpoint:NnnNnnNnn
1530 #1#2#3#4#5#6#7#8#9
1531 {
1532     \__draw_softpath_round_calc:nnnNnn
1533     \__draw_softpath_round_loop:Nnn
1534     {#8} {#9} #1 {#2} {#3}
1535     #4 {#5} {#6} #7 {#8} {#9}
1536 }

```

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 (#4, #5), and where the next point is (#1, #2). There are two types of calculations to do. First, we need to interpolate from those two points in the direction of the corner, in order to work out where the curve we are adding will start and end. From those, plus the points we already have, we work out where the control points will lie. All of this is done in an expansion to avoid multiple calls to `\tl_put_right:Nx`. The end point of the line is worked out up-front and saved: we need that if dealing with a close-path operation.

```

1537 \cs_new_protected:Npn \__draw_softpath_round_calc:nnnNnn #1#2#3#4#5#6
1538 {
1539     \tl_set:Nx \l__draw_softpath_curve_end_tl
1540     {
1541         \draw_point_interpolate_distance:nnn
1542         \l__draw_softpath_cornerii_dim
1543         { #5 , #6 } { #2 , #3 }
1544     }
1545     \tl_put_right:Nx \l__draw_softpath_part_tl
1546     {
1547         \exp_not:N #4
1548         \__draw_softpath_round_calc:fVnnnn
1549         {
1550             \draw_point_interpolate_distance:nnn
1551             \l__draw_softpath_corneri_dim
1552             { #5 , #6 }
1553             {
1554                 \l__draw_softpath_lastx_fp ,
1555                 \l__draw_softpath_lasty_fp
1556             }
1557         }
1558         \l__draw_softpath_curve_end_tl
1559         {#5} {#6} {#2} {#3}
1560     }
1561     \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1562     \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1563     #1
1564 }

```

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

```

1565 \cs_new:Npn \__draw_softpath_round_calc:nnnnnn #1#2#3#4#5#6
1566 {
1567   \__draw_softpath_round_calc:nnnnw {#3} {#4} {#5} {#6}
1568   #1 \q_mark #2 \q_stop
1569 }
1570 \cs_generate_variant:Nn \__draw_softpath_round_calc:nnnnnn { fV }

```

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

```

1571 \cs_new:Npn \__draw_softpath_round_calc:nnnnw
1572   #1#2#3#4 #5 , #6 \q_mark #7 , #8 \q_stop
1573 {
1574   {#5} {#6}
1575   \exp_not:N \__draw_softpath_curveto_opi:nn
1576   {
1577     \fp_to_dim:n
1578     { #5 + \c__draw_softpath_arc_fp * ( #1 - #5 ) }
1579   }
1580   {
1581     \fp_to_dim:n
1582     { #6 + \c__draw_softpath_arc_fp * ( #2 - #6 ) }
1583   }
1584   \exp_not:N \__draw_softpath_curveto_opii:nn
1585   {
1586     \fp_to_dim:n
1587     { #7 + \c__draw_softpath_arc_fp * ( #1 - #7 ) }
1588   }
1589   {
1590     \fp_to_dim:n
1591     { #8 + \c__draw_softpath_arc_fp * ( #2 - #8 ) }
1592   }
1593   \exp_not:N \__draw_softpath_curveto_opiii:nn
1594   {#7} {#8}
1595 }

```

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.

```

1596 \cs_new_protected:Npn \__draw_softpath_round_close:nn #1#2
1597 {
1598   \use:x
1599   {
1600     \__draw_softpath_round_calc:nnnNnn
1601     {
1602       \tl_set:Nx \exp_not:N \l__draw_softpath_move_tl
1603       {
1604         \__draw_softpath_moveto_op:nn
1605         \exp_not:N \exp_after:wN
1606         \exp_not:N \__draw_softpath_round_close:w
1607         \exp_not:N \l__draw_softpath_curve_end_tl
1608         \exp_not:N \q_stop

```

```

1609         }
1610         \use:x
1611         {
1612             \exp_not:N \exp_not:N \exp_not:N \use_i:nnnn
1613             {
1614                 \__draw_softpath_round_loop:Nnn
1615                 \__draw_softpath_close_op:nn
1616                 \exp_not:N \exp_after:wN
1617                 \exp_not:N \__draw_softpath_round_close:w
1618                 \exp_not:N \l__draw_softpath_curve_end_tl
1619                 \exp_not:N \q_stop
1620             }
1621         }
1622     }
1623     {#1} {#2}
1624     \__draw_softpath_lineto_op:nn
1625     \exp_after:wN \use_none:n \l__draw_softpath_move_tl
1626 }
1627 }
1628 \cs_new:Npn \__draw_softpath_round_close:w #1 , #2 \q_stop { {#1} {#2} }

```

Tidy up the parts of the path, complete the built token list and put it back into action.

```

1629 \cs_new_protected:Npn \__draw_softpath_round_end:
1630 {
1631     \tl_put_right:No \l__draw_softpath_main_tl
1632     \l__draw_softpath_move_tl
1633     \tl_put_right:No \l__draw_softpath_main_tl
1634     \l__draw_softpath_part_tl
1635     \tl_build_gclear:N \g__draw_softpath_main_tl
1636     \__draw_softpath_add:o \l__draw_softpath_main_tl
1637 }

```

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

```

1638 </initex | package>

```

8 l3draw-state implementation

```

1639 <*initex | package>
1640 <@@=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.

```

1641 \dim_new:N \g__draw_linewidth_dim

```

(End definition for `\g__draw_linewidth_dim`.)

