

File I

Implementation

1 l3draw implementation

```

1 <*initex | package>
2 <@@=draw>
3 <*package>
4 \ProvidesExplPackage{l3draw}{2018/03/05}{}
5   {L3 Experimental core drawing support}
6 </package>
7 \RequirePackage { l3color }
8
   Everything else is in the sub-files!
8 </initex | package>

```

2 l3draw-paths implementation

```

9 <*initex | package>
10 <@@=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 11 \tl_new:N \l__draw_path_tmp_tl
\l__draw_path_tmpb_fp 12 \fp_new:N \l__draw_path_tmpa_fp
                      13 \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`.)

2.1 Tracking paths

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

`\g__draw_path_lasty_dim`
`14 \dim_new:N \g__draw_path_lastx_dim`
`15 \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`
`16 \dim_new:N \g__draw_path_xmax_dim`
`\g__draw_path_ymax_dim`
`17 \dim_new:N \g__draw_path_xmin_dim`
`\g__draw_path_ymin_dim`
`18 \dim_new:N \g__draw_path_ymax_dim`
`19 \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:`
`20 \cs_new_protected:Npn __draw_path_update_limits:nn #1#2`
`21 {`
`22 \dim_gset:Nn \g__draw_path_xmax_dim`
`23 { \dim_max:nn \g__draw_path_xmax_dim {#1} }`
`24 \dim_gset:Nn \g__draw_path_xmin_dim`
`25 { \dim_min:nn \g__draw_path_xmin_dim {#1} }`
`26 \dim_gset:Nn \g__draw_path_ymax_dim`
`27 { \dim_max:nn \g__draw_path_ymax_dim {#2} }`
`28 \dim_gset:Nn \g__draw_path_ymin_dim`
`29 { \dim_min:nn \g__draw_path_ymin_dim {#2} }`
`30 \bool_if:NT \l_draw_bb_update_bool`
`31 {`
`32 \dim_gset:Nn \g__draw_xmax_dim`
`33 { \dim_max:nn \g__draw_xmax_dim {#1} }`
`34 \dim_gset:Nn \g__draw_xmin_dim`
`35 { \dim_min:nn \g__draw_xmin_dim {#1} }`
`36 \dim_gset:Nn \g__draw_ymax_dim`
`37 { \dim_max:nn \g__draw_ymax_dim {#2} }`
`38 \dim_gset:Nn \g__draw_ymin_dim`
`39 { \dim_min:nn \g__draw_ymin_dim {#2} }`
`40 }`
`41 }`
`42 \cs_new_protected:Npn __draw_path_reset_limits:`
`43 {`
`44 \dim_gset:Nn \g__draw_path_xmax_dim { -\c_max_dim }`
`45 \dim_gset:Nn \g__draw_path_xmin_dim { \c_max_dim }`
`46 \dim_gset:Nn \g__draw_path_ymax_dim { -\c_max_dim }`
`47 \dim_gset:Nn \g__draw_path_ymin_dim { \c_max_dim }`
`48 }`

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

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

`49 \cs_new_protected:Npn __draw_path_update_last:nn #1#2`
`50 {`
`51 \dim_gset:Nn \g__draw_path_lastx_dim {#1}`
`52 \dim_gset:Nn \g__draw_path_lasty_dim {#2}`
`53 }`

(End definition for _draw_path_update_last:nn.)

2.2 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 54 \dim_new:N \l__draw_corner_xarc_dim
55 \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.

```
56 \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.

```
57 \cs_new_protected:Npn \draw_path_corner_arc:nn #1#2
58 {
59   \dim_set:Nn \l__draw_corner_xarc_dim {#1}
60   \dim_set:Nn \l__draw_corner_yarc_dim {#2}
61   \bool_lazy_and:nnTF
62     { \dim_compare_p:nNn \l__draw_corner_xarc_dim = { 0pt } }
63     { \dim_compare_p:nNn \l__draw_corner_yarc_dim = { 0pt } }
64     { \bool_set_false:N \l__draw_corner_arc_bool }
65     { \bool_set_true:N \l__draw_corner_arc_bool }
66 }
```

(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.

```
67 \cs_new_protected:Npn \__draw_path_mark_corner:
68 {
69   \bool_if:NT \l__draw_corner_arc_bool
70   {
71     \__draw_softpath_roundpoint:VV
72       \l__draw_corner_xarc_dim
73       \l__draw_corner_yarc_dim
74   }
75 }
```

(End definition for __draw_path_mark_corner:.)

2.3 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
76 \cs_new_protected:Npn \draw_path_moveto:n #1
77 {
78   \__draw_point_process:nn
79   { \__draw_path_moveto:nn }
80   { \draw_point_transform:n {#1} }
81 }
82 \cs_new_protected:Npn \__draw_path_moveto:nn #1#2
83 {
84   \__draw_path_update_limits:nn {#1} {#2}
85   \__draw_softpath_moveto:nn {#1} {#2}
86   \__draw_path_update_last:nn {#1} {#2}
87 }
88 \cs_new_protected:Npn \draw_path_lineto:n #1
89 {
90   \__draw_point_process:nn
91   { \__draw_path_lineto:nn }
92   { \draw_point_transform:n {#1} }
93 }
94 \cs_new_protected:Npn \__draw_path_lineto:nn #1#2
95 {
96   \__draw_path_mark_corner:
97   \__draw_path_update_limits:nn {#1} {#2}
98   \__draw_softpath_lineto:nn {#1} {#2}
99   \__draw_path_update_last:nn {#1} {#2}
100 }
101 \cs_new_protected:Npn \draw_path_curveto:nnn #1#2#3
102 {
103   \__draw_point_process:nnn
104   {
105     \__draw_point_process:nn
106     {
107       \__draw_path_mark_corner:
108       \__draw_path_curveto:nnnnnn
109     }
110     { \draw_point_transform:n {#1} }
111   }
112   { \draw_point_transform:n {#2} }
113   { \draw_point_transform:n {#3} }
114 }
115 \cs_new_protected:Npn \__draw_path_curveto:nnnnnn #1#2#3#4#5#6
116 {
117   \__draw_path_update_limits:nn {#1} {#2}
118   \__draw_path_update_limits:nn {#3} {#4}
119   \__draw_path_update_limits:nn {#5} {#6}
120   \__draw_softpath_curveto:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
121   \__draw_path_update_last:nn {#5} {#6}
122 }

```

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

`\draw_path_close:` A simple wrapper.

```

123 \cs_new_protected:Npn \draw_path_close:
124 {
125     \__draw_path_mark_corner:
126     \__draw_softpath_closepath:
127 }

```

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

2.4 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
128 \cs_new_protected:Npn \draw_path_canvas_moveto:n #1
129 { \__draw_point_process:nn { \__draw_path_moveto:nn } {#1} }
130 \cs_new_protected:Npn \draw_path_canvas_lineto:n #1
131 { \__draw_point_process:nn { \__draw_path_lineto:nn } {#1} }
132 \cs_new_protected:Npn \draw_path_canvas_curveto:nnn #1#2#3
133 {
134     \__draw_point_process:nnn
135     {
136         \__draw_point_process:nn
137         {
138             \__draw_path_mark_corner:
139             \__draw_path_curveto:nnnnnn
140         }
141         {#1}
142     }
143     {#2} {#3}
144 }

```

(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 ??.)

2.5 Computed curves

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

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

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

`\c__draw_path_curveto_a_fp`
`\c__draw_path_curveto_b_fp`

and

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

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

```

145 \cs_new_protected:Npn \draw_path_curveto:nn #1#2
146 {
147     \__draw_point_process:nnn
148     { \__draw_path_curveto:nnnn }
149     { \draw_point_transform:n {#1} }
150     { \draw_point_transform:n {#2} }
151 }

```

```

152 \cs_new_protected:Npn \__draw_path_curveto:nnnn #1#2#3#4
153 {
154   \fp_set:Nn \l__draw_path_tmpa_fp { \c__draw_path_curveto_b_fp * #1 }
155   \fp_set:Nn \l__draw_path_tmpb_fp { \c__draw_path_curveto_b_fp * #2 }
156   \use:x
157   {
158     \__draw_path_mark_corner:
159     \__draw_path_curveto:nnnnnn
160     {
161       \fp_to_dim:n
162       {
163         \c__draw_path_curveto_a_fp * \g__draw_path_lastx_dim
164         + \l__draw_path_tmpa_fp
165       }
166     }
167     {
168       \fp_to_dim:n
169       {
170         \c__draw_path_curveto_a_fp * \g__draw_path_lasty_dim
171         + \l__draw_path_tmpb_fp
172       }
173     }
174     {
175       \fp_to_dim:n
176       { \c__draw_path_curveto_a_fp * #3 + \l__draw_path_tmpa_fp }
177     }
178     {
179       \fp_to_dim:n
180       { \c__draw_path_curveto_a_fp * #4 + \l__draw_path_tmpb_fp }
181     }
182     {#3}
183     {#4}
184   }
185 }
186 \fp_const:Nn \c__draw_path_curveto_a_fp { 1 / 3 }
187 \fp_const:Nn \c__draw_path_curveto_b_fp { 2 / 3 }

```

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

<pre> \draw_path_arc:nnn \draw_path_arc:nnnn __draw_path_arc:nnnn __draw_path_arc:nnNnn __draw_path_arc_auxi:nnnnNnn __draw_path_arc_auxi:fnnnNnn __draw_path_arc_auxi:fnfnNnn __draw_path_arc_auxii:nnnNnnnn __draw_path_arc_auxiii:nn __draw_path_arc_auxiv:nnnn __draw_path_arc_auxv:nn __draw_path_arc_auxvi:nn __draw_path_arc_add:nnnn \l__draw_path_arc_delta_fp \l__draw_path_arc_start_fp \c__draw_path_arc_90_fp \c__draw_path_arc_60_fp </pre>	<p>Drawing an arc means dividing the total curve required into sections: using B�lzier curves we can cover at most 90� at once. To allow for later manipulations, we aim to have roughly equal last segments to the line, with the split set at a final part of 115�.</p> <pre> 188 \cs_new_protected:Npn \draw_path_arc:nnn #1#2#3 189 { \draw_path_arc:nnnn {#1} {#2} {#3} {#3} } 190 \cs_new_protected:Npn \draw_path_arc:nnnn #1#2#3#4 191 { 192 \use:x 193 { 194 __draw_path_arc:nnnn 195 { \fp_eval:n {#1} } 196 { \fp_eval:n {#2} } 197 { \fp_to_dim:n {#3} } 198 { \fp_to_dim:n {#4} } 199 } </pre>
--	--

```

200 }
201 \cs_new_protected:Npn \__draw_path_arc:nnnn #1#2#3#4
202 {
203   \fp_compare:nNnTF {#1} > {#2}
204   { \__draw_path_arc:nnNnn {#1} {#2} - {#3} {#4} }
205   { \__draw_path_arc:nnNnn {#1} {#2} + {#3} {#4} }
206 }
207 \cs_new_protected:Npn \__draw_path_arc:nnNnn #1#2#3#4#5
208 {
209   \fp_set:Nn \l__draw_path_arc_start_fp {#1}
210   \fp_set:Nn \l__draw_path_arc_delta_fp { abs( #1 - #2 ) }
211   \fp_while_do:nNnn { \l__draw_path_arc_delta_fp } > { 90 }
212   {
213     \fp_compare:nNnTF \l__draw_path_arc_delta_fp > { 115 }
214     {
215       \__draw_path_arc_auxi:ffnnNnn
216       { \fp_to_decimal:N \l__draw_path_arc_start_fp }
217       { \fp_eval:n { \l__draw_path_arc_start_fp #3 90 } }
218       { 90 } {#2}
219       #3 {#4} {#5}
220     }
221     {
222       \__draw_path_arc_auxi:ffnnNnn
223       { \fp_to_decimal:N \l__draw_path_arc_start_fp }
224       { \fp_eval:n { \l__draw_path_arc_start_fp #3 60 } }
225       { 60 } {#2}
226       #3 {#4} {#5}
227     }
228   }
229   \__draw_path_mark_corner:
230   \__draw_path_arc_auxi:fnfnNnn
231   { \fp_to_decimal:N \l__draw_path_arc_start_fp }
232   {#2}
233   { \fp_eval:n { abs( \l__draw_path_arc_start_fp - #2 ) } }
234   {#2}
235   #3 {#4} {#5}
236 }

```

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: $\frac{4}{3}(\sqrt{2} - 1) = 0.552\,284\,75$. For other cases, we follow the calculation used by `pgf` but with the second common case of 60° pre-calculated for speed.

```

237 \cs_new_protected:Npn \__draw_path_arc_auxi:nnnnNnn #1#2#3#4#5#6#7
238 {
239   \use:x
240   {
241     \__draw_path_arc_auxii:nnnnNnnnn
242     {#1} {#2} {#4} #5 {#6} {#7}
243     {
244       \fp_to_dim:n
245       {
246         \cs_if_exist_use:cF
247         { c__draw_path_arc_ #3 _fp }
248         { 4/3 * tand( 0.25 * #3 ) }

```

```

249         * #6
250     }
251 }
252 {
253     \fp_to_dim:n
254     {
255         \cs_if_exist_use:cF
256         { c__draw_path_arc_ #3 _fp }
257         { 4/3 * tand( 0.25 * #3 ) }
258         * #7
259     }
260 }
261 }
262 }
263 \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

```

264 \cs_new_protected:Npn \__draw_path_arc_auxii:nnnnNnnnn #1#2#3#4#5#6#7#8
265 {
266     \tl_clear:N \l__draw_path_tmp_tl
267     \__draw_point_process:nn
268     { \__draw_path_arc_auxiii:nn }
269     {
270         \__draw_point_transform_noshift:n
271         { \draw_point_polar:nnn { #1 #4 90 } {#7} {#8} }
272     }
273     \__draw_point_process:nn
274     {
275         \__draw_point_process:nn
276         { \__draw_path_arc_auxiv:nnnn }
277         {
278             \draw_point_transform:n
279             { \draw_point_polar:nnn {#1} {#5} {#6} }
280         }
281     }
282     {
283         \draw_point_transform:n
284         { \draw_point_polar:nnn {#2} {#5} {#6} }
285     }
286     \__draw_point_process:nn
287     { \__draw_path_arc_auxv:nn }
288     {
289         \__draw_point_transform_noshift:n
290         { \draw_point_polar:nnn { #2 #4 -90 } {#7} {#8} }
291     }
292     \exp_after:wN \__draw_path_curveto:nnnnnn \l__draw_path_tmp_tl
293     \fp_set:Nn \l__draw_path_arc_delta_fp { abs ( #2 - #3 ) }
294     \fp_set:Nn \l__draw_path_arc_start_fp {#2}
295 }

```

The first control point.


```

296 \cs_new_protected:Npn \__draw_path_arc_auxiii:nn #1#2
297 {
298   \__draw_path_arc_aux_add:nn
299   { \g__draw_path_lastx_dim + #1 }
300   { \g__draw_path_lasty_dim + #2 }
301 }

```

The end point: simple arithmetic.