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

```

1642 \dim_new:N \l_draw_default_linewidth_dim
1643 \dim_set:Nn \l_draw_default_linewidth_dim { 0.4pt }

```

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

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

```

1644 \cs_new_protected:Npn \draw_linewidth:n #1
1645 {
1646   \dim_gset:Nn \g__draw_linewidth_dim { \fp_to_dim:n {#1} }
1647   \driver_draw_linewidth:n \g__draw_linewidth_dim
1648 }

```

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

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

```

\l__draw_tmp_seq
1649 \cs_new_protected:Npn \draw_dash_pattern:nn #1#2
1650 {
1651   \group_begin:
1652   \seq_set_from_clist:Nn \l__draw_tmp_seq {#1}
1653   \seq_set_map:Nn \l__draw_tmp_seq \l__draw_tmp_seq
1654     { \fp_to_dim:n {##1} }
1655   \use:x
1656   {
1657     \driver_draw_dash_pattern:nn
1658     { \seq_use:Nn \l__draw_tmp_seq { , } }
1659     { \fp_to_dim:n {#2} }
1660   }
1661   \group_end:
1662 }
1663 \seq_new:N \l__draw_tmp_seq

```

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

\draw_miterlimit:n Pass through to the driver layer.

```

1664 \cs_new_protected:Npn \draw_miterlimit:n #1
1665 { \driver_draw_miterlimit:n { \fp_eval:n {#1} } }

```

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

\draw_cap_but: All straight wrappers.

```

\draw_cap_rectangle: 1666 \cs_new_protected:Npn \draw_cap_but: { \driver_draw_cap_but: }
\draw_cap_round:      1667 \cs_new_protected:Npn \draw_cap_rectangle: { \driver_draw_cap_rectangle: }
\draw_evenodd_rule:   1668 \cs_new_protected:Npn \draw_cap_round: { \driver_draw_cap_round: }
\draw_nonzero_rule:   1669 \cs_new_protected:Npn \draw_evenodd_rule: { \driver_draw_evenodd_rule: }
\draw_join_bevel:     1670 \cs_new_protected:Npn \draw_nonzero_rule: { \driver_draw_nonzero_rule: }
\draw_join_miter:     1671 \cs_new_protected:Npn \draw_join_bevel: { \driver_draw_join_bevel: }
\draw_join_round:     1672 \cs_new_protected:Npn \draw_join_miter: { \driver_draw_join_miter: }
                     1673 \cs_new_protected:Npn \draw_join_round: { \driver_draw_join_round: }

```

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

\l__draw_color_tmp_tl Scratch space.

```

1674 \tl_new:N \l__draw_color_tmp_tl

```

(End definition for \l__draw_color_tmp_tl.)

`\draw_color:n` Much the same as for core color support but calling the relevant driver-level function.

```

\draw_color_fill:n 1675 \cs_new_eq:NN \draw_color:n \color_select:n
\draw_color_stroke:n 1676 \cs_new_protected:Npn \draw_color_fill:n #1
  \__draw_color:nn 1677 { \__draw_color:nn { fill } {#1} }
\__draw_color_aux:nn 1678 \cs_new_protected:Npn \draw_color_stroke:n #1
  \__draw_color_aux:Vn 1679 { \__draw_color:nn { stroke } {#1} }
  \__draw_color:nw 1680 \cs_new_protected:Npn \__draw_color:nn #1#2
  1681 {
  1682   \color_parse:nN {#2} \l__draw_color_tmp_tl
  1683   \__draw_color_aux:Vn \l__draw_color_tmp_tl {#1}
  1684 }
\__draw_select_cmyk:nw 1685 \cs_new_protected:Npn \__draw_color_aux:nn #1#2
  1686 { \__draw_color:nw {#2} #1 \q_stop }
\__draw_select_gray:nw 1687 \cs_generate_variant:Nn \__draw_color_aux:nn { V }
  1688 \cs_new_protected:Npn \__draw_color:nw #1#2 ~ #3 \q_stop
  1689 { \use:c { __draw_color_ #2 :nw } {#1} #3 \q_stop }
\__draw_select_rgb:nw 1690 \cs_new_protected:Npn \__draw_color_cmyk:nw #1#2 ~ #3 ~ #4 ~ #5 \q_stop
  1691 { \use:c { driver_draw_color_ #1 _cmyk:nnnn } {#2} {#3} {#4} {#5} }
\__draw_split_select:nw 1692 \cs_new_protected:Npn \__draw_color_gray:nw #1#2 \q_stop
  1693 { \use:c { driver_draw_color_ #1 _gray:n } {#2} }
  1694 \cs_new_protected:Npn \__draw_color_rgb:nw #1#2 ~ #3 ~ #4 \q_stop
  1695 { \use:c { driver_draw_color_ #1 _rgb:nnn } {#2} {#3} {#4} }
  1696 \cs_new_protected:Npn \__draw_color_spot:nw #1#2 ~ #3 \q_stop
  1697 { \use:c { driver_draw_color_ #1 _spot:nn } {#2} {#3} }

```

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

1698 `</initex | package>`

9 l3draw-transforms implementation

```

1699 <*initex | package>
1700 <@@=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.
- `\pgflowlevelsynccm`, `\pgflowlevel`: Likely to be added when use cases are encountered in other parts of the code.

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

```

1701 \bool_new:N \l__draw_matrix_active_bool

```

(End definition for `\l__draw_matrix_active_bool`.)

\l__draw_matrix_a_fp The active matrix and shifts.
\l__draw_matrix_b_fp 1702 \fp_new:N \l__draw_matrix_a_fp
\l__draw_matrix_c_fp 1703 \fp_new:N \l__draw_matrix_b_fp
\l__draw_xshift_dim 1704 \fp_new:N \l__draw_matrix_c_fp
\l__draw_yshift_dim 1705 \fp_new:N \l__draw_matrix_d_fp
1706 \dim_new:N \l__draw_xshift_dim
1707 \dim_new:N \l__draw_yshift_dim

(End definition for \l__draw_matrix_a_fp and others.)

\draw_transform_matrix_reset: Fast resetting.

\draw_transform_shift_reset: 1708 \cs_new_protected:Npn \draw_transform_matrix_reset:
1709 {
1710 \fp_set:Nn \l__draw_matrix_a_fp { 1 }
1711 \fp_zero:N \l__draw_matrix_b_fp
1712 \fp_zero:N \l__draw_matrix_c_fp
1713 \fp_set:Nn \l__draw_matrix_d_fp { 1 }
1714 }
1715 \cs_new_protected:Npn \draw_transform_shift_reset:
1716 {
1717 \dim_zero:N \l__draw_xshift_dim
1718 \dim_zero:N \l__draw_yshift_dim
1719 }
1720 \draw_transform_matrix_reset:
1721 \draw_transform_shift_reset:

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

\draw_transform_matrix_absolute:nmm 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 1722 \cs_new_protected:Npn \draw_transform_matrix_absolute:nmmn #1#2#3#4
1723 {
1724 \fp_set:Nn \l__draw_matrix_a_fp {#1}
1725 \fp_set:Nn \l__draw_matrix_b_fp {#2}
1726 \fp_set:Nn \l__draw_matrix_c_fp {#3}
1727 \fp_set:Nn \l__draw_matrix_d_fp {#4}
1728 \bool_lazy_all:nTF
1729 {
1730 { \fp_compare_p:nNn \l__draw_matrix_a_fp = \c_one_fp }
1731 { \fp_compare_p:nNn \l__draw_matrix_b_fp = \c_zero_fp }
1732 { \fp_compare_p:nNn \l__draw_matrix_c_fp = \c_zero_fp }
1733 { \fp_compare_p:nNn \l__draw_matrix_d_fp = \c_one_fp }
1734 }
1735 { \bool_set_false:N \l__draw_matrix_active_bool }
1736 { \bool_set_true:N \l__draw_matrix_active_bool }
1737 }
1738 \cs_new_protected:Npn \draw_transform_shift_absolute:n #1
1739 {
1740 _draw_point_process:nn
1741 { _draw_transform_shift_absolute:nn } {#1}
1742 }
1743 \cs_new_protected:Npn _draw_transform_shift_absolute:nn #1#2
1744 {

```

1745     \dim_set:Nn \l__draw_xshift_dim {#1}
1746     \dim_set:Nn \l__draw_yshift_dim {#2}
1747   }

```

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

```

\draw_transform_matrix:nnnn
  \__draw_transform:nnnn
    \draw_transform_shift:n
  \__draw_transform_shift:nn

```

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

```

1748 \cs_new_protected:Npn \draw_transform_matrix:nnnn #1#2#3#4
1749 {
1750   \use:x
1751   {
1752     \__draw_transform:nnnn
1753     { \fp_eval:n {#1} }
1754     { \fp_eval:n {#2} }
1755     { \fp_eval:n {#3} }
1756     { \fp_eval:n {#4} }
1757   }
1758 }
1759 \cs_new_protected:Npn \__draw_transform:nnnn #1#2#3#4
1760 {
1761   \use:x
1762   {
1763     \draw_transform_matrix_absolute:nnnn
1764     { #1 * \l__draw_matrix_a_fp + #2 * \l__draw_matrix_c_fp }
1765     { #1 * \l__draw_matrix_b_fp + #2 * \l__draw_matrix_d_fp }
1766     { #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_c_fp }
1767     { #3 * \l__draw_matrix_b_fp + #4 * \l__draw_matrix_d_fp }
1768   }
1769 }
1770 \cs_new_protected:Npn \draw_transform_shift:n #1
1771 {
1772   \__draw_point_process:nn
1773   { \__draw_transform_shift:nn } {#1}
1774 }
1775 \cs_new_protected:Npn \__draw_transform_shift:nn #1#2
1776 {
1777   \dim_set:Nn \l__draw_xshift_dim { \l__draw_xshift_dim + #1 }
1778   \dim_set:Nn \l__draw_yshift_dim { \l__draw_yshift_dim + #2 }
1779 }

```

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

```

\draw_transform_matrix_invert:
\__draw_transform_invert:n
\__draw_transform_invert:f
  \draw_transform_shift_invert:

```

Standard mathematics: calculate the inverse matrix and use that, then undo the shifts.

```

1780 \cs_new_protected:Npn \draw_transform_matrix_invert:
1781 {
1782   \bool_if:NT \l__draw_matrix_active_bool
1783   {
1784     \__draw_transform_invert:f
1785     {
1786       \fp_eval:n
1787       {
1788         1 /

```



```

1789             (
1790                 \l__draw_matrix_a_fp * \l__draw_matrix_d_fp
1791                 - \l__draw_matrix_b_fp * \l__draw_matrix_c_fp
1792             )
1793         }
1794     }
1795 }
1796 }
1797 \cs_new_protected:Npn \__draw_transform_invert:n #1
1798 {
1799     \fp_set:Nn \l__draw_matrix_a_fp
1800     { \l__draw_matrix_d_fp * #1 }
1801     \fp_set:Nn \l__draw_matrix_b_fp
1802     { -\l__draw_matrix_b_fp * #1 }
1803     \fp_set:Nn \l__draw_matrix_c_fp
1804     { -\l__draw_matrix_c_fp * #1 }
1805     \fp_set:Nn \l__draw_matrix_d_fp
1806     { \l__draw_matrix_a_fp * #1 }
1807 }
1808 \cs_generate_variant:Nn \__draw_transform_invert:n { f }
1809 \cs_new_protected:Npn \draw_transform_shift_invert:
1810 {
1811     \dim_set:Nn \l__draw_xshift_dim { -\l__draw_xshift_dim }
1812     \dim_set:Nn \l__draw_yshift_dim { -\l__draw_yshift_dim }
1813 }

```

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

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

```

1814 \cs_new_protected:Npn \draw_transform_triangle:nnn #1#2#3
1815 {
1816     \__draw_point_process:nnn
1817     {
1818         \__draw_point_process:nn
1819         { \__draw_tranform_triangle:nnnnnn }
1820         {#1}
1821     }
1822     {#2} {#3}
1823 }
1824 \cs_new_protected:Npn \__draw_tranform_triangle:nnnnnn #1#2#3#4#5#6
1825 {
1826     \use:x
1827     {
1828         \draw_transform_matrix_absolute:nnnn
1829         { #3 - #1 }
1830         { #4 - #2 }
1831         { #5 - #1 }
1832         { #6 - #2 }
1833         \draw_transform_shift_absolute:n { #1 , #2 }
1834     }
1835 }

```

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

`\draw_transform_scale:n` Lots of shortcuts.