```

302 \cs_new_protected:Npn \__draw_path_arc_auxiv:nnnn #1#2#3#4
303 {
304   \__draw_path_arc_aux_add:nn
305   { \g__draw_path_lastx_dim - #1 + #3 }
306   { \g__draw_path_lasty_dim - #2 + #4 }
307 }

```

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

```

308 \cs_new_protected:Npn \__draw_path_arc_auxv:nn #1#2
309 {
310   \exp_after:wN \__draw_path_arc_auxvi:nn
311   \l__draw_path_tmp_tl {#1} {#2}
312 }
313 \cs_new_protected:Npn \__draw_path_arc_auxvi:nn #1#2#3#4#5#6
314 {
315   \tl_set:Nn \l__draw_path_tmp_tl { {#1} {#2} }
316   \__draw_path_arc_aux_add:nn
317   { #5 + #3 }
318   { #6 + #4 }
319   \tl_put_right:Nn \l__draw_path_tmp_tl { {#3} {#4} }
320 }
321 \cs_new_protected:Npn \__draw_path_arc_aux_add:nn #1#2
322 {
323   \tl_put_right:Nx \l__draw_path_tmp_tl
324   { { \fp_to_dim:n {#1} } { \fp_to_dim:n {#2} } }
325 }
326 \fp_new:N \l__draw_path_arc_delta_fp
327 \fp_new:N \l__draw_path_arc_start_fp
328 \fp_const:cn { c__draw_path_arc_90_fp } { 4/3 * (sqrt(2) - 1) }
329 \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.

```

330 \cs_new_protected:Npn \draw_path_arc_axes:nnnn #1#2#3#4
331 {
332   \draw_transform_triangle:nnn { 0cm , 0cm } {#3} {#4}
333   \draw_path_arc:nnn {#1} {#2} { 1pt }
334 }

```

(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:nnnnnn
  \__draw_path_ellipse_arci:nnnnnn
  \__draw_path_ellipse_arcii:nnnnnn
  \__draw_path_ellipse_arciiii:nnnnnn
  \__draw_path_ellipse_arciv:nnnnnn
\c__draw_path_ellipse_fp

```

```

335 \cs_new_protected:Npn \draw_path_ellipse:nnn #1#2#3

```

```

336 {
337   \__draw_point_process:nnn
338   {
339     \__draw_point_process:nn
340     { \__draw_path_ellipse:nnnnnn }
341     { \draw_point_transform:n {#1} }
342   }
343   { \__draw_point_transform_noshift:n {#2} }
344   { \__draw_point_transform_noshift:n {#3} }
345 }
346 \cs_new_protected:Npn \__draw_path_ellipse:nnnnnn #1#2#3#4#5#6
347 {
348   \use:x
349   {
350     \__draw_path_moveto:nn
351     { \fp_to_dim:n { #1 + #3 } } { \fp_to_dim:n { #2 + #4 } }
352     \__draw_path_ellipse_arci:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
353     \__draw_path_ellipse_arcii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
354     \__draw_path_ellipse_arciiii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
355     \__draw_path_ellipse_arciv:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
356   }
357   \__draw_softpath_closepath:
358   \__draw_path_moveto:nn {#1} {#2}
359 }
360 \cs_new:Npn \__draw_path_ellipse_arci:nnnnnn #1#2#3#4#5#6
361 {
362   \__draw_path_curveto:nnnnnn
363   { \fp_to_dim:n { #1 + #3 + #5 * \c__draw_path_ellipse_fp } }
364   { \fp_to_dim:n { #2 + #4 + #6 * \c__draw_path_ellipse_fp } }
365   { \fp_to_dim:n { #1 + #3 * \c__draw_path_ellipse_fp + #5 } }
366   { \fp_to_dim:n { #2 + #4 * \c__draw_path_ellipse_fp + #6 } }
367   { \fp_to_dim:n { #1 + #5 } }
368   { \fp_to_dim:n { #2 + #6 } }
369 }
370 \cs_new:Npn \__draw_path_ellipse_arcii:nnnnnn #1#2#3#4#5#6
371 {
372   \__draw_path_curveto:nnnnnn
373   { \fp_to_dim:n { #1 - #3 * \c__draw_path_ellipse_fp + #5 } }
374   { \fp_to_dim:n { #2 - #4 * \c__draw_path_ellipse_fp + #6 } }
375   { \fp_to_dim:n { #1 - #3 + #5 * \c__draw_path_ellipse_fp } }
376   { \fp_to_dim:n { #2 - #4 + #6 * \c__draw_path_ellipse_fp } }
377   { \fp_to_dim:n { #1 - #3 } }
378   { \fp_to_dim:n { #2 - #4 } }
379 }
380 \cs_new:Npn \__draw_path_ellipse_arciiii:nnnnnn #1#2#3#4#5#6
381 {
382   \__draw_path_curveto:nnnnnn
383   { \fp_to_dim:n { #1 - #3 - #5 * \c__draw_path_ellipse_fp } }
384   { \fp_to_dim:n { #2 - #4 - #6 * \c__draw_path_ellipse_fp } }
385   { \fp_to_dim:n { #1 - #3 * \c__draw_path_ellipse_fp - #5 } }
386   { \fp_to_dim:n { #2 - #4 * \c__draw_path_ellipse_fp - #6 } }
387   { \fp_to_dim:n { #1 - #5 } }
388   { \fp_to_dim:n { #2 - #6 } }
389 }

```

```

390 \cs_new:Npn \__draw_path_ellipse_arciv:nnnnnn #1#2#3#4#5#6
391 {
392   \__draw_path_curveto:nnnnnn
393   { \fp_to_dim:n { #1 + #3 * \c__draw_path_ellipse_fp - #5 } }
394   { \fp_to_dim:n { #2 + #4 * \c__draw_path_ellipse_fp - #6 } }
395   { \fp_to_dim:n { #1 + #3 - #5 * \c__draw_path_ellipse_fp } }
396   { \fp_to_dim:n { #2 + #4 - #6 * \c__draw_path_ellipse_fp } }
397   { \fp_to_dim:n { #1 + #3 } }
398   { \fp_to_dim:n { #2 + #4 } }
399 }
400 \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.

```

401 \cs_new_protected:Npn \draw_path_circle:nn #1#2
402 { \draw_path_ellipse:nnn {#1} { #2 , Opt } { Opt , #2 } }

```

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

2.6 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
403 \cs_new_protected:Npn \draw_path_rectangle:nn #1#2
404 {
405   \__draw_point_process:nnn
406   {
407     \bool_lazy_or:nnTF
408     { \l__draw_corner_arc_bool }
409     { \l__draw_matrix_active_bool }
410     { \__draw_path_rectangle_rounded:nnnn }
411     { \__draw_path_rectangle:nnnn }
412   }
413   { \draw_point_transform:n {#1} }
414   {#2}
415 }
416 \cs_new_protected:Npn \__draw_path_rectangle:nnnn #1#2#3#4
417 {
418   \__draw_path_update_limits:nn {#1} {#2}
419   \__draw_path_update_limits:nn { #1 + #3 } { #2 + #4 }
420   \__draw_softpath_rectangle:nnnn {#1} {#2} {#3} {#4}
421   \__draw_path_update_last:nn {#1} {#2}
422 }
423 \cs_new_protected:Npn \__draw_path_rectangle_rounded:nnnn #1#2#3#4
424 {
425   \draw_path_moveto:n { #1 + #3 , #2 + #4 }
426   \draw_path_lineto:n { #1 , #2 + #4 }
427   \draw_path_lineto:n { #1 , #2 }
428   \draw_path_lineto:n { #1 + #3 , #2 }
429   \draw_path_close:
430   \draw_path_moveto:n { #1 , #2 }
431 }

```

(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`

```

432 \cs_new_protected:Npn \draw_path_rectangle_corners:nn #1#2
433 {
434   \__draw_point_process:nnn
435   { \__draw_path_rectangle_corners:nnnnn {#1} }
436   {#1} {#2}
437 }
438 \cs_new_protected:Npn \__draw_path_rectangle_corners:nnnnn #1#2#3#4#5
439 { \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 ??.)

2.7 Grids

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

```

\__draw_path_grid_auxi:nnnnnn
\__draw_path_grid_auxi:ffnnnn
\__draw_path_grid_auxii:nnnnnn
\__draw_path_grid_auxiii:nnnnnn
\__draw_path_grid_auxiiii:ffnnnn
\__draw_path_grid_auxiv:nnnnnnnn
\__draw_path_grid_auxiv:ffnnnnnn

440 \cs_new_protected:Npn \draw_path_grid:nnnn #1#2#3#4
441 {
442   \__draw_point_process:nnn
443   {
444     \__draw_path_grid_auxi:ffnnnn
445     { \dim_eval:n { \dim_abs:n {#1} } } }
446     { \dim_eval:n { \dim_abs:n {#2} } } }
447   }
448   {#3} {#4}
449 }
450 \cs_new_protected:Npn \__draw_path_grid_auxi:nnnnnn #1#2#3#4#5#6
451 {
452   \dim_compare:nNnTF {#3} > {#5}
453   { \__draw_path_grid_auxii:nnnnnn {#1} {#2} {#5} {#4} {#3} {#6} }
454   { \__draw_path_grid_auxii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6} }
455 }
456 \cs_generate_variant:Nn \__draw_path_grid_auxi:nnnnnn { ff }
457 \cs_new_protected:Npn \__draw_path_grid_auxii:nnnnnn #1#2#3#4#5#6
458 {
459   \dim_compare:nNnTF {#4} > {#6}
460   { \__draw_path_grid_auxiii:nnnnnn {#1} {#2} {#3} {#6} {#5} {#4} }
461   { \__draw_path_grid_auxiii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6} }
462 }
463 \cs_new_protected:Npn \__draw_path_grid_auxiii:nnnnnn #1#2#3#4#5#6
464 {
465   \__draw_path_grid_auxiv:ffnnnnnn
466   { \fp_to_dim:n { #1 * trunc(#3/(#1)) } } }
467   { \fp_to_dim:n { #2 * trunc(#4/(#2)) } } }
468   {#1} {#2} {#3} {#4} {#5} {#6}
469 }
470 \cs_new_protected:Npn \__draw_path_grid_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
471 {
472   \dim_step_inline:nnnn
473   {#1}
```

```

474     {#3}
475     {#7}
476     {
477         \draw_path_moveto:n { ##1 , #6 }
478         \draw_path_lineto:n { ##1 , #8 }
479     }
480     \dim_step_inline:nnnn
481     {#2}
482     {#4}
483     {#8}
484     {
485         \draw_path_moveto:n { #5 , ##1 }
486         \draw_path_lineto:n { #7 , ##1 }
487     }
488 }
489 \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 ??.)

2.8 Using paths

Actions to pass to the driver.

```

\l__draw_path_use_clip_bool
\l__draw_path_use_fill_bool
\l__draw_path_use_stroke_bool
490 \bool_new:N \l__draw_path_use_clip_bool
491 \bool_new:N \l__draw_path_use_fill_bool
492 \bool_new:N \l__draw_path_use_stroke_bool

```

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

Actions handled at the macro layer.

```

\l__draw_path_use_bb_bool
\l__draw_path_use_clear_bool
493 \bool_new:N \l__draw_path_use_bb_bool
494 \bool_new:N \l__draw_path_use_clear_bool

```

(End definition for \l__draw_path_use_bb_bool and \l__draw_path_use_clear_bool.)

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

```

\draw_path_use:n
\draw_path_use_clear:n
\__draw_path_use:n
\__draw_path_use_action_draw:
\__draw_path_use_action_fillstroke:
\__draw_path_use_stroke_bb:
\__draw_path_use_stroke_bb_aux:NnN
495 \cs_new_protected:Npn \draw_path_use:n #1
496 {
497     \tl_if_blank:nF {#1}
498     { \__draw_path_use:n {#1} }
499 }
500 \cs_new_protected:Npn \draw_path_use_clear:n #1
501 {
502     \bool_lazy_or:nnTF
503     { \tl_if_blank_p:n {#1} }
504     { \str_if_eq_p:nn {#1} { clear } }
505     {
506         \__draw_softpath_clear:
507         \__draw_path_reset_limits:
508     }
509     { \__draw_path_use:n { #1 , clear } }
510 }

```

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).

```

511 \cs_new_protected:Npn \__draw_path_use:n #1
512 {
513   \bool_set_false:N \l__draw_path_use_clip_bool
514   \bool_set_false:N \l__draw_path_use_fill_bool
515   \bool_set_false:N \l__draw_path_use_stroke_bool
516   \clist_map_inline:nn {#1}
517   {
518     \cs_if_exist:CTF { l__draw_path_use_ ##1 _ bool }
519     { \bool_set_true:c { l__draw_path_use_ ##1 _ bool } }
520     {
521       \cs_if_exist_use:cF { __draw_path_use_action_ ##1 : }
522       { \ERROR }
523     }
524   }
525   \__draw_softpath_round_corners:
526   \bool_lazy_and:nnT
527   { \l_draw_bb_update_bool }
528   { \l__draw_path_use_stroke_bool }
529   { \__draw_path_use_stroke_bb: }
530   \bool_if:NTF \l__draw_path_use_clear_bool
531   { \__draw_softpath_use_clear: }
532   { \__draw_softpath_use: }
533   \bool_if:NT \l__draw_path_use_clip_bool
534   { \driver_draw_clip: }
535   \bool_lazy_or:nnT
536   { \l__draw_path_use_fill_bool }
537   { \l__draw_path_use_stroke_bool }
538   {
539     \use:c
540     {
541       driver_draw_
542       \bool_if:NT \l__draw_path_use_fill_bool { fill }
543       \bool_if:NT \l__draw_path_use_stroke_bool { stroke }
544       :
545     }
546   }
547 }
548 \cs_new_protected:Npn \__draw_path_use_action_draw:
549 {
550   \bool_set_true:N \l__draw_path_use_stroke_bool
551 }
552 \cs_new_protected:Npn \__draw_path_use_action_fillstroke:
553 {
554   \bool_set_true:N \l__draw_path_use_fill_bool
555   \bool_set_true:N \l__draw_path_use_stroke_bool
556 }

```

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.

```

557 \cs_new_protected:Npn \__draw_path_use_stroke_bb:
558 {

```

```

559 \__draw_path_use_stroke_bb_aux:NnN x { max } +
560 \__draw_path_use_stroke_bb_aux:NnN y { max } +
561 \__draw_path_use_stroke_bb_aux:NnN x { min } -
562 \__draw_path_use_stroke_bb_aux:NnN y { min } -
563 }
564 \cs_new_protected:Npn \__draw_path_use_stroke_bb_aux:NnN #1#2#3
565 {
566   \dim_compare:nNnF { \dim_use:c { g__draw_ #1#2 _dim } } = { #3 -\c_max_dim }
567   {
568     \dim_gset:cn { g__draw_ #1#2 _dim }
569     {
570       \use:c { dim_ #2 :nn }
571       { \dim_use:c { g__draw_ #1#2 _dim } }
572       {
573         \dim_use:c { g__draw_path_ #1#2 _dim }
574         #3 0.5 \g__draw_linewidth_dim
575       }
576     }
577   }
578 }

```

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

2.9 Scoping paths

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

```

\l__draw_path_xmax_dim
\l__draw_path_xmin_dim
\l__draw_path_ymax_dim
\l__draw_path_ymin_dim
\l__draw_softpath_corners_bool
579 \dim_new:N \l__draw_path_lastx_dim
580 \dim_new:N \l__draw_path_lasty_dim
581 \dim_new:N \l__draw_path_xmax_dim
582 \dim_new:N \l__draw_path_xmin_dim
583 \dim_new:N \l__draw_path_ymax_dim
584 \dim_new:N \l__draw_path_ymin_dim
585 \dim_new:N \l__draw_softpath_lastx_dim
586 \dim_new:N \l__draw_softpath_lasty_dim
587 \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.

```

\draw_path_scope_end:
588 \cs_new_protected:Npn \draw_path_scope_begin:
589 {
590   \group_begin:
591   \dim_set_eq:NN \l__draw_path_lastx_dim \g__draw_path_lastx_dim
592   \dim_set_eq:NN \l__draw_path_lasty_dim \g__draw_path_lasty_dim
593   \dim_set_eq:NN \l__draw_path_xmax_dim \g__draw_path_xmax_dim
594   \dim_set_eq:NN \l__draw_path_xmin_dim \g__draw_path_xmin_dim
595   \dim_set_eq:NN \l__draw_path_ymax_dim \g__draw_path_ymax_dim
596   \dim_set_eq:NN \l__draw_path_ymin_dim \g__draw_path_ymin_dim
597   \dim_set_eq:NN \l__draw_softpath_lastx_dim \g__draw_softpath_lastx_dim
598   \dim_set_eq:NN \l__draw_softpath_lasty_dim \g__draw_softpath_lasty_dim
599   \__draw_path_reset_limits:

```

```

600     \__draw_softpath_concat:n { }
601     \tl_set_eq:NN \l__draw_softpath_main_tl \g__draw_softpath_main_tl
602     \bool_set_eq:NN
603         \l__draw_softpath_corners_bool
604         \g__draw_softpath_corners_bool
605     \__draw_softpath_clear:
606 }
607 \cs_new_protected:Npn \draw_path_scope_end:
608 {
609     \bool_gset_eq:NN
610         \g__draw_softpath_corners_bool
611         \l__draw_softpath_corners_bool
612     \__draw_softpath_clear:
613     \tl_gset_eq:NN \g__draw_softpath_main_tl \l__draw_softpath_main_tl
614     \dim_gset_eq:NN \g__draw_softpath_lastx_dim \l__draw_softpath_lastx_dim
615     \dim_gset_eq:NN \g__draw_softpath_lasty_dim \l__draw_softpath_lasty_dim
616     \dim_gset_eq:NN \g__draw_path_xmax_dim \l__draw_path_xmax_dim
617     \dim_gset_eq:NN \g__draw_path_xmin_dim \l__draw_path_xmin_dim
618     \dim_gset_eq:NN \g__draw_path_ymax_dim \l__draw_path_ymax_dim
619     \dim_gset_eq:NN \g__draw_path_ymin_dim \l__draw_path_ymin_dim
620     \dim_gset_eq:NN \g__draw_path_lastx_dim \l__draw_path_lastx_dim
621     \dim_gset_eq:NN \g__draw_path_lasty_dim \l__draw_path_lasty_dim
622     \group_end:
623 }

```

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

```

624 \</initex | package>

```

3 l3draw-points implementation

```

625 \<*initex | package>

```

```

626 \<@@=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}`.
- `\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.

3.1 Support functions

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

```

\__draw_point_process:nn
  \__draw_point_process_auxi:nn
    \__draw_point_process_auxi:fn
      \__draw_point_process_auxii:nw
        \__draw_point_process:nnn
          \__draw_point_process_auxiii:nnn
            \__draw_point_process_auxiii:ffn
              \__draw_point_process_auxiv:nw
                627 \cs_new:Npn \__draw_point_process:nn #1#2
                628 {
                629   \__draw_point_process_auxi:fn
                630   { \__draw_point_to_dim:n {#2} }
                631   {#1}
                632 }
                633 \cs_new:Npn \__draw_point_process_auxi:nn #1#2
                634 { \__draw_point_process_auxii:nw {#2} #1 \q_stop }
                635 \cs_generate_variant:Nn \__draw_point_process_auxi:nn { f }
                636 \cs_new:Npn \__draw_point_process_auxii:nw #1 #2 , #3 \q_stop
                637 { #1 {#2} {#3} }
                638 \cs_new:Npn \__draw_point_process:nnn #1#2#3
                639 {
                640   \__draw_point_process_auxiii:ffn
                641   { \__draw_point_to_dim:n {#2} }
                642   { \__draw_point_to_dim:n {#3} }
                643   {#1}
                644 }
                645 \cs_new:Npn \__draw_point_process_auxiii:nnn #1#2#3
                646 { \__draw_point_process_auxiv:nw {#3} #1 \q_mark #2 \q_stop }
                647 \cs_generate_variant:Nn \__draw_point_process_auxiii:nnn { ff }
                648 \cs_new:Npn \__draw_point_process_auxiv:nw #1 #2 , #3 \q_mark #4 , #5 \q_stop
                649 { #1 {#2} {#3} {#4} {#5} }

```

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

```

\__draw_point_to_dim:n
\__draw_point_to_dim_aux:n
\__draw_point_to_dim_aux:f
\__draw_point_to_dim_aux:w
  650 \cs_new:Npn \__draw_point_to_dim:n #1
  651 { \__draw_point_to_dim_aux:f { \fp_eval:n {#1} } }
  652 \cs_new:Npn \__draw_point_to_dim_aux:n #1
  653 { \__draw_point_to_dim_aux:w #1 }
  654 \cs_generate_variant:Nn \__draw_point_to_dim_aux:n { f }
  655 \cs_new:Npn \__draw_point_to_dim_aux:w ( #1 , ~ #2 ) { #1pt , #2pt }

```

3.2 Co-ordinates

The most basic way of giving points is as simple (x, y) co-ordinates.

Simply turn the given values into dimensions.

```

\draw_point:nn
  656 \cs_new:Npn \draw_point:nn #1#2
  657 { \__draw_point_to_dim:n { #1 , #2 } }

```

3.3 Polar co-ordinates

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

```
\draw_point_polar:nn
\draw_point_polar:nnn
__draw_draw_polar:nnn
__draw_draw_polar:fnn
658 \cs_new:Npn \draw_point_polar:nn #1#2
659 { \draw_point_polar:nnn {#1} {#2} {#2} }
660 \cs_new:Npn \draw_point_polar:nnn #1#2#3
661 { __draw_draw_polar:fnn { \fp_eval:n {#1} } {#2} {#3} }
662 \cs_new:Npn __draw_draw_polar:nnn #1#2#3
663 { __draw_point_to_dim:n { cosd(#1) * (#2) , sind(#1) * (#3) } }
664 \cs_generate_variant:Nn __draw_draw_polar:nnn { f }
```

3.4 Point expression arithmetic

These functions all take point expressions as arguments.

Simple mathematics.

```
\draw_point_add:nn
\draw_point_diff:nn
\draw_point_scale:nn
665 \cs_new:Npn \draw_point_add:nn #1#2
666 { __draw_point_to_dim:n { (#1) + (#2) } }
667 \cs_new:Npn \draw_point_diff:nn #1#2
668 { __draw_point_to_dim:n { (#2) - (#1) } }
669 \cs_new:Npn \draw_point_scale:nn #1#2
670 { __draw_point_to_dim:n { #1 * (#2) } }
```

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.*

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

```
671 \cs_new:Npn \draw_point_unit_vector:n #1
672 { __draw_point_process:nn { __draw_point_unit_vector:nn } {#1} }
673 \cs_new:Npn __draw_point_unit_vector:nn #1#2
674 {
675   __draw_point_to_dim:n
676   { ( #1 , #2 ) / (sqrt(#1 * #1 + #2 * #2)) }
677 }
```

3.5 Intersection calculations

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

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

and

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

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

```

678 \cs_new:Npn \draw_point_intersect_lines:nnnn #1#2#3#4
679 {
680   \__draw_point_process:nnn
681   {
682     \__draw_point_process:nnn
683     { \__draw_point_intersect_lines:nnnnnnnn } {#3} {#4}
684   }
685   {#1} {#2}
686 }

```

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

```

#1 x3

#2 y3

#3 x4

#4 y4

#5 x1

#6 y1

#7 x2

#8 y2

```

so now just have to do all of the calculation.

```

687 \cs_new:Npn \__draw_point_intersect_lines:nnnnnnnn #1#2#3#4#5#6#7#8
688 {
689   \__draw_point_intersect_lines_aux:fffff
690   { \fp_eval:n { #1 * #4 - #2 * #3 } }
691   { \fp_eval:n { #5 * #8 - #6 * #7 } }
692   { \fp_eval:n { #1 - #3 } }
693   { \fp_eval:n { #5 - #7 } }
694   { \fp_eval:n { #2 - #4 } }
695   { \fp_eval:n { #6 - #8 } }
696 }
697 \cs_new:Npn \__draw_point_intersect_lines_aux:nnnnnn #1#2#3#4#5#6
698 {
699   \__draw_point_to_dim:n
700   {
701     ( #2 * #3 - #1 * #4 , #2 * #5 - #1 * #6 )
702     / ( #4 * #5 - #6 * #3 )
703   }
704 }
705 \cs_generate_variant:Nn \__draw_point_intersect_lines_aux:nnnnnn { ffffff }

```

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

```

\draw_point_intersect_circles:nnnnn
\__draw_point_intersect_circles_auxi:nnnnnnn
\__draw_point_intersect_circles_auxii:nnnnnnn
\__draw_point_intersect_circles_auxiii:ffnnnnn
\__draw_point_intersect_circles_auxiiii:nnnnnnn
\__draw_point_intersect_circles_auxv:nnnnnnnn
\__draw_point_intersect_circles_auxvi:ffnnnnnnn
\__draw_point_intersect_circles_auxvii:nnnnnnnn
\__draw_point_intersect_circles_auxviii:ffnnnnnn
\__draw_point_intersect_circles_auxix:nnnnnnnn
\__draw_point_intersect_circles_auxx:ffnnnnnnn
\__draw_point_intersect_circles_auxxi:nnnnnnnn
\__draw_point_intersect_circles_auxxii:ffnnnnnn
\__draw_point_intersect_circles_auxxiii:nnnnnnnn
\__draw_point_intersect_circles_auxxiv:ffnnnnnnn
\__draw_point_intersect_circles_auxxv:nnnnnnnn
\__draw_point_intersect_circles_auxxvi:ffnnnnnnn
\__draw_point_intersect_circles_auxxvii:nnnnnnnn
\__draw_point_intersect_circles_auxxviii:ffnnnnnn

```

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.

```

706 \cs_new:Npn \draw_point_intersect_circles:nnnnn #1#2#3#4#5
707 {
708   \__draw_point_process:nnn
709   { \__draw_point_intersect_circles_auxi:nnnnnnn {#2} {#4} {#5} }
710   {#1} {#3}
711 }
712 \cs_new:Npn \__draw_point_intersect_circles_auxi:nnnnnnn #1#2#3#4#5#6#7
713 {
714   \__draw_point_intersect_circles_auxii:ffnnnnnn
715   { \fp_eval:n {#1} } { \fp_eval:n {#2} } {#4} {#5} {#6} {#7} {#3}
716 }

```

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.

```

717 \cs_new:Npn \__draw_point_intersect_circles_auxii:nnnnnnn #1#2#3#4#5#6#7

```

```

718 {
719   \_draw_point_intersect_circles_auxiii:ffnnnnnn
720   { \fp_eval:n { #5 - #3 } }
721   { \fp_eval:n { #6 - #4 } }
722   {#1} {#2} {#3} {#4} {#7}
723 }
724 \cs_generate_variant:Nn \_draw_point_intersect_circles_auxii:nnnnnnnn { ff }
725 \cs_new:Npn \_draw_point_intersect_circles_auxiii:nnnnnnnn #1#2#3#4#5#6#7
726 {
727   \_draw_point_intersect_circles_auxiv:fnnnnnnnn
728   { \fp_eval:n { sqrt( #1 * #1 + #2 * #2 ) } }
729   {#1} {#2} {#3} {#4} {#5} {#6} {#7}
730 }
731 \cs_generate_variant:Nn \_draw_point_intersect_circles_auxiii:nnnnnnnn { 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.

```

732 \cs_new:Npn \_draw_point_intersect_circles_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
733 {
734   \_draw_point_intersect_circles_auxv:ffnnnnnnnn
735   { \fp_eval:n { 1 / #1 } }
736   { \fp_eval:n { #4 * #4 } }
737   {#1} {#2} {#3} {#5} {#6} {#7} {#8}
738 }
739 \cs_generate_variant:Nn \_draw_point_intersect_circles_auxiv:nnnnnnnn { f }
740 \cs_new:Npn \_draw_point_intersect_circles_auxv:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
741 {
742   \_draw_point_intersect_circles_auxvi:fnnnnnnnn
743   { \fp_eval:n { 0.5 * #1 * ( #2 + #3 * #3 - #6 * #6 ) } }
744   {#1} {#2} {#4} {#5} {#7} {#8} {#9}
745 }
746 \cs_generate_variant:Nn \_draw_point_intersect_circles_auxv:nnnnnnnnnn { 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.

```

747 \cs_new:Npn \_draw_point_intersect_circles_auxvi:nnnnnnnnnn #1#2#3#4#5#6#7#8
748 {

```

```

749 \__draw_point_intersect_circles_auxvii:fffnnnn
750 { \fp_eval:n { #1 * #2 } }
751 { \int_if_odd:nTF {#8} { 1 } { -1 } }
752 { \fp_eval:n { sqrt ( #3 - #1 * #1 ) * #2 } }
753 {#4} {#5} {#6} {#7}
754 }
755 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxvi:nnnnnnnn { f }
756 \cs_new:Npn \__draw_point_intersect_circles_auxvii:nnnnnnnn #1#2#3#4#5#6#7
757 {
758 \__draw_point_to_dim:n
759 { #6 + #4 * #1 + #2 * #3 * #5 , #7 + #5 * #1 + -1 * #2 * #3 * #4 }
760 }
761 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxvii:nnnnnnnn { fff }

```

3.6 Interpolation on a line (vector) or arc

Simple maths after expansion.

```

\draw_point_interpolate_line:nnm
\__draw_point_interpolate_line_aux:nnnnn
\__draw_point_interpolate_line_aux:fnnnn
\__draw_point_interpolate_line_aux:nnnnnn
\__draw_point_interpolate_line_aux:fnnnnn
762 \cs_new:Npn \draw_point_interpolate_line:nnn #1#2#3
763 {
764 \__draw_point_process:nnn
765 { \__draw_point_interpolate_line_aux:fnnnn { \fp_eval:n {#1} } }
766 {#2} {#3}
767 }
768 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnn #1#2#3#4#5
769 {
770 \__draw_point_interpolate_line_aux:fnnnnn { \fp_eval:n { 1 - #1 } }
771 {#1} {#2} {#3} {#4} {#5}
772 }
773 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnnn { f }
774 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnnn #1#2#3#4#5#6
775 { \__draw_point_to_dim:n { #2 * #3 + #1 * #5 , #2 * #4 + #1 * #6 } }
776 \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:nnm
\__draw_point_interpolate_distance:nnnnn
\__draw_point_interpolate_distance:nnnnnn
\__draw_point_interpolate_distance:fnnnnn
777 \cs_new:Npn \draw_point_interpolate_distance:nnn #1#2#3
778 {
779 \__draw_point_process:nn
780 { \__draw_point_interpolate_distance:nnnn {#1} {#3} }
781 {#2}
782 }
783 \cs_new:Npn \__draw_point_interpolate_distance:nnnn #1#2#3#4
784 {
785 \__draw_point_process:nn
786 {
787 \__draw_point_interpolate_distance:fnnnn
788 { \fp_eval:n {#1} } {#3} {#4}
789 }
790 { \draw_point_unit_vector:n { ( #2 ) - ( #3 , #4 ) } }
791 }
792 \cs_new:Npn \__draw_point_interpolate_distance:nnnnn #1#2#3#4#5
793 { \__draw_point_to_dim:n { #2 + #1 * #4 , #3 + #1 * #5 } }
794 \cs_generate_variant:Nn \__draw_point_interpolate_distance:nnnnnn { f }

```

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

```
\draw_point_interpolate_arcaxes:nnnnnnn
raw_point_interpolate_arcaxes_auxi:nnnnnnnnn
raw_point_interpolate_arcaxes_auxii:nnnnnnnnn
raw_point_interpolate_arcaxes_auxiii:nnnnnnnnn
raw_point_interpolate_arcaxes_auxiv:nnnnnnnnn
raw_point_interpolate_arcaxes_auxv:nnnnnnnnn
```

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.

```
795 \cs_new:Npn \draw_point_interpolate_arcaxes:nnnnnn #1#2#3#4#5#6
796 {
797   \_draw_point_process:nnn
798   {
799     \_draw_point_process:nn
800     { \fp_eval:n {#1} } {#5} {#6} }
801     {#4}
802   }
803   {#2} {#3}
804 }
805 \cs_new:Npn \_draw_point_interpolate_arcaxes_auxi:nnnnnnnnn #1#2#3#4#5#6#7#8#9
806 {
807   \_draw_point_interpolate_arcaxes_auxii:nnnnnnnnn
808   { \fp_eval:n {#1} } {#2} {#3} {#6} {#7} {#8} {#9} {#4} {#5}
809 }
```

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.

```
810 \cs_new:Npn \_draw_point_interpolate_arcaxes_auxii:nnnnnnnnn #1#2#3#4#5#6#7#8#9
811 {
812   \_draw_point_interpolate_arcaxes_auxiii:nnnnnnnnn
813   { \fp_eval:n { #1 * (#3) + ( 1 - #1 ) * (#2) } }
814   {#4} {#5} {#6} {#7} {#8} {#9}
815 }
816 \cs_generate_variant:Nn \_draw_point_interpolate_arcaxes_auxii:nnnnnnnnn { f }
817 \cs_new:Npn \_draw_point_interpolate_arcaxes_auxiii:nnnnnnnnn #1#2#3#4#5#6#7
818 {
819   \_draw_point_interpolate_arcaxes_auxiv:ffnnnnnnnn
820   { \fp_eval:n { cosd (#1) } }
821   { \fp_eval:n { sind (#1) } }
822   {#2} {#3} {#4} {#5} {#6} {#7}
```

```

823 }
824 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnn { f }
825 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
826 {
827   \__draw_point_to_dim:n
828     { #3 + #1 * #5 + #2 * #7 , #4 + #1 * #6 + #2 * #8 }
829 }
830 \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:nnnnn
\draw_point_interpolate_curve_auxi:nnnnnnnnn
\draw_point_interpolate_curve_auxii:nnnnnnnnn
\draw_point_interpolate_curve_auxiii:nnnnnnnnn
\draw_point_interpolate_curve_auxiiii:nnnnnnnnn
\draw_point_interpolate_curve_auxv:nnnnnnnnn
\draw_point_interpolate_curve_auxvi:nnnnnnnnn
\draw_point_interpolate_curve_auxvii:nnnnnnnnn
\draw_point_interpolate_curve_auxviii:nnnnnnnnn
\draw_point_interpolate_curve_auxviiii:nnnnnnnnn

```

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.