```

\draw_transform_xscale:n 1836 \cs_new_protected:Npn \draw_transform_scale:n #1
\draw_transform_yscale:n 1837 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { #1 } }
\draw_transform_xshift:n 1838 \cs_new_protected:Npn \draw_transform_xscale:n #1
\draw_transform_yshift:n 1839 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { 1 } }
\draw_transform_xslant:n 1840 \cs_new_protected:Npn \draw_transform_yscale:n #1
\draw_transform_yslant:n 1841 { \draw_transform_matrix:nnnn { 1 } { 0 } { 0 } { #1 } }
1842 \cs_new_protected:Npn \draw_transform_xshift:n #1
1843 { \draw_transform_shift:n { #1 , Opt } }
1844 \cs_new_protected:Npn \draw_transform_yshift:n #1
1845 { \draw_transform_shift:n { Opt , #1 } }
1846 \cs_new_protected:Npn \draw_transform_xslant:n #1
1847 { \draw_transform_matrix:nnnn { 1 } { 0 } { #1 } { 1 } }
1848 \cs_new_protected:Npn \draw_transform_yslant:n #1
1849 { \draw_transform_matrix:nnnn { 1 } { #1 } { 0 } { 1 } }

```

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

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

```

\__draw_transform_rotate:n 1850 \cs_new_protected:Npn \draw_transform_rotate:n #1
\__draw_transform_rotate:f 1851 { \__draw_transform_rotate:f { \fp_eval:n {#1} } }
\__draw_transform_rotate:nn 1852 \cs_new_protected:Npn \__draw_transform_rotate:n #1
\__draw_transform_rotate:ff 1853 {
1854   \__draw_transform_rotate:ff
1855   { \fp_eval:n { cosd(#1) } }
1856   { \fp_eval:n { sind(#1) } }
1857 }
1858 \cs_generate_variant:Nn \__draw_transform_rotate:n { f }
1859 \cs_new_protected:Npn \__draw_transform_rotate:nn #1#2
1860 { \draw_transform_matrix:nnnn {#1} {#2} { -#2 } { #1 } }
1861 \cs_generate_variant:Nn \__draw_transform_rotate:nn { ff }

```

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

1862 `</initex | package>`

Index

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

B

`\begin` . . . 169, 783, 1025, 1029, 1053, 1056

bool commands:

`\bool_gset_eq:Nn` 764

`\bool_gset_false:N` 1334, 1436

`\bool_gset_true:N` 1339, 1385

`\bool_if:NTF`

. 19, 113, 192, 231, 680, 695,

696, 700, 1148, 1180, 1367, 1420, 1782

`\bool_lazy_all:nTF` 1728

`\bool_lazy_and:nnTF`

. 223, 675, 1466, 1500

`\bool_lazy_any:nTF` 1511

`\bool_lazy_or:nnTF` . 556, 651, 683, 688

`\bool_new:N` . 84, 218, 639, 640, 641,

642, 643, 742, 1204, 1322, 1338, 1701

`\bool_set_eq:Nn` 756

`\bool_set_false:N`

. 94, 226, 662, 663, 664, 1735

`\bool_set_true:N` 96,

227, 668, 705, 709, 710, 1223, 1736

box commands:

`\box_dp:N` 15, 65

`\box_gset_eq:Nn` 154

`\box_gset_wd:Nn` 98, 131

`\box_ht:N` 15, 67

`\box_if_exist:NTF` 91

`\box_move_down:nn` 1253

`\box_move_up:nn` 49

`\box_new:N` 11, 78, 79, 1205, 1206

`\box_set_dp:Nn` 53, 1258

`\box_set_eq:Nn` 143

`\box_set_ht:Nn` 52, 1256

`\box_set_wd:Nn` 54, 126, 1259

`\box_use_drop:N`

. 50, 55, 100, 127, 132, 1254, 1262

`\box_wd:N` 15, 64, 66

C

clist commands:

`\clist_map_inline:Nn` . . 122, 139, 150

`\clist_map_inline:nn` 665

`\clist_new:N` 85, 87

`\clist_set:Nn` 86, 1239

coffin commands:

`\coffin_typeset:Nnnnn` 62

`\coffin_wd:N` 64

color commands:

`\color_parse:nN` 1682

`\color_select:n` 1675

cs commands:

`\cs_generate_variant:Nn`

. 418, 605, 638, 842, 850, 882, 901,

908, 916, 923, 932, 938, 950, 953,

971, 989, 997, 1003, 1024, 1038,

1052, 1073, 1108, 1127, 1140, 1325,

1387, 1570, 1687, 1808, 1858, 1861

`\cs_if_exist:NTF` 667

`\cs_if_exist_use:NTF` . . 401, 410, 670

`\cs_new:Npn` 509, 519, 529,

539, 787, 793, 795, 797, 804, 806,

808, 816, 818, 821, 830, 835, 838,

840, 843, 844, 846, 848, 851, 853,

858, 864, 874, 883, 889, 894, 902,

909, 917, 924, 933, 939, 945, 951,

954, 960, 969, 972, 978, 983, 990,

998, 1004, 1010, 1016, 1030, 1039,

1045, 1057, 1059, 1068, 1098, 1100,

1109, 1114, 1128, 1130, 1132, 1141,

1146, 1173, 1178, 1565, 1571, 1628

`\cs_new_eq:Nn` 1675

`\cs_new_protected:Npn` . . . 12, 17,

58, 73, 88, 111, 120, 137, 148, 182,

204, 211, 219, 229, 238, 244, 250,

256, 263, 274, 282, 287, 289, 291,

300, 307, 343, 345, 356, 362, 392,

419, 448, 454, 460, 465, 473, 482,

487, 495, 550, 552, 565, 572, 581,

587, 589, 599, 606, 612, 619, 644,

649, 660, 703, 707, 712, 719, 743,

761, 1080, 1082, 1084, 1086, 1090,

1208, 1215, 1236, 1268, 1275, 1286,

1295, 1303, 1311, 1323, 1326, 1331,

1340, 1349, 1358, 1363, 1373, 1381,

1388, 1390, 1392, 1394, 1396, 1398,

1400, 1402, 1403, 1405, 1407, 1418,

1438, 1462, 1472, 1489, 1498, 1509,

1529, 1537, 1596, 1629, 1644, 1649,

1664, 1666, 1667, 1668, 1669, 1670,

1671, 1672, 1673, 1676, 1678, 1680,

1685, 1688, 1690, 1692, 1694, 1696,

1708, 1715, 1722, 1738, 1743, 1748,

1759, 1770, 1775, 1780, 1797, 1809,

1814, 1824, 1836, 1838, 1840, 1842,

1844, 1846, 1848, 1850, 1852, 1859

D

dim commands:

<code>\dim_abs:n</code>	594, 595
<code>\dim_compare:nNnTF</code>	601, 608, 721, 1243
<code>\dim_compare_p:nNn</code>	224, 225, 1467, 1468
<code>\dim_eval:n</code>	594, 595
<code>\dim_gset:Nn</code>	184, 186,
	188, 190, 194, 196, 198, 200, 206,
	207, 208, 209, 213, 214, 723, 1210,
	1211, 1212, 1213, 1369, 1370, 1646
<code>\dim_gset_eq:NN</code>	
	768, 769, 770, 771, 772, 773,
	774, 775, 1278, 1297, 1298, 1299, 1300
<code>\dim_gzero:N</code> ..	1245, 1246, 1247, 1248
<code>\dim_max:nn</code>	185, 189, 195, 199
<code>\dim_min:nn</code>	187, 191, 197, 201
<code>\dim_new:N</code>	176,
	177, 178, 179, 180, 181, 216, 217,
	734, 735, 736, 737, 738, 739, 740,
	741, 1074, 1075, 1076, 1077, 1078,
	1079, 1200, 1201, 1202, 1203, 1265,
	1282, 1283, 1284, 1285, 1336, 1337,
	1413, 1414, 1641, 1642, 1706, 1707
<code>\dim_set:Nn</code>	221,
	222, 1092, 1093, 1464, 1465, 1643,
	1745, 1746, 1777, 1778, 1811, 1812
<code>\dim_set_eq:NN</code>	
	746, 747, 748, 749, 750, 751,
	752, 753, 1272, 1289, 1290, 1291, 1292
<code>\dim_step_inline:nnnn</code>	621, 629
<code>\dim_use:N</code> ..	721, 726, 728, 1345, 1346
<code>\dim_zero:N</code>	1717, 1718
<code>\c_max_dim</code>	206, 207, 208,
	209, 721, 1210, 1211, 1212, 1213, 1243

draw commands:

<code>\l_draw_bb_update_bool</code>	
	19, 192, 676, 1204, 1223
<code>\draw_begin:</code>	1215
<code>\draw_box_use:N</code>	12
<code>\draw_cap_but:</code>	1230, 1666
<code>\draw_cap_rectangle:</code>	1666
<code>\draw_cap_round:</code>	1666
<code>\draw_coffin_use:Nnn</code>	58
<code>\draw_color:n</code>	1228, 1675
<code>\draw_color_fill:n</code>	1675
<code>\draw_color_stroke:n</code>	1675
<code>\draw_dash_pattern:nn</code> ...	1233, 1649
<code>\l_draw_default_linewidth_dim</code> ...	
	103, 1227, 1642
<code>\draw_end:</code>	1215
<code>\draw_evenodd_rule:</code>	1666
<code>\draw_join_bevel:</code>	1666
<code>\draw_join_miter:</code>	1231, 1666
<code>\draw_join_round:</code>	1666

<code>\draw_layer_begin:n</code>	88
<code>\draw_layer_end:</code>	88
<code>\draw_layer_new:n</code>	73
<code>\l_draw_layers_clist</code>	
	85, 122, 139, 150, 1239
<code>\draw_linewidth:n</code>	103, 1227, 1644
<code>\draw_miterlimit:n</code>	1232, 1664
<code>\draw_nonzero_rule:</code>	1229, 1666
<code>\draw_path_arc:nnn</code>	343, 485
<code>\draw_path_arc:nnnn</code>	343
<code>\draw_path_arc_axes:nnnn</code>	482
<code>\draw_path_canvas_curveto:nnn</code> ..	287
<code>\draw_path_canvas_lineto:n</code>	287
<code>\draw_path_canvas_moveto:n</code>	287
<code>\draw_path_circle:nn</code>	550
<code>\draw_path_close:</code>	282, 578
<code>\draw_path_corner_arc:nn</code>	219
<code>\draw_path_curveto:nn</code>	300
<code>\draw_path_curveto:nnn</code>	238
<code>\draw_path_ellipse:nnn</code>	487, 551
<code>\draw_path_grid:nnnn</code>	589
<code>\draw_path_lineto:n</code>	
	238, 575, 576, 577, 627, 635
<code>\draw_path_moveto:n</code>	
	238, 574, 579, 626, 634
<code>\draw_path_rectangle:nn</code>	552, 588
<code>\draw_path_rectangle_corners:nn</code> ..	581
<code>\draw_path_scope_begin:</code>	
	743, 1273, 1306
<code>\draw_path_scope_end:</code>	743, 1277, 1314
<code>\draw_path_use:n</code>	644
<code>\draw_path_use_clear:n</code>	644
<code>\draw_point_interpolate_arcaxes:nnnnnn</code>	972
<code>\draw_point_interpolate_curve:nnnnnn</code>	1004
<code>\draw_point_interpolate_curve:nnnnnnnn</code>	1004
<code>\draw_point_interpolate_curve_-auxi:nnnnnnnnnn</code>	1004
<code>\draw_point_interpolate_curve_-auxii:nnnnnnnnnn</code>	1004
<code>\draw_point_interpolate_curve_-auxiii:nnnnnnnn</code>	1004
<code>\draw_point_interpolate_curve_-auxiv:nnnnnnnn</code>	1004