```

831 \cs_new:Npn \draw_point_interpolate_curve:nnnnnn #1#2#3#4#5
832 {
833   \__draw_point_process:nnn
834   {
835     \__draw_point_process:nnn
836     { \__draw_point_interpolate_curve_auxi:nnnnnnnnn {#1} }
837     {#4} {#5}
838   }
839   {#2} {#3}
840 }
841 \cs_new:Npn \__draw_point_interpolate_curve_auxi:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
842 {
843   \__draw_point_interpolate_curve_auxii:nnnnnnnnnn
844   { \fp_eval:n {#1} }
845   {#6} {#7} {#8} {#9} {#2} {#3} {#4} {#5}
846 }

```

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

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

In the second stage, we can drop the final point

$$\begin{aligned}x_1'' &= (1-p)x_1' + px_2' \\ y_1'' &= (1-p)y_1' + py_2' \\ x_2'' &= (1-p)x_2' + px_3' \\ y_2'' &= (1-p)y_2' + py_3'\end{aligned}$$

and for the final stage only need one set of calculations

$$\begin{aligned}P_x &= (1-p)x_1'' + px_2'' \\ P_y &= (1-p)y_1'' + py_2''\end{aligned}$$

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

```

847 \cs_new:Npn \__draw_point_interpolate_curve_auxii:nnnnnnnnn
848   #1#2#3#4#5#6#7#8#9
849   {
850     \__draw_point_interpolate_curve_auxiii:fnnnnnn
851     { \fp_eval:n { 1 - #1 } }
852     {#1}
853     { {#2} {#3} } { {#4} {#5} } { {#6} {#7} } { {#8} {#9} }
854   }
855 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxii:nnnnnnnnn { f }
856 % \begin{macrocode}
857 % We need to do the first cycle, but haven't got enough arguments to keep
858 % everything in play at once. So here we use a bit of argument re-ordering
859 % and a single auxiliary to get the job done.
860 % \begin{macrocode}
861 \cs_new:Npn \__draw_point_interpolate_curve_auxiii:nnnnnn #1#2#3#4#5#6
862   {
863     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #3 #4
864     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #4 #5
865     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #5 #6
866     \prg_do_nothing:
867     \__draw_point_interpolate_curve_auxvi:n { {#1} {#2} }
868   }
869 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxiii:nnnnnn { f }
870 \cs_new:Npn \__draw_point_interpolate_curve_auxiv:nnnnnn #1#2#3#4#5#6
871   {
872     \__draw_point_interpolate_curve_auxv:ffw
873     { \fp_eval:n { #1 * #3 + #2 * #5 } }
874     { \fp_eval:n { #1 * #4 + #2 * #6 } }
875   }
876 \cs_new:Npn \__draw_point_interpolate_curve_auxv:nnw
877   #1#2#3 \prg_do_nothing: #4#5
878   {
879     #3
880     \prg_do_nothing:
881     #4 { #5 {#1} {#2} }
882   }
883 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxv:nnw { ff }
884 % \begin{macrocode}
885 % Get the arguments back into the right places and to the second and
886 % third cycles directly.

```

```

887 % \begin{macrocode}
888 \cs_new:Npn \__draw_point_interpolate_curve_auxvi:n #1
889 { \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1 }
890 \cs_new:Npn \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1#2#3#4#5#6#7#8
891 {
892   \__draw_point_interpolate_curve_auxviii:ffffnn
893   { \fp_eval:n { #1 * #5 + #2 * #3 } }
894   { \fp_eval:n { #1 * #6 + #2 * #4 } }
895   { \fp_eval:n { #1 * #7 + #2 * #5 } }
896   { \fp_eval:n { #1 * #8 + #2 * #6 } }
897   {#1} {#2}
898 }
899 \cs_new:Npn \__draw_point_interpolate_curve_auxviii:nnnnnn #1#2#3#4#5#6
900 {
901   \__draw_point_to_dim:n
902   { #5 * #3 + #6 * #1 , #5 * #4 + #6 * #2 }
903 }
904 \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 ??.)

3.7 Vector support

As well as co-ordinates relative to the drawing

```

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

```

(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
911 \cs_new_protected:Npn \draw_xvec:n #1
912 { \__draw_vec:nn { x } {#1} }
913 \cs_new_protected:Npn \draw_yvec:n #1
914 { \__draw_vec:nn { y } {#1} }
915 \cs_new_protected:Npn \draw_zvec:n #1
916 { \__draw_vec:nn { z } {#1} }
917 \cs_new_protected:Npn \__draw_vec:nn #1#2
918 {
919   \__draw_point_process:nn { \__draw_vec:nnn {#1} } {#2}
920 }
921 \cs_new_protected:Npn \__draw_vec:nnn #1#2#3
922 {
923   \dim_set:cn { l__draw_ #1 vec_x_dim } {#2}
924   \dim_set:cn { l__draw_ #1 vec_y_dim } {#3}
925 }

```

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

Initialise the vectors.

```

926 \draw_xvec:n { 1cm , 0cm }
927 \draw_yvec:n { 0cm , 1cm }
928 \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 929 \cs_new:Npn \draw_point_vec:nn #1#2
\__draw_point_vec:ff 930 { \__draw_point_vec:ff { \fp_eval:n {#1} } { \fp_eval:n {#2} } }
\draw_point_vec:nnn 931 \cs_new:Npn \__draw_point_vec:nn #1#2
\__draw_point_vec:nnn 932 {
\__draw_point_vec:fff 933   \__draw_point_to_dim:n
934   {
935     #1 * \l__draw_xvec_x_dim + #2 * \l__draw_yvec_x_dim ,
936     #1 * \l__draw_xvec_y_dim + #2 * \l__draw_yvec_y_dim
937   }
938 }
939 \cs_generate_variant:Nn \__draw_point_vec:nn { ff }
940 \cs_new:Npn \draw_point_vec:nnn #1#2#3
941 {
942   \__draw_point_vec:fff
943   { \fp_eval:n {#1} } { \fp_eval:n {#2} } { \fp_eval:n {#3} }
944 }
945 \cs_new:Npn \__draw_point_vec:nnn #1#2#3
946 {
947   \__draw_point_to_dim:n
948   {
949     #1 * \l__draw_xvec_x_dim
950     + #2 * \l__draw_yvec_x_dim
951     + #3 * \l__draw_zvec_x_dim
952   ,
953     #1 * \l__draw_xvec_y_dim
954     + #2 * \l__draw_yvec_y_dim
955     + #3 * \l__draw_zvec_y_dim
956   }
957 }
958 \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 959 \cs_new:Npn \draw_point_vec_polar:nn #1#2
\__draw_point_vec_polar:nnn 960 { \draw_point_vec_polar:nnn {#1} {#2} {#2} }
\__draw_point_vec_polar:fnn 961 \cs_new:Npn \draw_point_vec_polar:nnn #1#2#3
962 { \__draw_draw_vec_polar:fnn { \fp_eval:n {#1} } {#2} {#3} }
963 \cs_new:Npn \__draw_draw_vec_polar:nnn #1#2#3
964 {
965   \__draw_point_to_dim:n
966   {
967     cosd(#1) * (#2) * \l__draw_xvec_x_dim ,
968     sind(#1) * (#3) * \l__draw_yvec_y_dim
969   }
970 }
971 \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 ??.)

3.8 Transformations

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

```

972 \cs_new:Npn \draw_point_transform:n #1
973 {
974   \__draw_point_process:nn
975   { \__draw_point_transform:nn } {#1}
976 }
977 \cs_new:Npn \__draw_point_transform:nn #1#2
978 {
979   \bool_if:NTF \l__draw_matrix_active_bool
980   {
981     \__draw_point_to_dim:n
982     {
983       (
984         \l__draw_matrix_a_fp * #1
985         + \l__draw_matrix_c_fp * #2
986         + \l__draw_xshift_dim
987       )
988       ,
989       (
990         \l__draw_matrix_b_fp * #1
991         + \l__draw_matrix_d_fp * #2
992         + \l__draw_yshift_dim
993       )
994     }
995   }
996   {
997     \__draw_point_to_dim:n
998     {
999       (#1, #2)
1000       + ( \l__draw_xshift_dim , \l__draw_yshift_dim )
1001     }
1002   }
1003 }
```

(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.

```

1004 \cs_new:Npn \__draw_point_transform_noshift:n #1
1005 {
1006   \__draw_point_process:nn
1007   { \__draw_point_transform_noshift:nn } {#1}
1008 }
1009 \cs_new:Npn \__draw_point_transform_noshift:nn #1#2
1010 {
1011   \bool_if:NTF \l__draw_matrix_active_bool
1012   {
1013     \__draw_point_to_dim:n
```

```

1014         {
1015             (
1016                 \l__draw_matrix_a_fp * #1
1017             + \l__draw_matrix_c_fp * #2
1018             )
1019             ,
1020             (
1021                 \l__draw_matrix_b_fp * #1
1022             + \l__draw_matrix_d_fp * #2
1023             )
1024         }
1025     }
1026     { \__draw_point_to_dim:n { (#1, #2) } }
1027 }

```

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

```
1028 </initex | package>
```

4 l3draw-scopes implementation

```
1029 <*initex | package>
```

```
1030 <@@=draw>
```

4.1 Drawing environment

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

```

\g__draw_xmin_dim
\g__draw_ymax_dim
\g__draw_ymin_dim
1031 \dim_new:N \g__draw_xmax_dim
1032 \dim_new:N \g__draw_xmin_dim
1033 \dim_new:N \g__draw_ymax_dim
1034 \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.

```
1035 \bool_new:N \l_draw_bb_update_bool
```

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

`\l__draw_main_box` Box for setting the drawing.

```
1036 \box_new:N \l__draw_main_box
```

(End definition for `\l__draw_main_box`.)

`\g__draw_id_int` The drawing number.

```
1037 \int_new:N \g__draw_id_int
```

(End definition for `\g__draw_id_int`.)

`__draw_reset_bb:` A simple auxiliary.

```

1038 \cs_new_protected:Npn \__draw_reset_bb:
1039 {
1040   \dim_gset:Nn \g__draw_xmax_dim { -\c_max_dim }
1041   \dim_gset:Nn \g__draw_xmin_dim { \c_max_dim }
1042   \dim_gset:Nn \g__draw_ymax_dim { -\c_max_dim }
1043   \dim_gset:Nn \g__draw_ymin_dim { \c_max_dim }
1044 }

```

(End definition for `__draw_reset_bb:`.)

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

`\draw_end:`

```

1045 \cs_new_protected:Npn \draw_begin:
1046 {
1047   \group_begin:
1048   \int_gincr:N \g__draw_id_int
1049   \hbox_set:Nw \l__draw_main_box
1050   \driver_draw_begin:
1051   \__draw_reset_bb:
1052   \__draw_path_reset_limits:
1053   \bool_set_true:N \l_draw_bb_update_bool
1054   \draw_transform_matrix_reset:
1055   \draw_transform_shift_reset:
1056   \__draw_softpath_clear:
1057   \draw_linewidth:n { \l_draw_default_linewidth_dim }
1058   \draw_color:n { . }
1059 }
1060 \cs_new_protected:Npn \draw_end:
1061 {
1062   \driver_draw_end:
1063   \hbox_set_end:
1064   \dim_compare:nNnT \g__draw_xmin_dim = \c_max_dim
1065   {
1066     \dim_gzero:N \g__draw_xmax_dim
1067     \dim_gzero:N \g__draw_xmin_dim
1068     \dim_gzero:N \g__draw_ymax_dim
1069     \dim_gzero:N \g__draw_ymin_dim
1070   }
1071   \hbox_set:Nn \l__draw_main_box
1072   {
1073     \skip_horizontal:n { -\g__draw_xmin_dim }
1074     \box_move_down:nn { \g__draw_ymin_dim }
1075     { \box_use_drop:N \l__draw_main_box }
1076   }
1077   \box_set_ht:Nn \l__draw_main_box
1078   { \g__draw_ymax_dim - \g__draw_ymin_dim }
1079   \box_set_dp:Nn \l__draw_main_box { Opt }
1080   \box_set_wd:Nn \l__draw_main_box
1081   { \g__draw_xmax_dim - \g__draw_xmin_dim }
1082   \mode_leave_vertical:

```

```

1083     \box_use_drop:N \l__draw_main_box
1084   \group_end:
1085 }

```

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

4.2 Scopes

`\l__draw_linewidth_dim` Storage for local variables.
`\l__draw_fill_color_tl` 1086 `\dim_new:N \l__draw_linewidth_dim`
`\l__draw_stroke_color_tl` 1087 `\tl_new:N \l__draw_fill_color_tl`
1088 `\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: 1089 \cs_new_protected:Npn \draw_scope_begin:
1090 {
1091   \driver_draw_scope_begin:
1092   \group_begin:
1093     \dim_set_eq:NN \l__draw_linewidth_dim \g__draw_linewidth_dim
1094     \draw_path_scope_begin:
1095   }
1096 \cs_new_protected:Npn \draw_scope_end:
1097 {
1098   \draw_path_scope_end:
1099   \dim_gset_eq:NN \g__draw_linewidth_dim \l__draw_linewidth_dim
1100   \group_end:
1101   \driver_draw_scope_end:
1102 }

```

(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` 1103 `\dim_new:N \l__draw_xmax_dim`
`\l__draw_ymax_dim` 1104 `\dim_new:N \l__draw_xmin_dim`
`\l__draw_ymin_dim` 1105 `\dim_new:N \l__draw_ymax_dim`
1106 `\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: 1107 \cs_new_protected:Npn \__draw_scope_bb_begin:
1108 {
1109   \group_begin:
1110     \dim_set_eq:NN \l__draw_xmax_dim \g__draw_xmax_dim
1111     \dim_set_eq:NN \l__draw_xmin_dim \g__draw_xmin_dim
1112     \dim_set_eq:NN \l__draw_ymax_dim \g__draw_ymax_dim
1113     \dim_set_eq:NN \l__draw_ymin_dim \g__draw_ymin_dim
1114     \__draw_reset_bb:
1115   }
1116 \cs_new_protected:Npn \__draw_scope_bb_end:
1117 {
1118   \dim_gset_eq:NN \g__draw_xmax_dim \l__draw_xmax_dim

```

```

1119         \dim_gset_eq:NN \g__draw_xmin_dim \l__draw_xmin_dim
1120         \dim_gset_eq:NN \g__draw_ymax_dim \l__draw_ymax_dim
1121         \dim_gset_eq:NN \g__draw_ymin_dim \l__draw_ymin_dim
1122     \group_end:
1123 }

```

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

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

```

\draw_suspend_end:
1124 \cs_new_protected:Npn \draw_suspend_begin:
1125 {
1126     \__draw_scope_bb_begin:
1127     \draw_path_scope_begin:
1128     \draw_transform_matrix_reset:
1129     \draw_transform_shift_reset:
1130 }
1131 \cs_new_protected:Npn \draw_suspend_end:
1132 {
1133     \draw_path_scope_end:
1134     \__draw_scope_bb_end:
1135 }

```

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

```

1136 </initex | package>

```

5 l3draw-softpath implementation

```

1137 <*initex | package>

```

```

1138 <@@=draw>

```

5.1 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. Paths need to be global (as specials are), so we cannot use `l3tl-build` or a similar approach. Instead, we use the same idea as `pgf`: use a series of buffer macros such that in most cases we don't add tokens to the main list. This will get slow only for *enormous* paths.

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.

```

\g__draw_softpath_buffer_a_tl
\g__draw_softpath_buffer_b_tl
1139 \tl_new:N \g__draw_softpath_main_tl
1140 \tl_new:N \g__draw_softpath_buffer_a_tl
1141 \tl_new:N \g__draw_softpath_buffer_b_tl

```

(End definition for `\g__draw_softpath_main_tl`, `\g__draw_softpath_buffer_a_tl`, and `\g__draw_softpath_buffer_b_tl`.)


```

\g__draw_softpath_buffer_a_int Tracking data.
\g__draw_softpath_buffer_b_int 1142 \int_new:N \g__draw_softpath_buffer_a_int
1143 \int_new:N \g__draw_softpath_buffer_b_int

(End definition for \g__draw_softpath_buffer_a_int and \g__draw_softpath_buffer_b_int.)

\g__draw_softpath_corners_bool Allow for optimised path use.
1144 \bool_new:N \g__draw_softpath_corners_bool

(End definition for \g__draw_softpath_corners_bool.)