<code>\draw_point_interpolate_curve_-auxv:nnw</code>	1004
<code>\draw_point_interpolate_curve_-auxvi:n</code>	1004
<code>\draw_point_interpolate_curve_-auxvii:nnnnnnnnnn</code>	1004
<code>\draw_point_interpolate_curve_-auxviii:nnnnnnnn</code>	1004

\draw_point_interpolate_distance:nnn	1692
..... 954, 1541, 1550	
\draw_point_interpolate_line:nnn	1694
..... 939	
\draw_point_intersect_circles:nnnn	1696
..... 1674, 1682, 1683	
\draw_point_intersect_lines:nnnn	1698
..... 883	
\draw_point_polar:nn	1699
..... 844	
\draw_point_polar:nnn	1699
..... 426, 432, 436, 442, 844	
\draw_point_transform:n	1699
..... 23, 26, 29, 32, 242, 254, 270, 271,	
..... 272, 304, 305, 431, 435, 491, 562, 1141	
\draw_point_unit_vector:n	1699
..... 851, 967	
\draw_point_vec:nn	1699
..... 1098	
\draw_point_vec:nnn	1699
..... 1098	
\draw_point_vec_polar:nn	1699
..... 1128	
\draw_point_vec_polar:nnn	1699
..... 1128	
\draw_scope_begin:	1699
..... 1268	
\draw_scope_end:	1699
..... 1275	
\draw_suspend_begin:	1699
..... 1303	
\draw_suspend_end:	1699
..... 1303	
\draw_transform_matrix:nnnn	1748,
..... 1837, 1839, 1841, 1847, 1849, 1860	
\draw_transform_matrix_absolute:nnnn	1722, 1763, 1828
..... 1722, 1763, 1828	
\draw_transform_matrix_invert:	1780
..... 1780	
\draw_transform_matrix_reset:	1224, 1307, 1708
..... 1224, 1307, 1708	
\draw_transform_rotate:n	1850
..... 1850	
\draw_transform_scale:n	1836
..... 1836	
\draw_transform_shift:n	1748, 1843, 1845
..... 1748, 1843, 1845	
\draw_transform_shift_absolute:n	1722, 1833
..... 1722, 1833	
\draw_transform_shift_invert:	1780
..... 1780	
\draw_transform_shift_reset:	1225, 1308, 1708
..... 1225, 1308, 1708	
\draw_transform_triangle:nnn	484, 1814
..... 484, 1814	
\draw_transform_xscale:n	1836
..... 1836	
\draw_transform_xshift:n	1836
..... 1836	
\draw_transform_xslant:n	1836
..... 1836	
\draw_transform_yscale:n	1836
..... 1836	
\draw_transform_yshift:n	1836
..... 1836	
\draw_transform_yslant:n	1836
..... 1836	
\draw_xvec:n	1080, 1095
..... 1080, 1095	
\draw_yvec:n	1080, 1096
..... 1080, 1096	
\draw_zvec:n	1080, 1097
..... 1080, 1097	
draw internal commands:	
_draw_box_use:Nnnnn	12, 63
_draw_color:nn	1675
_draw_color:nw	1675
_draw_color_aux:nn	1675
_draw_color_cmyk:nw	1690
_draw_color_gray:nw	1692
_draw_color_rgb:nw	1694
_draw_color_spot:nw	1696
\l_draw_color_tmp_tl	1674, 1682, 1683
\l_draw_corner_arc_bool	218, 226, 227, 231, 557
\l_draw_corner_xarc_dim	216, 221, 224, 234
\l_draw_corner_yarc_dim	216, 222, 225, 235
_draw_draw_polar:nnn	844
_draw_draw_vec_polar:nnn	1131, 1132, 1140
\l_draw_fill_color_tl	1265
\g_draw_id_int	1207, 1218
\l_draw_layer_close_bool	84, 94, 96, 113
\l_draw_layer_main_box	126, 127, 1205, 1234
\l_draw_layer_tl	82, 93, 97
\g_draw_layers_clist	85
_draw_layers_insert:	120, 1240
_draw_layers_restore:	137, 1313
_draw_layers_save:	137, 1309
\g_draw_linewidth_dim	729, 1272, 1278, 1641, 1646, 1647
\l_draw_linewidth_dim	1265, 1272, 1278
\l_draw_main_box	1205, 1219,
..... 1250, 1254, 1256, 1258, 1259, 1262	
\l_draw_matrix_a_fp	40, 1153, 1185, 1702, 1710, 1724,
..... 1730, 1764, 1766, 1790, 1799, 1806	
\l_draw_matrix_active_bool	558, 1148, 1180, 1701, 1735, 1736, 1782
\l_draw_matrix_b_fp	41, 1159, 1190, 1702, 1711, 1725,
..... 1731, 1765, 1767, 1791, 1801, 1802	
\l_draw_matrix_c_fp	42, 1154, 1186, 1702, 1712, 1726,
..... 1732, 1764, 1766, 1791, 1803, 1804	
\l_draw_matrix_d_fp	43, 1160, 1191, 1705, 1713, 1727,
..... 1733, 1765, 1767, 1790, 1800, 1805	
_draw_path_arc:nnnn	343
_draw_path_arc:nnNnn	343
\c_draw_path_arc_60_fp	343
\c_draw_path_arc_90_fp	343
_draw_path_arc_add:nnnn	343
_draw_path_arc_aux_add:nn	450, 456, 468, 473
_draw_path_arc_auxi:nnnnNnn	343, 370, 377
_draw_path_arc_auxii:nnnnNnnnn	343

_draw_path_arc_auxiii:nn	343	_draw_path_update_limits:nn	22, 25, 28, 31, 182, 246, 259, 276, 277, 278, 567, 568
_draw_path_arc_auxiv:nnnn	343	_draw_path_use:n	644
_draw_path_arc_auxv:nn	343	_draw_path_use_action_draw:	644
_draw_path_arc_auxvi:nn	343	_draw_path_use_action_fillstroke:	644
\l_draw_path_arc_delta_fp	343	\l_draw_path_use_bb_bool	642
\l_draw_path_arc_start_fp	343	\l_draw_path_use_clear_bool	642, 700
_draw_path_curveto:nnnn	300	\l_draw_path_use_clip_bool	639, 662, 680