__draw_softpath_add:n The softpath itself is quite simple. We use three token lists to hold the data: two buffers
__draw_softpath_add:x of limited length, and the main list of arbitrary size. Most of the time this will mean
__draw_softpath_concat:n that we don't add to the full list, so performance will be acceptable.
\__draw_softpath_reset_buffers: 1145 \cs_new_protected:Npn \__draw_softpath_add:n #1
1146 {
1147   \int_compare:nNnTF \g__draw_softpath_buffer_a_int < { 40 }
1148   {
1149     \int_gincr:N \g__draw_softpath_buffer_a_int
1150     \tl_gput_right:Nn \g__draw_softpath_buffer_a_tl {#1}
1151   }
1152   {
1153     \int_compare:nNnTF \g__draw_softpath_buffer_b_int < { 40 }
1154     {
1155       \int_gincr:N \g__draw_softpath_buffer_b_int
1156       \tl_gset:Nx \g__draw_softpath_buffer_b_tl
1157       {
1158         \exp_not:V \g__draw_softpath_buffer_b_tl
1159         \exp_not:V \g__draw_softpath_buffer_a_tl
1160         \exp_not:n {#1}
1161       }
1162       \int_gzero:N \g__draw_softpath_buffer_a_int
1163       \tl_gclear:N \g__draw_softpath_buffer_a_tl
1164     }
1165     { \__draw_softpath_concat:n {#1} }
1166   }
1167 }
1168 \cs_generate_variant:Nn \__draw_softpath_add:n { x }
1169 \cs_new_protected:Npn \__draw_softpath_concat:n #1
1170 {
1171   \tl_gset:Nx \g__draw_softpath_main_tl
1172   {
1173     \exp_not:V \g__draw_softpath_main_tl
1174     \exp_not:V \g__draw_softpath_buffer_b_tl
1175     \exp_not:V \g__draw_softpath_buffer_a_tl
1176     \exp_not:n {#1}
1177   }
1178   \__draw_softpath_reset_buffers:
1179 }
1180 \cs_new_protected:Npn \__draw_softpath_reset_buffers:
1181 {
1182   \int_gzero:N \g__draw_softpath_buffer_a_int
1183   \tl_gclear:N \g__draw_softpath_buffer_a_tl
1184   \int_gzero:N \g__draw_softpath_buffer_b_int

```

```

1185     \tl_gclear:N \g__draw_softpath_buffer_b_tl
1186 }

```

(End definition for __draw_softpath_add:n, __draw_softpath_concat:n, and __draw_softpath_reset_buffers:.)

```

\__draw_softpath_use: Using and clearing is trivial.
\__draw_softpath_clear: 1187 \cs_new_protected:Npn \__draw_softpath_use:
\__draw_softpath_use_clear: 1188 {
1189     \g__draw_softpath_main_tl
1190     \g__draw_softpath_buffer_b_tl
1191     \g__draw_softpath_buffer_a_tl
1192 }
1193 \cs_new_protected:Npn \__draw_softpath_clear:
1194 {
1195     \tl_gclear:N \g__draw_softpath_main_tl
1196     \tl_gclear:N \g__draw_softpath_buffer_a_tl
1197     \tl_gclear:N \g__draw_softpath_buffer_b_tl
1198     \bool_gset_false:N \g__draw_softpath_corners_bool
1199 }
1200 \cs_new_protected:Npn \__draw_softpath_use_clear:
1201 {
1202     \__draw_softpath_use:
1203     \__draw_softpath_clear:
1204 }

```

(End definition for __draw_softpath_use:, __draw_softpath_clear:, and __draw_softpath_use_clear:.)

```

\g__draw_softpath_lastx_dim For tracking the end of the path (to close it).
\g__draw_softpath_lasty_dim 1205 \dim_new:N \g__draw_softpath_lastx_dim
1206 \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.
1207 \bool_new:N \g__draw_softpath_move_bool
1208 \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 1209 \cs_new_protected:Npn \__draw_softpath_closepath:
\__draw_softpath_moveto:nn 1210 {
1211     \__draw_softpath_add:x
1212     {
1213         \__draw_softpath_close_op:nn
1214         { \dim_use:N \g__draw_softpath_lastx_dim }
1215         { \dim_use:N \g__draw_softpath_lasty_dim }
1216     }
1217 }
1218 \cs_new_protected:Npn \__draw_softpath_curveto:nnnnnn #1#2#3#4#5#6
1219 {
1220     \__draw_softpath_add:n
1221     {

```

```

1222         \_draw_softpath_curveto_opi:nn {#1} {#2}
1223         \_draw_softpath_curveto_opii:nn {#3} {#4}
1224         \_draw_softpath_curveto_opiii:nn {#5} {#6}
1225     }
1226 }
1227 \cs_new_protected:Npn \_draw_softpath_lineto:nn #1#2
1228 {
1229     \_draw_softpath_add:n
1230     { \_draw_softpath_lineto_op:nn {#1} {#2} }
1231 }
1232 \cs_new_protected:Npn \_draw_softpath_moveto:nn #1#2
1233 {
1234     \_draw_softpath_add:n
1235     { \_draw_softpath_moveto_op:nn {#1} {#2} }
1236     \bool_if:NT \g__draw_softpath_move_bool
1237     {
1238         \dim_gset:Nn \g__draw_softpath_lastx_dim {#1}
1239         \dim_gset:Nn \g__draw_softpath_lasty_dim {#2}
1240     }
1241 }
1242 \cs_new_protected:Npn \_draw_softpath_rectangle:nnnn #1#2#3#4
1243 {
1244     \_draw_softpath_add:n
1245     {
1246         \_draw_softpath_rectangle_opi:nn {#1} {#2}
1247         \_draw_softpath_rectangle_opii:nn {#3} {#4}
1248     }
1249 }
1250 \cs_new_protected:Npn \_draw_softpath_roundpoint:nn #1#2
1251 {
1252     \_draw_softpath_add:n
1253     { \_draw_softpath_roundpoint_op:nn {#1} {#2} }
1254     \bool_gset_true:N \g__draw_softpath_corners_bool
1255 }
1256 \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 1257 \cs_new_protected:Npn \_draw_softpath_close_op:nn #1#2
\_draw_softpath_curveto_opii:nn 1258 { \driver_draw_closepath: }
\_draw_softpath_curveto_opiii:nn 1259 \cs_new_protected:Npn \_draw_softpath_curveto_opi:nn #1#2
\_draw_softpath_lineto_op:nn 1260 { \_draw_softpath_curveto_opi:nnNnnNnn {#1} {#2} }
\_draw_softpath_moveto_op:nn 1261 \cs_new_protected:Npn \_draw_softpath_curveto_opi:nnNnnNnn #1#2#3#4#5#6#7#8
\_draw_softpath_roundpoint_op:nn 1262 { \driver_draw_curveto:nnnnnn {#1} {#2} {#4} {#5} {#7} {#8} }
\_draw_softpath_rectangle_opi:nn 1263 \cs_new_protected:Npn \_draw_softpath_curveto_opii:nn #1#2
\_draw_softpath_rectangle_opii:nn 1264 { \_draw_softpath_curveto_opii:nn }
\_draw_softpath_curveto_opi:nnNnnNnn 1265 \cs_new_protected:Npn \_draw_softpath_curveto_opiii:nn #1#2
\_draw_softpath_rectangle_opi:nnNnn 1266 { \_draw_softpath_curveto_opiii:nn }
1267 \cs_new_protected:Npn \_draw_softpath_lineto_op:nn #1#2
1268 { \driver_draw_lineto:nn {#1} {#2} }
1269 \cs_new_protected:Npn \_draw_softpath_moveto_op:nn #1#2

```

```

1270 { \driver_draw_moveto:nn {#1} {#2} }
1271 \cs_new_protected:Npn \__draw_softpath_roundpoint_op:nn #1#2 { }
1272 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nn #1#2
1273 { \__draw_softpath_rectangle_opi:nnNnn {#1} {#2} }
1274 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nnNnn #1#2#3#4#5
1275 { \driver_draw_rectangle:nnnn {#1} {#2} {#4} {#5} }
1276 \cs_new_protected:Npn \__draw_softpath_rectangle_opii:nn #1#2 { }

```

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

5.2 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.

```
1277 \tl_new:N \l__draw_softpath_main_tl
```

(End definition for `\l__draw_softpath_main_tl`.)

`\l__draw_softpath_part_tl` Data structures.

```
1278 \tl_new:N \l__draw_softpath_part_tl
1279 \tl_new:N \l__draw_softpath_curve_end_tl
```

(End definition for `\l__draw_softpath_part_tl`.)

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

```

\l__draw_softpath_lasty_fp 1280 \fp_new:N \l__draw_softpath_lastx_fp
\l__draw_softpath_corneri_dim 1281 \fp_new:N \l__draw_softpath_lasty_fp
\l__draw_softpath_cornerii_dim 1282 \dim_new:N \l__draw_softpath_corneri_dim
\l__draw_softpath_corneriii_dim 1283 \dim_new:N \l__draw_softpath_cornerii_dim
\l__draw_softpath_first_tl 1284 \tl_new:N \l__draw_softpath_first_tl
\l__draw_softpath_move_tl 1285 \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.

```
1286 \fp_const:Nn \c__draw_softpath_arc_fp { 4/3 * (sqrt(2) - 1) }
```

(End definition for `\c__draw_softpath_arc_fp`.)

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

```

\__draw_softpath_round_loop:Nnn 1287 \cs_new_protected:Npn \__draw_softpath_round_corners:
\__draw_softpath_round_action:nn 1288 {
\__draw_softpath_round_action:Nnn 1289 { \bool_if:NT \g__draw_softpath_corners_bool
\__draw_softpath_round_action_curveto:NnnNnn 1290 {
\__draw_softpath_round_action_close: 1291 \group_begin:
\__draw_softpath_round_lookahead:NnnNnn 1292 \tl_clear:N \l__draw_softpath_main_tl
\__draw_softpath_round_roundpoint:NnnNnnNnn 1293 \tl_clear:N \l__draw_softpath_part_tl
\__draw_softpath_round_calc:nnnNnn 1294 \fp_zero:N \l__draw_softpath_lastx_fp
\__draw_softpath_round_calc:nnnnnn 1295 \fp_zero:N \l__draw_softpath_lasty_fp
\__draw_softpath_round_calc:fVnnnn 1296 \tl_clear:N \l__draw_softpath_first_tl
\__draw_softpath_round_calc:nnnnw
\__draw_softpath_round_close:nn
\__draw_softpath_round_close:w
\__draw_softpath_round_end:

```

```

1297         \tl_clear:N \l__draw_softpath_move_tl
1298         \__draw_softpath_concat:n { }
1299         \exp_after:wN \__draw_softpath_round_loop:Nnn
1300         \g__draw_softpath_main_tl
1301         \q_recursion_tail ? ?
1302         \q_recursion_stop
1303     \group_end:
1304 }
1305 \bool_gset_false:N \g__draw_softpath_corners_bool
1306 }

```

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.

```

1307 \cs_new_protected:Npn \__draw_softpath_round_loop:Nnn #1#2#3
1308 {
1309     \quark_if_recursion_tail_stop_do:Nn #1 { \__draw_softpath_round_end: }
1310     \token_if_eq_meaning:NNTF #1 \__draw_softpath_roundpoint_op:nn
1311     { \__draw_softpath_round_action:nn {#2} {#3} }
1312     {
1313         \tl_if_empty:NT \l__draw_softpath_first_tl
1314         { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1315         \fp_set:Nn \l__draw_softpath_lastx_fp {#2}
1316         \fp_set:Nn \l__draw_softpath_lasty_fp {#3}
1317         \token_if_eq_meaning:NNTF #1 \__draw_softpath_moveto_op:nn
1318         {
1319             \tl_put_right:No \l__draw_softpath_main_tl
1320             \l__draw_softpath_move_tl
1321             \tl_put_right:No \l__draw_softpath_main_tl
1322             \l__draw_softpath_part_tl
1323             \tl_set:Nn \l__draw_softpath_move_tl { #1 {#2} {#3} }
1324             \tl_clear:N \l__draw_softpath_first_tl
1325             \tl_clear:N \l__draw_softpath_part_tl
1326         }
1327         { \tl_put_right:Nn \l__draw_softpath_part_tl { #1 {#2} {#3} } }
1328         \__draw_softpath_round_loop:Nnn
1329     }
1330 }
1331 \cs_new_protected:Npn \__draw_softpath_round_action:nn #1#2
1332 {
1333     \dim_set:Nn \l__draw_softpath_corneri_dim {#1}
1334     \dim_set:Nn \l__draw_softpath_cornerii_dim {#2}
1335     \bool_lazy_and:nnTF
1336     { \dim_compare_p:nNn \l__draw_softpath_corneri_dim = { Opt } }
1337     { \dim_compare_p:nNn \l__draw_softpath_cornerii_dim = { Opt } }
1338     { \__draw_softpath_round_loop:Nnn }
1339     { \__draw_softpath_round_action:Nnn }
1340 }

```

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.

```

1341 \cs_new_protected:Npn \__draw_softpath_round_action:Nnn #1#2#3
1342 {
1343   \tl_if_empty:NT \l__draw_softpath_first_tl
1344   { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1345   \token_if_eq_meaning:NNTF #1 \__draw_softpath_curveto_opi:nn
1346   { \__draw_softpath_round_action_curveto:NnnNnn }
1347   {
1348     \token_if_eq_meaning:NNTF #1 \__draw_softpath_close_op:nn
1349     { \__draw_softpath_round_action_close: }
1350     {
1351       \token_if_eq_meaning:NNTF #1 \__draw_softpath_lineto_op:nn
1352       { \__draw_softpath_round_lookahead:NnnNnn }
1353       { \__draw_softpath_round_loop:Nnn }
1354     }
1355   }
1356   #1 {#2} {#3}
1357 }

```

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.

```

1358 \cs_new_protected:Npn \__draw_softpath_round_action_curveto:NnnNnn
1359   #1#2#3#4#5#6
1360 {
1361   \tl_put_right:Nn \l__draw_softpath_part_tl
1362   { #1 {#2} {#3} #4 {#5} {#6} }
1363   \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1364   \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1365   \__draw_softpath_round_lookahead:NnnNnn
1366 }
1367 \cs_new_protected:Npn \__draw_softpath_round_action_close:
1368 {
1369   \bool_lazy_and:nnTF
1370   { ! \tl_if_empty_p:N \l__draw_softpath_first_tl }
1371   { ! \tl_if_empty_p:N \l__draw_softpath_move_tl }
1372   {
1373     \exp_after:wN \__draw_softpath_round_close:nn
1374     \l__draw_softpath_first_tl
1375   }
1376   { \__draw_softpath_round_loop:Nnn }
1377 }

```

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.

```

1378 \cs_new_protected:Npn \__draw_softpath_round_lookahead:NnnNnn #1#2#3#4#5#6
1379 {
1380   \bool_lazy_any:nTF
1381   {
1382     { \token_if_eq_meaning_p:NN #4 \__draw_softpath_lineto_op:nn }
1383     { \token_if_eq_meaning_p:NN #4 \__draw_softpath_curveto_opi:nn }
1384     { \token_if_eq_meaning_p:NN #4 \__draw_softpath_close_op:nn }
1385   }
1386   {
1387     \__draw_softpath_round_calc:nnnNnn
1388     \__draw_softpath_round_loop:Nnn {#5} {#6}

```

```

1389     }
1390     {
1391         \token_if_eq_meaning:NNTF #4 \__draw_softpath_roundpoint_op:nn
1392         { \__draw_softpath_round_roundpoint:NnnNnnNnn }
1393         { \__draw_softpath_round_loop:Nnn }
1394     }
1395     #1 {#2} {#3}
1396     #4 {#5} {#6}
1397 }
1398 \cs_new_protected:Npn \__draw_softpath_round_roundpoint:NnnNnnNnn
1399 #1#2#3#4#5#6#7#8#9
1400 {
1401     \__draw_softpath_round_calc:nnnNnn
1402     \__draw_softpath_round_loop:Nnn
1403     {#8} {#9} #1 {#2} {#3}
1404     #4 {#5} {#6} #7 {#8} {#9}
1405 }

```

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.

```

1406 \cs_new_protected:Npn \__draw_softpath_round_calc:nnnNnn #1#2#3#4#5#6
1407 {
1408     \tl_set:Nx \l__draw_softpath_curve_end_tl
1409     {
1410         \draw_point_interpolate_distance:nnn
1411         \l__draw_softpath_cornerii_dim
1412         { #5 , #6 } { #2 , #3 }
1413     }
1414     \tl_put_right:Nx \l__draw_softpath_part_tl
1415     {
1416         \exp_not:N #4
1417         \__draw_softpath_round_calc:fVnnnn
1418         {
1419             \draw_point_interpolate_distance:nnn
1420             \l__draw_softpath_corneri_dim
1421             { #5 , #6 }
1422             {
1423                 \l__draw_softpath_lastx_fp ,
1424                 \l__draw_softpath_lasty_fp
1425             }
1426         }
1427         \l__draw_softpath_curve_end_tl
1428         {#5} {#6} {#2} {#3}
1429     }
1430     \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1431     \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1432     #1
1433 }

```

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).

```

1434 \cs_new:Npn \__draw_softpath_round_calc:nnnnnn #1#2#3#4#5#6
1435 {
1436   \__draw_softpath_round_calc:nnnnw {#3} {#4} {#5} {#6}
1437   #1 \q_mark #2 \q_stop
1438 }
1439 \cs_generate_variant:Nn \__draw_softpath_round_calc:nnnnnn { fV }

```

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

```

1440 \cs_new:Npn \__draw_softpath_round_calc:nnnnw
1441   #1#2#3#4 #5 , #6 \q_mark #7 , #8 \q_stop
1442 {
1443   {#5} {#6}
1444   \exp_not:N \__draw_softpath_curveto_opi:nn
1445   {
1446     \fp_to_dim:n
1447     { #5 + \c__draw_softpath_arc_fp * ( #1 - #5 ) }
1448   }
1449   {
1450     \fp_to_dim:n
1451     { #6 + \c__draw_softpath_arc_fp * ( #2 - #6 ) }
1452   }
1453   \exp_not:N \__draw_softpath_curveto_opii:nn
1454   {
1455     \fp_to_dim:n
1456     { #7 + \c__draw_softpath_arc_fp * ( #1 - #7 ) }
1457   }
1458   {
1459     \fp_to_dim:n
1460     { #8 + \c__draw_softpath_arc_fp * ( #2 - #8 ) }
1461   }
1462   \exp_not:N \__draw_softpath_curveto_opiii:nn
1463   {#7} {#8}
1464 }

```

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.

```

1465 \cs_new_protected:Npn \__draw_softpath_round_close:nn #1#2
1466 {
1467   \use:x
1468   {
1469     \__draw_softpath_round_calc:nnnNnn
1470     {
1471       \tl_set:Nx \exp_not:N \l__draw_softpath_move_tl
1472       {
1473         \__draw_softpath_moveto_op:nn
1474         \exp_not:N \exp_after:wN
1475         \exp_not:N \__draw_softpath_round_close:w
1476         \exp_not:N \l__draw_softpath_curve_end_tl
1477         \exp_not:N \q_stop

```



```

1478     }
1479     \use:x
1480     {
1481         \exp_not:N \exp_not:N \exp_not:N \use_i:nnnn
1482         {
1483             \__draw_softpath_round_loop:Nnn
1484             \__draw_softpath_close_op:nn
1485             \exp_not:N \exp_after:wN
1486             \exp_not:N \__draw_softpath_round_close:w
1487             \exp_not:N \l__draw_softpath_curve_end_tl
1488             \exp_not:N \q_stop
1489         }
1490     }
1491 }
1492 {#1} {#2}
1493 \__draw_softpath_lineto_op:nn
1494 \exp_after:wN \use_none:n \l__draw_softpath_move_tl
1495 }
1496 }
1497 \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.

```

1498 \cs_new_protected:Npn \__draw_softpath_round_end:
1499 {
1500     \tl_put_right:No \l__draw_softpath_main_tl
1501     \l__draw_softpath_move_tl
1502     \tl_put_right:No \l__draw_softpath_main_tl
1503     \l__draw_softpath_part_tl
1504     \tl_gset_eq:NN \g__draw_softpath_main_tl \l__draw_softpath_main_tl
1505 }

```

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

```

1506 </initex | package>

```

6 l3draw-state implementation

```

1507 <*initex | package>

```

```

1508 <@@=draw>

```

`\g__draw_linewidth_dim` Linewidth for strokes: global as the scope for this relies on the graphics state. The inner
`\g__draw_inner_linewidth_dim` line width is used for places where two lines are used.

```

1509 \dim_new:N \g__draw_linewidth_dim
1510 \dim_new:N \g__draw_inner_linewidth_dim

```

(End definition for `\g__draw_linewidth_dim` and `\g__draw_inner_linewidth_dim`.)

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

```

1511 \dim_new:N \l_draw_default_linewidth_dim
1512 \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. The inner
`\draw_inner_linewidth:n` version is handled at the macro layer but is given a consistent interface here.

```

1513 \cs_new_protected:Npn \draw_linewidth:n #1
1514 {
1515     \dim_gset:Nn \g__draw_linewidth_dim { \fp_to_dim:n {#1} }
1516     \driver_draw_linewidth:n \g__draw_linewidth_dim
1517 }
1518 \cs_new_protected:Npn \draw_inner_linewidth:n #1
1519 { \dim_gset:Nn \g__draw_inner_linewidth_dim { \fp_to_dim:n {#1} } }

```

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

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

```

1520 \cs_new_protected:Npn \draw_miterlimit:n #1
1521 { \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: 1522 \cs_new_protected:Npn \draw_cap_but: { \driver_draw_cap_but: }
\draw_cap_round:      1523 \cs_new_protected:Npn \draw_cap_rectangle: { \driver_draw_cap_rectangle: }
\draw_evenodd_rule:   1524 \cs_new_protected:Npn \draw_cap_round: { \driver_draw_cap_round: }
\draw_nonzero_rule:   1525 \cs_new_protected:Npn \draw_evenodd_rule: { \driver_draw_evenodd_rule: }
\draw_join_bevel:     1526 \cs_new_protected:Npn \draw_nonzero_rule: { \driver_draw_nonzero_rule: }
\draw_join_miter:     1527 \cs_new_protected:Npn \draw_join_bevel: { \driver_draw_join_bevel: }
\draw_join_round:     1528 \cs_new_protected:Npn \draw_join_miter: { \driver_draw_join_miter: }
                     1529 \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.

```

1530 \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    1531 \cs_new_eq:NN \draw_color:n \color_select:n
\draw_color_stroke:n  1532 \cs_new_protected:Npn \draw_color_fill:n #1
    \__draw_color:nn  1533 { \__draw_color:nn { fill } {#1} }
\__draw_color_aux:nn  1534 \cs_new_protected:Npn \draw_color_stroke:n #1
    \__draw_color_aux:Vn 1535 { \__draw_color:nn { stroke } {#1} }
    \__draw_color:nw    1536 \cs_new_protected:Npn \__draw_color:nn #1#2
    {
1537     \color_parse:nN {#2} \l__draw_color_tmp_tl
1538     \__draw_color_aux:Vn \l__draw_color_tmp_tl {#1}
1539 }
1540 }
1541 \cs_new_protected:Npn \__draw_color_aux:nn #1#2
1542 { \__draw_color:nw {#2} #1 \q_stop }
1543 \cs_generate_variant:Nn \__draw_color_aux:nn { V }
1544 \cs_new_protected:Npn \__draw_color:nw #1#2 ~ #3 \q_stop
1545 { \use:c { \__draw_color_ #2 :nw } {#1} #3 \q_stop }
1546 \cs_new_protected:Npn \__draw_color_cmyk:nw #1#2 ~ #3 ~ #4 ~ #5 \q_stop
1547 { \use:c { driver_draw_color_ #1 _cmyk:nnnn } {#2} {#3} {#4} {#5} }
1548 \cs_new_protected:Npn \__draw_color_gray:nw #1#2 \q_stop

```

```

1549 { \use:c { driver_draw_color_ #1 _gray:n } {#2} }
1550 \cs_new_protected:Npn \__draw_color_rgb:nw #1#2 ~ #3 ~ #4 \q_stop
1551 { \use:c { driver_draw_color_ #1 _rgb:nnn } {#2} {#3} {#4} }
1552 \cs_new_protected:Npn \__draw_color_spot:nw #1#2 ~ #3 \q_stop
1553 { \use:c { driver_draw_color_ #1 _spot:nn } {#2} {#3} }

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

1554 \</initex | package>

```

7 l3draw-transforms implementation

```

1555 \*initex | package>
1556 \@@=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.
1557 \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 1558 \fp_new:N \l__draw_matrix_a_fp
\l__draw_matrix_c_fp 1559 \fp_new:N \l__draw_matrix_b_fp
\l__draw_xshift_dim 1560 \fp_new:N \l__draw_matrix_c_fp
\l__draw_yshift_dim 1561 \fp_new:N \l__draw_matrix_d_fp
1562 \dim_new:N \l__draw_xshift_dim
1563 \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: 1564 \cs_new_protected:Npn \draw_transform_matrix_reset:
1565 {
1566   \fp_set:Nn \l__draw_matrix_a_fp { 1 }
1567   \fp_zero:N \l__draw_matrix_b_fp
1568   \fp_zero:N \l__draw_matrix_c_fp
1569   \fp_set:Nn \l__draw_matrix_d_fp { 1 }
1570 }
1571 \cs_new_protected:Npn \draw_transform_shift_reset:
1572 {
1573   \dim_zero:N \l__draw_xshift_dim
1574   \dim_zero:N \l__draw_yshift_dim
1575 }
1576 \draw_transform_matrix_reset:
1577 \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:nnnn` Setting the transform matrix is straight-forward, with just a bit of expansion to sort out.
`\draw_transform_shift:n` With the mechanism active, the identity matrix is set.

```

1578 \cs_new_protected:Npn \draw_transform_matrix:nnnn #1#2#3#4
1579 {
1580   \fp_set:Nn \l__draw_matrix_a_fp {#1}
1581   \fp_set:Nn \l__draw_matrix_b_fp {#2}
1582   \fp_set:Nn \l__draw_matrix_c_fp {#3}
1583   \fp_set:Nn \l__draw_matrix_d_fp {#4}
1584   \bool_lazy_all:nTF
1585   {
1586     { \fp_compare_p:nNn \l__draw_matrix_a_fp = \c_one_fp }
1587     { \fp_compare_p:nNn \l__draw_matrix_b_fp = \c_zero_fp }
1588     { \fp_compare_p:nNn \l__draw_matrix_c_fp = \c_zero_fp }
1589     { \fp_compare_p:nNn \l__draw_matrix_d_fp = \c_one_fp }
1590   }
1591   { \bool_set_false:N \l__draw_matrix_active_bool }
1592   { \bool_set_true:N \l__draw_matrix_active_bool }
1593 }
1594 \cs_new_protected:Npn \draw_transform_shift:n #1
1595 {
1596   \__draw_point_process:nn
1597   { \__draw_transform_shift:nn } {#1}
1598 }
1599 \cs_new_protected:Npn \__draw_transform_shift:nn #1#2
1600 {
1601   \dim_set:Nn \l__draw_xshift_dim {#1}
1602   \dim_set:Nn \l__draw_yshift_dim {#2}
1603 }

```

(End definition for `\draw_transform_matrix:nnnn`, `\draw_transform_shift:n`, and `__draw_transform_shift:nn`. These functions are documented on page ??.)

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

```

1604 \cs_new_protected:Npn \draw_transform_matrix_concat:nnnn #1#2#3#4
1605 {
1606   \use:x
1607   {
1608     \__draw_transform_concat:nnnn
1609     { \fp_eval:n {#1} }
1610     { \fp_eval:n {#2} }
1611     { \fp_eval:n {#3} }
1612     { \fp_eval:n {#4} }
1613   }
1614 }
1615 \cs_new_protected:Npn \__draw_transform_concat:nnnn #1#2#3#4
1616 {
1617   \use:x
1618   {
1619     \draw_transform_matrix:nnnn
1620     { #1 * \l__draw_matrix_a_fp + #2 * \l__draw_matrix_c_fp }

```

```

1621         { #1 * \l__draw_matrix_b_fp + #2 * \l__draw_matrix_d_fp }
1622         { #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_c_fp }
1623         { #3 * \l__draw_matrix_b_fp + #4 * \l__draw_matrix_d_fp }
1624     }
1625 }
1626 \cs_new_protected:Npn \draw_transform_shift_concat:n #1
1627 {
1628     \__draw_point_process:nn
1629     { \__draw_transform_shift_concat:nn } {#1}
1630 }
1631 \cs_new_protected:Npn \__draw_transform_shift_concat:nn #1#2
1632 {
1633     \dim_set:Nn \l__draw_xshift_dim { \l__draw_xshift_dim + #1 }
1634     \dim_set:Nn \l__draw_yshift_dim { \l__draw_yshift_dim + #2 }
1635 }

```

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

<pre> \draw_transform_matrix_invert: __draw_transform_invert:n __draw_transform_invert:f \draw_transform_shift_invert: </pre>	<p>Standard mathematics: calculate the inverse matrix and use that, then undo the shifts.</p> <pre> 1636 \cs_new_protected:Npn \draw_transform_matrix_invert: 1637 { 1638 \bool_if:NT \l__draw_matrix_active_bool 1639 { 1640 __draw_transform_invert:f 1641 { 1642 \fp_eval:n 1643 { 1644 1 / 1645 (1646 \l__draw_matrix_a_fp * \l__draw_matrix_d_fp 1647 - \l__draw_matrix_b_fp * \l__draw_matrix_c_fp 1648) 1649 } 1650 } 1651 } 1652 } 1653 \cs_new_protected:Npn __draw_transform_invert:n #1 1654 { 1655 \fp_set:Nn \l__draw_matrix_a_fp 1656 { \l__draw_matrix_d_fp * #1 } 1657 \fp_set:Nn \l__draw_matrix_b_fp 1658 { -\l__draw_matrix_b_fp * #1 } 1659 \fp_set:Nn \l__draw_matrix_c_fp 1660 { -\l__draw_matrix_c_fp * #1 } 1661 \fp_set:Nn \l__draw_matrix_d_fp 1662 { \l__draw_matrix_a_fp * #1 } 1663 } 1664 \cs_generate_variant:Nn __draw_transform_invert:n { f } 1665 \cs_new_protected:Npn \draw_transform_shift_invert: 1666 { 1667 \dim_set:Nn \l__draw_xshift_dim { -\l__draw_xshift_dim } 1668 \dim_set:Nn \l__draw_yshift_dim { -\l__draw_yshift_dim } 1669 } </pre>
---	--

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

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

```

1670 \cs_new_protected:Npn \draw_transform_triangle:nnn #1#2#3
1671 {
1672   \__draw_point_process:nnn
1673   {
1674     \__draw_point_process:nn
1675     { \__draw_tranform_triangle:nnnnnn }
1676     {#1}
1677   }
1678   {#2} {#3}
1679 }
1680 \cs_new_protected:Npn \__draw_tranform_triangle:nnnnnn #1#2#3#4#5#6
1681 {
1682   \use:x
1683   {
1684     \draw_transform_matrix:nnnn
1685     { #3 - #1 }
1686     { #4 - #2 }
1687     { #5 - #1 }
1688     { #6 - #2 }
1689     \draw_transform_shift:n { #1 , #2 }
1690   }
1691 }

```

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

`\draw_transform_scale:n` Lots of shortcuts.

```

\draw_transform_xscale:n 1692 \cs_new_protected:Npn \draw_transform_scale:n #1
\draw_transform_yscale:n 1693 { \draw_transform_matrix_concat:nnnn { #1 } { 0 } { 0 } { #1 } }
\draw_transform_xshift:n 1694 \cs_new_protected:Npn \draw_transform_xscale:n #1
\draw_transform_yshift:n 1695 { \draw_transform_matrix_concat:nnnn { #1 } { 0 } { 0 } { 1 } }
\draw_transform_xslant:n 1696 \cs_new_protected:Npn \draw_transform_yscale:n #1
\draw_transform_yslant:n 1697 { \draw_transform_matrix_concat:nnnn { 1 } { 0 } { 0 } { #1 } }
1698 \cs_new_protected:Npn \draw_transform_xshift:n #1
1699 { \draw_transform_shift_concat:n { #1 , 0 } }
1700 \cs_new_protected:Npn \draw_transform_yshift:n #1
1701 { \draw_transform_shift_concat:n { 0 , #1 } }
1702 \cs_new_protected:Npn \draw_transform_xslant:n #1
1703 { \draw_transform_matrix_concat:nnnn { 1 } { 0 } { #1 } { 1 } }
1704 \cs_new_protected:Npn \draw_transform_yslant:n #1
1705 { \draw_transform_matrix_concat: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 1706 \cs_new_protected:Npn \draw_transform_rotate:n #1
\__draw_transform_rotate:f 1707 { \__draw_transform_rotate:f { \fp_eval:n {#1} } }
\__draw_transform_rotate:nn 1708 \cs_new_protected:Npn \__draw_transform_rotate:n #1
\__draw_transform_rotate:ff 1709 {
1710   \__draw_transform_rotate:ff
1711   { \fp_eval:n { cosd(#1) } }