_draw_path_curveto:nnnnnn	238, 296, 314, 444, 511, 521, 531, 541	\l_draw_path_use_fill_bool	639, 663, 684, 689, 695, 709
\c_draw_path_curveto_a_fp	300	_draw_path_use_stroke_bb:	644
\c_draw_path_curveto_b_fp	300	_draw_path_use_stroke_bb_-	644
_draw_path_ellipse:nnnnnn	487	aux:NnN	644
_draw_path_ellipse_arci:nnnnnn	487	\l_draw_path_use_stroke_bool	639, 664, 677, 685, 690, 696, 705, 710
_draw_path_ellipse_arcii:nnnnnn	487	\g_draw_path_xmax_dim	178, 184, 185, 206, 748, 770
_draw_path_ellipse_arciiii:nnnnnn	487	\l_draw_path_xmax_dim	734, 748, 770
_draw_path_ellipse_arciiv:nnnnnn	487	\g_draw_path_xmin_dim	178, 186, 187, 207, 749, 771
\c_draw_path_ellipse_fp	487	\l_draw_path_xmin_dim	734, 749, 771
_draw_path_grid_auxi:nnnnnn	589	\g_draw_path_ymax_dim	178, 188, 189, 208, 750, 772
_draw_path_grid_auxiii:nnnnnn	589	\l_draw_path_ymax_dim	734, 750, 772
_draw_path_grid_auxiiii:nnnnnn	589	\g_draw_path_ymin_dim	178, 190, 191, 209, 751, 773
_draw_path_grid_auxiv:nnnnnnnn	589	\l_draw_path_ymin_dim	734, 751, 773
\g_draw_path_lastx_dim	176, 213, 318, 451, 457, 746, 774	_draw_point_interpolate_-	
\l_draw_path_lastx_dim	734, 746, 774	arcaxes_auxi:nnnnnnnnnn	972
\g_draw_path_lasty_dim	176, 214, 325, 452, 458, 747, 775	_draw_point_interpolate_-	
\l_draw_path_lasty_dim	734, 747, 775	arcaxes_auxii:nnnnnnnnnn	972
_draw_path_lineto:nn	238, 290	_draw_point_interpolate_-	
_draw_path_mark_corner:	229, 258, 267, 284, 295, 313, 384	arcaxes_auxiii:nnnnnnnn	972
_draw_path_moveto:nn	238, 288, 499, 507	_draw_point_interpolate_-	
_draw_path_rectangle:nnnn	552	arcaxes_auxiv:nnnnnnnnnn	972
_draw_path_rectangle_corners:nnnn	581	_draw_point_interpolate_curve_-	
_draw_path_rectangle_corners:nnnnnn	584, 587	auxi:nnnnnnnnnn	1007, 1010
_draw_path_rectangle_rounded:nnnn	552	_draw_point_interpolate_curve_-	
_draw_path_reset_limits:	182, 656, 754, 1222	auxii:nnnnnnnnnn	1012, 1016, 1024
\l_draw_path_tmp_tl	173, 421, 444, 463, 467, 471, 475	_draw_point_interpolate_curve_-	
\l_draw_path_tmpa_fp	173, 309, 319, 331	auxiii:nnnnnnnn	1019, 1030, 1038
\l_draw_path_tmpb_fp	173, 310, 326, 335	_draw_point_interpolate_curve_-	
_draw_path_update_last:nn	211, 248, 261, 280, 570	auxiv:nnnnnnnn	1032, 1033, 1034, 1039
		_draw_point_interpolate_curve_-	
		auxv:nnw	1041, 1045, 1052
		_draw_point_interpolate_curve_-	
		auxvi:n	1036, 1057
		_draw_point_interpolate_curve_-	
		auxvii:nnnnnnnnnn	1058, 1059
		_draw_point_interpolate_curve_-	
		auxviii:nnnnnnnn	1061, 1068, 1073

_draw_point_interpolate_-		_draw_point_to_dim_aux:w	838
distance:nnnn	957, 960	_draw_point_transform:nn	1141
_draw_point_interpolate_-		_draw_point_transform_noshift:n	
distance:nnnnn	954, 964	425, 441, 492, 493, 1173
_draw_point_interpolate_-		_draw_point_transform_noshift:nn	
distance:nnnnnn	954	1173
_draw_point_interpolate_line_-		_draw_point_unit_vector:nn	851
aux:nnnnn	939	_draw_point_vec:nn	1098
_draw_point_interpolate_line_-		_draw_point_vec:nnn	1098
aux:nnnnnn	939	_draw_point_vec_polar:nnn	1128
_draw_point_intersect_circles_-		_draw_reset_bb:	1208, 1221, 1293
auxi:nnnnnnn	883	_draw_scope_bb_begin:	1286, 1305
_draw_point_intersect_circles_-		_draw_scope_bb_end:	1286, 1315
auxii:nnnnnnn	883	_draw_select_cmyk:nw	1675
_draw_point_intersect_circles_-		_draw_select_gray:nw	1675
auxiii:nnnnnnn	883	_draw_select_rgb:nw	1675
_draw_point_intersect_circles_-		_draw_softpath_add:n	
auxiv:nnnnnnnn	883	767, 1323, 1342,
_draw_point_intersect_circles_-		1351, 1360, 1365, 1375, 1383, 1636	
_draw_point_intersect_circles_-		\c_draw_softpath_arc_fp	
auxv:nnnnnnnnn	883	1417, 1578, 1582, 1587, 1591
_draw_point_intersect_circles_-		_draw_softpath_clear:	
auxvi:nnnnnnnnn	883	655, 701, 759, 763, 1226, 1326
_draw_point_intersect_circles_-		_draw_softpath_close_op:nn	
auxvii:nnnnnnn	883	1344, 1388, 1479, 1515, 1615
_draw_point_intersect_lines:nnnnnn		_draw_softpath_closepath:	
.....	858	285, 506, 1340
_draw_point_intersect_lines:nnnnnnnn		\l_draw_softpath_corneri_dim	
.....	858	1411, 1464, 1467, 1551
_draw_point_intersect_lines_-		\l_draw_softpath_cornerii_dim	
aux:nnnnnn	858	1411, 1465, 1468, 1542
_draw_point_process:nn		\g_draw_softpath_corners_bool	
.....	21, 24, 27, 30, 240, 252,	758, 765, 1322, 1334, 1385, 1420, 1436	