```

```

1712     { \fp_eval:n { sind(#1) } }
1713   }
1714   \cs_generate_variant:Nn \_draw_transform_rotate:n { f }
1715   \cs_new_protected:Npn \_draw_transform_rotate:nn #1#2
1716     { \draw_transform_matrix_concat:nnnn {#1} {#2} { -#2 } { #1 } }
1717   \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 ??.)

1718 \</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	
<code>\begin</code>	856, 860, 884, 887
bool commands:	
<code>\bool_gset_eq:NN</code>	609
<code>\bool_gset_false:N</code>	1198, 1305
<code>\bool_gset_true:N</code>	1208, 1254
<code>\bool_if:NTF</code>	30, 69, 530, 533, 542, 543, 979, 1011, 1236, 1289, 1638
<code>\bool_lazy_all:nTF</code>	1584
<code>\bool_lazy_and:nnTF</code> 61, 526, 1335, 1369	
<code>\bool_lazy_any:nTF</code>	1380
<code>\bool_lazy_or:nnTF</code>	407, 502, 535
<code>\bool_new:N</code>	56, 490, 491, 492, 493, 494, 587, 1035, 1144, 1207, 1557
<code>\bool_set_eq:NN</code>	602
<code>\bool_set_false:N</code>	64, 513, 514, 515, 1591
<code>\bool_set_true:N</code>	65, 519, 550, 554, 555, 1053, 1592
box commands:	
<code>\box_move_down:nn</code>	1074
<code>\box_new:N</code>	1036
<code>\box_set_dp:Nn</code>	1079
<code>\box_set_ht:Nn</code>	1077
<code>\box_set_wd:Nn</code>	1080
<code>\box_use_drop:N</code>	1075, 1083
C	
clist commands:	
<code>\clist_map_inline:nn</code>	516
color commands:	
<code>\color_parse:nN</code>	1538
<code>\color_select:n</code>	1531
cs commands:	
<code>\cs_generate_variant:Nn</code>	263, 456, 489, 635, 647, 654, 664, 705, 724, 731, 739, 746, 755, 761, 773, 776, 794, 816, 824, 830, 855, 869, 883, 904, 939, 958, 971, 1168, 1256, 1439, 1543, 1664, 1714, 1717
<code>\cs_if_exist:NTF</code>	518
<code>\cs_if_exist_use:NTF</code>	246, 255, 521
<code>\cs_new:Npn</code>	360, 370, 380, 390, 627, 633, 636, 638, 645, 648, 650, 652, 655, 656, 658, 660, 662, 665, 667, 669, 671, 673, 678, 687, 697, 706, 712, 717, 725, 732, 740, 747, 756, 762, 768, 774, 777, 783, 792, 795, 805, 810, 817, 825, 831, 841, 847, 861, 870, 876, 888, 890, 899, 929, 931, 940, 945, 959, 961, 963, 972, 977, 1004, 1009, 1434, 1440, 1497
<code>\cs_new_eq:NN</code>	1531
<code>\cs_new_protected:Npn</code>	20, 42, 49, 57, 67, 76, 82, 88, 94, 101, 115, 123, 128, 130, 132, 145, 152, 188, 190, 201, 207, 237, 264, 296, 302, 308, 313, 321, 330, 335, 346, 401, 403, 416, 423, 432, 438, 440, 450, 457, 463, 470, 495, 500, 511, 548, 552, 557, 564, 588, 607, 911, 913, 915, 917, 921, 1038, 1045, 1060, 1089, 1096, 1107, 1116, 1124, 1131, 1145, 1169, 1180, 1187, 1193, 1200, 1209, 1218, 1227, 1232, 1242, 1250, 1257, 1259, 1261, 1263, 1265, 1267, 1269, 1271, 1272, 1274, 1276, 1287, 1307, 1331, 1341, 1358, 1367, 1378, 1398, 1406, 1465, 1498, 1513, 1518, 1520, 1522, 1523, 1524, 1525, 1526, 1527, 1528, 1529, 1532, 1534, 1536, 1541, 1544, 1546, 1548, 1550, 1552, 1564, 1571, 1578, 1594, 1599, 1604, 1615, 1626, 1631, 1636, 1653, 1665, 1670, 1680, 1692, 1694, 1696, 1698, 1700, 1702, 1704, 1706, 1708, 1715
D	
dim commands:	
<code>\dim_abs:n</code>	445, 446
<code>\dim_compare:nNnTF</code> 452, 459, 566, 1064	
<code>\dim_compare_p:nNn</code>	62, 63, 1336, 1337
<code>\dim_eval:n</code>	445, 446
<code>\dim_gset:Nn</code>	22, 24, 26, 28, 32, 34, 36, 38, 44, 45, 46, 47, 51, 52, 568, 1040, 1041, 1042, 1043, 1238, 1239, 1515, 1519
<code>\dim_gset_eq:NN</code>	614, 615, 616, 617, 618, 619, 620, 621, 1099, 1118, 1119, 1120, 1121
<code>\dim_gzero:N</code>	1066, 1067, 1068, 1069
<code>\dim_max:nn</code>	23, 27, 33, 37
<code>\dim_min:nn</code>	25, 29, 35, 39
<code>\dim_new:N</code>	14, 15, 16, 17, 18, 19, 54, 55, 579, 580, 581, 582, 583, 584, 585,

586, 905, 906, 907, 908, 909, 910, 1031, 1032, 1033, 1034, 1086, 1103, 1104, 1105, 1106, 1205, 1206, 1282, 1283, 1509, 1510, 1511, 1562, 1563	\draw_path_rectangle_corners:nn	432
\dim_set:Nn	\draw_path_scope_begin:	588, 1094, 1127
59, 60, 923, 924, 1333, 1334, 1512, 1601, 1602, 1633, 1634, 1667, 1668	\draw_path_scope_end:	588, 1098, 1133
\dim_set_eq:NN	\draw_path_use:n	495
.... 591, 592, 593, 594, 595, 596, 597, 598, 1093, 1110, 1111, 1112, 1113	\draw_path_use_clear:n	495
\dim_step_inline:nnnn	\draw_point:nn	656
\dim_use:N .. 566, 571, 573, 1214, 1215	\draw_point_add:nn	665
\dim_zero:N	\draw_point_diff:nn	665
\c_max_dim	\draw_point_interpolate_arcaxes:nnnnnn	795
47, 566, 1040, 1041, 1042, 1043, 1064	\draw_point_interpolate_curve:nnnnn	831
draw commands:	\draw_point_interpolate_curve:nnnnnn	831
\l_draw_bb_update_bool	\draw_point_interpolate_curve_	
..... 30, 527, 1035, 1053	auxi:nnnnnnnn	831
\draw_begin:	\draw_point_interpolate_curve_	
..... 1045	auxii:nnnnnnnn	831
\draw_cap_but:	\draw_point_interpolate_curve_	
..... 1522	auxiii:nnnnnn	831
\draw_cap_rectangle:	\draw_point_interpolate_curve_	
..... 1522	auxiv:nnnnnn	831
\draw_cap_round:	\draw_point_interpolate_curve_	
..... 1522	auxv:nnw	831
\draw_color:n	\draw_point_interpolate_curve_	
1058, 1531	auxvi:n	831
\draw_color_fill:n	\draw_point_interpolate_curve_	
..... 1531	auxvii:nnnnnnnn	831
\draw_color_stroke:n	\draw_point_interpolate_curve_	
..... 1531	auxviii:nnnnnn	831
\l_draw_default_linewidth_dim ..	\draw_point_interpolate_distance:nnn	777, 1410, 1419
..... 1057, 1511	\draw_point_interpolate_line:nnn	762
\draw_end:	\draw_point_intersect_circles:nnnnn	706
..... 1045 706	
\draw_evenodd_rule:	\draw_point_intersect_lines:nnnn	678
..... 1522	\draw_point_polar:nn	658
\draw_inner_linewidth:n	\draw_point_polar:nnn	271, 279, 284, 290, 658
..... 1513	\draw_point_scale:nn	665
\draw_join_bevel:	\draw_point_transform:n	80, 92, 110, 112, 113, 149, 150, 278, 283, 341, 413, 972
..... 1522	\draw_point_unit_vector:n .. 671, 790	
\draw_join_miter:	\draw_point_vec:nn	929
..... 1522	\draw_point_vec:nnn	929
\draw_join_round:	\draw_point_vec_polar:nn	959
..... 1522	\draw_point_vec_polar:nnn	959
\draw_linewidth:n	\draw_scope_begin:	1089
1057, 1513	\draw_scope_end:	1096
\draw_miterlimit:n	\draw_suspend_begin:	1124
..... 1520	\draw_suspend_end:	1124
\draw_nonzero_rule:		
..... 1522		
\draw_path_arc:nnn		
188, 333		
\draw_path_arc:nnnn		
188		
\draw_path_arc_axes:nnnn		
330		
\draw_path_canvas_curveto:nnn ..		
128		
\draw_path_canvas_lineto:n		
128		
\draw_path_canvas_moveto:n		
128		
\draw_path_circle:nn		
401		
\draw_path_close:		
123, 429		
\draw_path_corner_arc:nn		
57		
\draw_path_curveto:nn		
145		
\draw_path_curveto:nnn		
76		
\draw_path_ellipse:nnn		
335, 402		
\draw_path_grid:nnnn		
440		
\draw_path_lineto:n		
..... 76, 426, 427, 428, 478, 486		
\draw_path_moveto:n		
..... 76, 425, 430, 477, 485		
\draw_path_rectangle:nn		
403, 439		

\draw_transform_matrix:nnnn	1578, 1619, 1684
\draw_transform_matrix_concat:nnnn	1604, 1693, 1695, 1697, 1703, 1705, 1716
\draw_transform_matrix_invert:		1636
\draw_transform_matrix_reset:	...	1054, 1128, 1564
\draw_transform_rotate:n	1706
\draw_transform_scale:n	1692
\draw_transform_shift:n	..	1578, 1689
\draw_transform_shift_concat:n	..	1604, 1699, 1701
\draw_transform_shift_invert:	..	1636
\draw_transform_shift_reset:	...	1055, 1129, 1564
\draw_transform_triangle:nnn	...	332, 1670
\draw_transform_xscale:n	1692
\draw_transform_xshift:n	1692
\draw_transform_xslant:n	1692
\draw_transform_yscale:n	1692
\draw_transform_yshift:n	1692
\draw_transform_yslant:n	1692
\draw_xvec:n	911, 926
\draw_yvec:n	911, 927
\draw_zvec:n	911, 928
draw internal commands:		
_draw_color:nn	1531
_draw_color:nw	1531
_draw_color_aux:nn	1531
_draw_color_cmyk:nw	1546
_draw_color_gray:nw	1548
_draw_color_rgb:nw	1550
_draw_color_spot:nw	1552
\l_draw_color_tmp_tl	1530, 1538, 1539	
\l_draw_corner_arc_bool	56, 64, 65, 69, 408
\l_draw_corner_xarc_dim	54, 59, 62, 72	
\l_draw_corner_yarc_dim	54, 60, 63, 73	
_draw_draw_polar:nnn	658
_draw_draw_vec_polar:nnn	962, 963, 971
\l_draw_fill_color_tl	1086
\g_draw_id_int	1037, 1048
\g_draw_inner_linewidth_dim	...	1509, 1519
\g_draw_linewidth_dim	574, 1093, 1099, 1509, 1515, 1516
\l_draw_linewidth_dim	1086, 1093, 1099
\l_draw_main_box	1036, 1049, 1071, 1075, 1077, 1079, 1080, 1083
\l__draw_matrix_a_fp	984, 1016, 1558, 1566, 1580, 1586, 1620, 1622, 1646, 1655, 1662
\l__draw_matrix_active_bool	409, 979, 1011, 1557, 1591, 1592, 1638
\l__draw_matrix_b_fp	990, 1021, 1558, 1567, 1581, 1587, 1621, 1623, 1647, 1657, 1658
\l__draw_matrix_c_fp	985, 1017, 1558, 1568, 1582, 1588, 1620, 1622, 1647, 1659, 1660
\l__draw_matrix_d_fp	991, 1022, 1561, 1569, 1583, 1589, 1621, 1623, 1646, 1656, 1661
_draw_path_arc:nnnn	188
_draw_path_arc:nnNnn	188
\c__draw_path_arc_60_fp	188
\c__draw_path_arc_90_fp	188
_draw_path_arc_add:nnnn	188
_draw_path_arc_aux_add:nn	298, 304, 316, 321
_draw_path_arc_auxi:nnnnNnn	...	188, 215, 222
_draw_path_arc_auxii:nnnNnnnn	188
_draw_path_arc_auxiii:nn	188
_draw_path_arc_auxiv:nnnn	...	188
_draw_path_arc_auxv:nn	188
_draw_path_arc_auxvi:nn	188
\l_draw_path_arc_delta_fp	188
\l__draw_path_arc_start_fp	188
_draw_path_curveto:nnnn	145
_draw_path_curveto:nnnnnn	76, 139, 159, 292, 362, 372, 382, 392
\c__draw_path_curveto_a_fp	145
\c__draw_path_curveto_b_fp	145
_draw_path_ellipse:nnnnnn	...	335
_draw_path_ellipse_arci:nnnnnn	335
_draw_path_ellipse_arcii:nnnnnn	335
_draw_path_ellipse_arciiii:nnnnnn	335
_draw_path_ellipse_arciiv:nnnnnn	335
\c__draw_path_ellipse_fp	335
_draw_path_grid_auxi:nnnnnn	..	440
_draw_path_grid_auxii:nnnnnn	..	440
_draw_path_grid_auxiii:nnnnnn	440
_draw_path_grid_auxiiii:nnnnnn	440
_draw_path_grid_auxiv:nnnnnnnn	440
\g__draw_path_lastx_dim	14, 51, 163, 299, 305, 591, 620
\l__draw_path_lastx_dim	579, 591, 620	
\g__draw_path_lasty_dim	14, 52, 170, 300, 306, 592, 621

\l__draw_path_lasty_dim	579 , 592 , 621	__draw_point_interpolate_-	
__draw_path_lineto:nn	76 , 131	arcaxes_auxiii:nnnnnnnn	795
__draw_path_mark_corner:		__draw_point_interpolate_-	
.....	67 , 96 , 107 , 125 , 138 , 158 , 229	arcaxes_auxiv:nnnnnnnn	795
__draw_path_moveto:nn		__draw_point_interpolate_curve_-	
.....	76 , 129 , 350 , 358	auxi:nnnnnnnnnn	836 , 841
__draw_path_rectangle:nnnn	403	__draw_point_interpolate_curve_-	
__draw_path_rectangle_corners:nnnn		auxii:nnnnnnnnnn	843 , 847 , 855
.....	432	__draw_point_interpolate_curve_-	
__draw_path_rectangle_corners:nnnnn		auxiii:nnnnnnnn	850 , 861 , 869
.....	435 , 438	__draw_point_interpolate_curve_-	
__draw_path_rectangle_rounded:nnnn		auxiv:nnnnnn	863 , 864 , 865 , 870
.....	403	__draw_point_interpolate_curve_-	
__draw_path_reset_limits:		auxv:nnw	872 , 876 , 883
.....	20 , 507 , 599 , 1052	__draw_point_interpolate_curve_-	
\l__draw_path_tmp_tl		auxvi:n	867 , 888
.....	11 , 266 , 292 , 311 , 315 , 319 , 323	__draw_point_interpolate_curve_-	
\l__draw_path_tmpa_fp	11 , 154 , 164 , 176	auxvii:nnnnnnnn	889 , 890
\l__draw_path_tmpb_fp	11 , 155 , 171 , 180	__draw_point_interpolate_curve_-	
__draw_path_update_last:nn		auxviii:nnnnnn	892 , 899 , 904
.....	49 , 86 , 99 , 121 , 421	__draw_point_interpolate_-	
__draw_path_update_limits:nn		distance:nnnn	780 , 783
... 20 , 84 , 97 , 117 , 118 , 119 , 418 , 419		__draw_point_interpolate_-	
__draw_path_use:n	495	distance:nnnnnn	777 , 787
__draw_path_use_action_draw:	495	__draw_point_interpolate_-	
__draw_path_use_action_fillstroke:		distance:nnnnnn	777
.....	495	__draw_point_interpolate_line_-	
\l__draw_path_use_bb_bool	493	aux:nnnnnn	762
\l__draw_path_use_clear_bool	493 , 530	__draw_point_interpolate_line_-	
\l__draw_path_use_clip_bool		aux:nnnnnn	762
.....	490 , 513 , 533	__draw_point_intersect_circles_-	
\l__draw_path_use_fill_bool		auxi:nnnnnnnn	706
.....	490 , 514 , 536 , 542 , 554	__draw_point_intersect_circles_-	
__draw_path_use_stroke_bb:	495	auxii:nnnnnnnn	706
__draw_path_use_stroke_bb_-		__draw_point_intersect_circles_-	
aux:NnN	495	auxiii:nnnnnnnn	706
\l__draw_path_use_stroke_bool		__draw_point_intersect_circles_-	
.....	490 , 515 , 528 , 537 , 543 , 550 , 555	auxiv:nnnnnnnn	706
\g__draw_path_xmax_dim		__draw_point_intersect_circles_-	
.....	16 , 22 , 23 , 44 , 593 , 616	auxv:nnnnnnnnnn	706
\l__draw_path_xmax_dim	579 , 593 , 616	__draw_point_intersect_circles_-	
\g__draw_path_xmin_dim		auxvi:nnnnnnnn	706
.....	16 , 24 , 25 , 45 , 594 , 617	__draw_point_intersect_circles_-	
\l__draw_path_xmin_dim	579 , 594 , 617	auxvii:nnnnnnnn	706
\g__draw_path_ymax_dim		__draw_point_intersect_lines:nnnnnnn	
.....	16 , 26 , 27 , 46 , 595 , 618	678
\l__draw_path_ymax_dim	579 , 595 , 618	__draw_point_intersect_lines:nnnnnnnn	
\g__draw_path_ymin_dim		678
.....	16 , 28 , 29 , 47 , 596 , 619	__draw_point_intersect_lines_-	
\l__draw_path_ymin_dim	579 , 596 , 619	aux:nnnnnn	678
__draw_point_interpolate_-		__draw_point_process:nn	78 ,
arcaxes_auxi:nnnnnnnnnn	795	90 , 105 , 129 , 131 , 136 , 267 , 273 ,	
__draw_point_interpolate_-		275 , 286 , 339 , 627 , 672 , 779 , 785 ,	
arcaxes_auxii:nnnnnnnnnn	795	799 , 919 , 974 , 1006 , 1596 , 1628 , 1674	

```

\__draw_point_process:nnn .. 103,
    134, 147, 337, 405, 434, 442, 627,
    680, 682, 708, 764, 797, 833, 835, 1672
\__draw_point_process_auxi:nn .. 627
\__draw_point_process_auxii:nw .. 627
\__draw_point_process_auxiii:nnn 627
\__draw_point_process_auxiv:nw .. 627
\__draw_point_to_dim:n .....
    630, 641, 642, 650, 827,
    901, 933, 947, 965, 981, 997, 1013, 1026
\__draw_point_to_dim_aux:n .... 650
\__draw_point_to_dim_aux:w .... 650
\__draw_point_transform:nn .... 972
\__draw_point_transform_noshift:n
    270, 289, 343, 344, 1004
\__draw_point_transform_noshift:nn
    1004
\__draw_point_unit_vector:nn .. 671
\__draw_point_vec:nn ..... 929
\__draw_point_vec:nnn ..... 929
\__draw_point_vec_polar:nnn ... 959
\__draw_reset_bb: ... 1038, 1051, 1114
\__draw_scope_bb_begin: .. 1107, 1126
\__draw_scope_bb_end: ... 1107, 1134
\__draw_select_cmyk:nw ..... 1531
\__draw_select_gray:nw ..... 1531
\__draw_select_rgb:nw ..... 1531
\__draw_softpath_add:n .... 1145,
    1211, 1220, 1229, 1234, 1244, 1252
\c__draw_softpath_arc_fp .....
    1286, 1447, 1451, 1456, 1460
\g__draw_softpath_buffer_a_int ..
    1142, 1147, 1149, 1162, 1182
\g__draw_softpath_buffer_a_tl ...
    1139, 1150,
    1159, 1163, 1175, 1183, 1191, 1196
\g__draw_softpath_buffer_b_int ..
    1142, 1153, 1155, 1184
\g__draw_softpath_buffer_b_tl ...
    1139,
    1156, 1158, 1174, 1185, 1190, 1197
\__draw_softpath_clear: .....
    506, 605, 612, 1056, 1187
\__draw_softpath_close_op:nn ...
    1213, 1257, 1348, 1384, 1484
\__draw_softpath_closepath: ....
    126, 357, 1209
\__draw_softpath_concat:n .....
    600, 1145, 1298
\l__draw_softpath_corneri_dim ...
    1280, 1333, 1336, 1420
\l__draw_softpath_cornerii_dim ..
    1280, 1334, 1337, 1411
\g__draw_softpath_corners_bool ..
    604, 610, 1144, 1198, 1254, 1289, 1305
\l__draw_softpath_corners_bool ..
    579, 603, 611
\l__draw_softpath_curve_end_tl ..
    1279, 1408, 1427, 1476, 1487
\__draw_softpath_curveto:nnnnnn ..
    120, 1209
\__draw_softpath_curveto_opi:nn ..
    1222, 1257, 1345, 1383, 1444
\__draw_softpath_curveto_-
    opi:nnNnnNnn ..... 1257
\__draw_softpath_curveto_opii:nn
    1223, 1257, 1453
\__draw_softpath_curveto_-
    opiii:nn ..... 1224, 1257, 1462
\l__draw_softpath_first_tl .....
    1280, 1296, 1313,
    1314, 1324, 1343, 1344, 1370, 1374
\g__draw_softpath_lastx_dim ....
    597, 614, 1205, 1214, 1238
\l__draw_softpath_lastx_dim ....
    585, 597, 614
\l__draw_softpath_lastx_fp .....
    1280, 1294, 1315, 1363, 1423, 1430
\g__draw_softpath_lasty_dim ....
    598, 615, 1205, 1215, 1239
\l__draw_softpath_lasty_dim ....
    586, 598, 615
\l__draw_softpath_lasty_fp .....
    1280, 1295, 1316, 1364, 1424, 1431
\__draw_softpath_lineto:nn . 98, 1209
\__draw_softpath_lineto_op:nn ...
    1230, 1257, 1351, 1382, 1493
\g__draw_softpath_main_tl .....
    601, 613, 1139,
    1171, 1173, 1189, 1195, 1300, 1504
\l__draw_softpath_main_tl .....
    15, 601, 613, 1277,
    1292, 1319, 1321, 1500, 1502, 1504
\g__draw_softpath_move_bool ....
    1207, 1236
\l__draw_softpath_move_tl .....
    1280, 1297,
    1320, 1323, 1371, 1471, 1494, 1501
\__draw_softpath_moveto:nn . 85, 1209
\__draw_softpath_moveto_op:nn ...
    1235, 1257, 1317, 1473
\l__draw_softpath_part_tl .....
    1278, 1293,
    1322, 1325, 1327, 1361, 1414, 1503
\__draw_softpath_rectangle:nnnn ..
    420, 1209

```

_draw_softpath_rectangle_- opi:nn	1246, 1257	_g__draw_xmin_dim	34, 35, 1031, 1041, 1064, 1067, 1073, 1081, 1111, 1119
_draw_softpath_rectangle_- opi:nnNnn	1257	_l__draw_xmin_dim	1103, 1111, 1119
_draw_softpath_rectangle_- opii:nn	1247, 1257	_l__draw_xshift_dim	986, 1000, 1558, 1573, 1601, 1633, 1667
_draw_softpath_reset_buffers:	1145	_l__draw_xvec_x_dim	905, 935, 949, 967
_draw_softpath_round_action:nn	1287	_l__draw_xvec_y_dim	905, 936, 953
_draw_softpath_round_action:Nnn	1287	_g__draw_ymax_dim	36, 37, 1031, 1042, 1068, 1078, 1112, 1120
_draw_softpath_round_action_- close:	1287	_l__draw_ymax_dim	1103, 1112, 1120
_draw_softpath_round_action_- curveto:NnnNnn	1287	_g__draw_ymin_dim	38, 39, 1031, 1043, 1069, 1074, 1078, 1113, 1121
_draw_softpath_round_calc:nnnNnn	1287	_l__draw_ymin_dim	1103, 1113, 1121
_draw_softpath_round_calc:nnnnnn	1287	_l__draw_yshift_dim	992, 1000, 1558, 1574, 1602, 1634, 1668
_draw_softpath_round_calc:nnnnw	1287	_l__draw_yvec_x_dim	905, 935, 950
_draw_softpath_round_close:nn	1287	_l__draw_yvec_y_dim	905, 936, 954, 968
_draw_softpath_round_close:w	1287	_l__draw_zvec_x_dim	905, 951
_draw_softpath_round_corners:	525, 1287	_l__draw_zvec_y_dim	905, 955
_draw_softpath_round_end:	1287	driver commands:	
_draw_softpath_round_lookahead:NnnNnn	1287	_driver_draw_begin:	1050
_draw_softpath_round_loop:Nnn	1287	_driver_draw_cap_but:	1522
_draw_softpath_round_roundpoint:NnnNnnNnn	1287	_driver_draw_cap_rectangle:	1523
_draw_softpath_roundpoint:nn	71, 1209	_driver_draw_cap_round:	1524
_draw_softpath_roundpoint_- op:nn	1253, 1257, 1310, 1391	_driver_draw_clip:	534
_draw_softpath_use:	532, 1187	_driver_draw_closepath:	1258
_draw_softpath_use_clear:	531, 1187	_driver_draw_curveto:nnnnnn	1262
_draw_split_select:nw	1531	_driver_draw_end:	1062
_l__draw_stroke_color_tl	1086	_driver_draw_evenodd_rule:	1525
_draw_tranform_triangle:nnnnnn	1675, 1680	_driver_draw_join_bevel:	1527
_draw_transform_concat:nnnn	1604	_driver_draw_join_miter:	1528
_draw_transform_invert:n	1636	_driver_draw_join_round:	1529
_draw_transform_rotate:n	1706	_driver_draw_lineto:nn	1268
_draw_transform_rotate:nn	1706	_driver_draw_linewidth:n	1516
_draw_transform_shift:nn	1578	_driver_draw_miterlimit:n	1521
_draw_transform_shift_concat:nn	1604	_driver_draw_moveto:nn	1270
_draw_vec:nn	911	_driver_draw_nonzero_rule:	1526
_draw_vec:nnn	911	_driver_draw_rectangle:nnnn	1275
_g__draw_xmax_dim	32, 33, 1031, 1040, 1066, 1081, 1110, 1118	_driver_draw_scope_begin:	1091
_l__draw_xmax_dim	1103, 1110, 1118	_driver_draw_scope_end:	1101
		E	
		_ERROR	522
		exp commands:	
		_exp_after:wN	292, 310, 1299, 1373, 1474, 1485, 1494
		_exp_not:N	1416, 1444, 1453, 1462, 1471, 1474, 1475, 1476, 1477, 1481, 1485, 1486, 1487, 1488
		_exp_not:n	1158, 1159, 1160, 1173, 1174, 1175, 1176
		F	
		fp commands:	
		_fp_compare:nNnTF	203, 213

<code>\fp_compare_p:nNn</code>	1586, 1587, 1588, 1589		
<code>\fp_const:Nn</code>	186, 187, 328, 329, 400, 1286		
<code>\fp_eval:n</code>	195, 196, 217, 224, 233, 651, 661, 690, 691, 692, 693, 694, 695, 715, 720, 721, 728, 735, 736, 743, 750, 752, 765, 770, 788, 808, 813, 820, 821, 844, 851, 873, 874, 893, 894, 895, 896, 930, 943, 962, 1521, 1609, 1610, 1611, 1612, 1642, 1707, 1711, 1712		
<code>\fp_new:N</code>	12, 13, 326, 327, 1280, 1281, 1558, 1559, 1560, 1561		
<code>\fp_set:Nn</code>	154, 155, 209, 210, 293, 294, 1315, 1316, 1363, 1364, 1430, 1431, 1566, 1569, 1580, 1581, 1582, 1583, 1655, 1657, 1659, 1661		
<code>\fp_to_decimal:N</code>	216, 223, 231		
<code>\fp_to_dim:n</code>	161, 168, 175, 179, 197, 198, 244, 253, 324, 351, 363, 364, 365, 366, 367, 368, 373, 374, 375, 376, 377, 378, 383, 384, 385, 386, 387, 388, 393, 394, 395, 396, 397, 398, 466, 467, 1446, 1450, 1455, 1459, 1515, 1519		
<code>\fp_use:N</code>	400		
<code>\fp_while_do:nNnn</code>	211		
<code>\fp_zero:N</code>	1294, 1295, 1567, 1568		
<code>\c_one_fp</code>	1586, 1589		
<code>\c_zero_fp</code>	1587, 1588		
		G	
group commands:			
<code>\group_begin:</code>	590, 1047, 1092, 1109, 1291		
<code>\group_end:</code>	622, 1084, 1100, 1122, 1303		
		H	
hbox commands:			
<code>\hbox_set:Nn</code>	1071		
<code>\hbox_set:Nw</code>	1049		
<code>\hbox_set_end:</code>	1063		
		I	
int commands:			
<code>\int_compare:nNnTF</code>	1147, 1153		
<code>\int_gincr:N</code>	1048, 1149, 1155		
<code>\int_gzero:N</code>	1162, 1182, 1184		
<code>\int_if_odd:nTF</code>	751		
<code>\int_new:N</code>	1037, 1142, 1143		
		M	
mode commands:			
<code>\mode_leave_vertical:</code>	1082		
		P	
<code>\pgfextractx</code>	16		
<code>\pgfextracty</code>	16		
<code>\pgfgetlastxy</code>	16		
<code>\pgfgettransform</code>	43		
<code>\pgfgettransformentries</code>	43		
<code>\pgflowlevel</code>	43		
<code>\pgflowlevelsynccm</code>	43		
<code>\pgfpatharcto</code>	1		
<code>\pgfpatharctoprecomputed</code>	1		
<code>\pgfpathcosine</code>	1		
<code>\pgfpathcurvebetweentime</code>	1		
<code>\pgfpathcurvebetweentimecontinue</code>	1		
<code>\pgfpathparabola</code>	1		
<code>\pgfpathsine</code>	1		
<code>\pgfpointborderellipse</code>	16		
<code>\pgfpointborderrectangle</code>	16		
<code>\pgfpointcylindrical</code>	16		
<code>\pgfpointorigin</code>	16		
<code>\pgfpointspherical</code>	16		
<code>\pgfqpoint</code>	17		
<code>\pgfqpointpolar</code>	17		
<code>\pgfqpointscale</code>	17		
<code>\pgfqpointxy</code>	17		
<code>\pgfqpointxyz</code>	17		
<code>\pgftransformarcsaxesatime</code>	43		
<code>\pgftransformarrow</code>	43		
<code>\pgftransformcurveatime</code>	43		
<code>\pgftransformlineatime</code>	43		
prg commands:			
<code>\prg_do_nothing:</code>	866, 877, 880		
<code>\ProvidesExplPackage</code>	4		
		Q	
quark commands:			
<code>\q_mark</code>	646, 648, 1437, 1441		
<code>\quark_if_recursion_tail_stop_</code> do:Nn	1309		
<code>\q_recursion_stop</code>	1302		
<code>\q_recursion_tail</code>	1301		
<code>\q_stop</code>	634, 636, 646, 648, 1437, 1441, 1477, 1488, 1497, 1542, 1544, 1545, 1546, 1548, 1550, 1552		
		R	
<code>\RequirePackage</code>	7		
		S	
skip commands:			
<code>\skip_horizontal:n</code>	1073		
str commands:			
<code>\str_if_eq_p:nn</code>	504		

T			
tl commands:		\tl_set:Nn	
\tl_clear:N	266, 1292, 1293, 1296, 1297, 1324, 1325	. . . 315, 1314, 1323, 1344, 1408, 1471	
\tl_gclear:N 1163, 1183, 1185, 1195, 1196, 1197	\tl_set_eq:NN	601
\tl_gput_right:Nn	1150	token commands:	
\tl_gset:Nn	1156, 1171	\token_if_eq_meaning:NNTF	
\tl_gset_eq:NN	613, 1504	. . 1310, 1317, 1345, 1348, 1351, 1391	
\tl_if_blank:nTF	497	\token_if_eq_meaning_p:NN	
\tl_if_blank_p:n	503 1382, 1383, 1384	
\tl_if_empty:NNTF	1313, 1343	U	
\tl_if_empty_p:N	1370, 1371	use commands:	
\tl_new:N 11, 1087, 1088, 1139, 1140, 1141, 1277, 1278, 1279, 1284, 1285, 1530	\use:N	539, 570, 1545, 1547, 1549, 1551, 1553
\tl_put_right:Nn . . 319, 323, 1319, 1321, 1327, 1361, 1414, 1500, 1502		\use:n	156, 192, 239, 348, 1467, 1479, 1606, 1617, 1682
		\use_i:nn	16
		\use_i:nnnn	1481
		\use_ii:nn	16
		\use_none:n	1494