288, 290, 422, 438, 787, 852, 956,		\l_draw_softpath_corners_bool	
962, 1088, 1143, 1175, 1740, 1772, 1818		734, 757, 766
_draw_point_process:nnn	302,	\l_draw_softpath_curve_end_tl	
428, 554, 583, 591, 787, 885, 941, 1816		1410, 1539, 1558, 1607, 1618
_draw_point_process:nnnn		_draw_softpath_curveto:nnnnnn	
.....	265, 293, 489, 787, 974	279, 1340
_draw_point_process:nnnnn		_draw_softpath_curveto_opi:nn	
.....	787, 860, 1006	1353, 1388, 1476, 1514, 1575
_draw_point_process_auxi:nn	787	_draw_softpath_curveto_-	
_draw_point_process_auxii:nw	787	opi:nnNnnNnn	1388
_draw_point_process_auxiii:nnn	787	_draw_softpath_curveto_opii:nn	
_draw_point_process_auxiv:nw	787	1354, 1388, 1584
_draw_point_process_auxv:nnnn	787	_draw_softpath_curveto_-	
_draw_point_process_auxvi:nw	787	opiii:nn	1355, 1388, 1593
_draw_point_process_auxvii:nnnnn		\l_draw_softpath_first_tl	
.....	787	1411, 1427, 1444,
_draw_point_process_auxviii:nw	787	1445, 1455, 1474, 1475, 1501, 1505	
_draw_point_to_dim:n	790,	\l_draw_softpath_internal_tl	
800, 801, 811, 812, 813, 824, 825,		1321, 1328, 1329, 1429, 1431
826, 827, 838, 1000, 1070, 1102,		\g_draw_softpath_lastx_dim	
1116, 1134, 1150, 1166, 1182, 1195		752, 768, 1336, 1345, 1369
_draw_point_to_dim_aux:n	838		

\l__draw_softpath_lastx_dim	__draw_softpath_round_corners: .
. 740, 752, 768 674, 1418
\l__draw_softpath_lastx_fp	__draw_softpath_round_end: . . 1418
. 1411, 1425, 1446, 1494, 1554, 1561	__draw_softpath_round_lookahead:NnnNnn
\g__draw_softpath_lasty_dim 1418
. 753, 769, 1336, 1346, 1370	__draw_softpath_round_loop:Nnn 1418
\l__draw_softpath_lasty_dim	__draw_softpath_round_roundpoint:NnnNnnNnn
. 741, 753, 769 1418
\l__draw_softpath_lasty_fp	__draw_softpath_roundpoint:nn . .
. 1411, 1426, 1447, 1495, 1555, 1562 233, 1340
__draw_softpath_lineto:nn 260, 1340	__draw_softpath_roundpoint_-
__draw_softpath_lineto_op:nn . . .	op:nn 1384, 1388, 1441, 1522
. 1361, 1388, 1482, 1513, 1624	__draw_softpath_use: 679, 1326
\g__draw_softpath_main_tl	__draw_split_select:nw 1675
. 755, 1320, 1324, 1328, 1333, 1429, 1635	\l__draw_stroke_color_tl 1265
\l__draw_softpath_main_tl	\l__draw_tmp_box 11, 35, 46,
. 19, 755, 767, 1408,	50, 52, 53, 54, 55, 61, 63, 64, 65, 66, 67
1423, 1450, 1452, 1631, 1633, 1636	\l__draw_tmp_seq 1649
\g__draw_softpath_move_bool	__draw_transform_triangle:nnnnnn
. 1338, 1367 1819, 1824
\l__draw_softpath_move_tl	__draw_transform:nnnn 1748
. 1411, 1428,	__draw_transform_invert:n . . . 1780
1451, 1454, 1502, 1602, 1625, 1632	__draw_transform_rotate:n . . . 1850
__draw_softpath_moveto:nn 247, 1340	__draw_transform_rotate:nn . . 1850
__draw_softpath_moveto_op:nn . . .	__draw_transform_shift:nn . . . 1748
. 1366, 1388, 1448, 1604	__draw_transform_shift_absolute:nn
\l__draw_softpath_part_tl 1722
. 1409, 1424,	__draw_vec:nn 1080
1453, 1456, 1458, 1492, 1545, 1634	__draw_vec:nnn 1080
__draw_softpath_rectangle:nnnn .	\g__draw_xmax_dim 194,
. 569, 1340	195, 1200, 1210, 1245, 1260, 1289, 1297
__draw_softpath_rectangle_-	\l__draw_xmax_dim . . . 1282, 1289, 1297
opi:nn 1377, 1388	\g__draw_xmin_dim
__draw_softpath_rectangle_- 196, 197, 1200, 1211,
opi:nnNnn 1388	1243, 1246, 1252, 1260, 1290, 1298
__draw_softpath_rectangle_-	\l__draw_xmin_dim . . . 1282, 1290, 1298
opii:nn 1378, 1388	\l__draw_xshift_dim 48, 1155,
__draw_softpath_round_action:nn	1169, 1702, 1717, 1745, 1777, 1811
. 1418	\l__draw_xvec_x_dim
__draw_softpath_round_action:Nnn 1074, 1104, 1118, 1136
. 1418	\l__draw_xvec_y_dim . . 1074, 1105, 1122
__draw_softpath_round_action_-	\g__draw_ymax_dim 198,
close: 1418	199, 1200, 1212, 1247, 1257, 1291, 1299
__draw_softpath_round_action_-	\l__draw_ymax_dim . . . 1282, 1291, 1299
curveto:NnnNnn 1418	\g__draw_ymin_dim . . 200, 201, 1200,
__draw_softpath_round_calc:nnnNnn	1213, 1248, 1253, 1257, 1292, 1300
. 1418	\l__draw_ymin_dim . . . 1282, 1292, 1300
__draw_softpath_round_calc:nnnnnn	\l__draw_yshift_dim 49, 1161,
. 1418	1169, 1702, 1718, 1746, 1778, 1812
__draw_softpath_round_calc:nnnnw	\l__draw_yvec_x_dim . . 1074, 1104, 1119
. 1418	\l__draw_yvec_y_dim
__draw_softpath_round_close:nn 1418 1074, 1105, 1123, 1137
__draw_softpath_round_close:w 1418	\l__draw_zvec_x_dim 1074, 1120
	\l__draw_zvec_y_dim 1074, 1124

