

File I

Implementation

1 l3draw implementation

```
1 <*package>
2 <@@=draw>
3 \ProvidesExplPackage{l3draw}{2021-02-18}{}
4 {L3 Experimental core drawing support}
```

1.1 Internal auxiliaries

`\s__draw_mark` Internal scan marks.

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

(End definition for \s__draw_mark and \s__draw_stop.)

`\q__draw_recursion_tail` Internal recursion quarks.

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

(End definition for \q__draw_recursion_tail and \q__draw_recursion_stop.)

`_draw_if_recursion_tail_stop_do:Nn` Functions to query recursion quarks.

```
9 \_kernel_quark_new_test:N \_draw_if_recursion_tail_stop_do:Nn
```

(End definition for _draw_if_recursion_tail_stop_do:Nn.)

Everything else is in the sub-files!

```
10 </package>
```

2 l3draw-boxes implementation

```
11 <*package>
```

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

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

`\l__draw_tmp_box`

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

(End definition for \l__draw_tmp_box.)

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

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

```
15 {
```

```

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

```

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

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

```

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

```

```

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

```

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

```
72 </package>
```

3 I3draw-layers implementation

```
73 <*package>
```

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

3.1 User interface

`\draw_layer_new:n`

```

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

```

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

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

```

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

```

(End definition for `\l__draw_layer_tl`.)

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

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

(End definition for `\l__draw_layer_close_bool`.)

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

```

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

```

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

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

```

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

```

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

3.2 Internal cross-links

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

```

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

```

```

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

```

(End definition for `__draw_layers_insert:.`)

```

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

```

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

```

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

```

4 l3draw-paths implementation

```

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

```

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

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

`\l__draw_path_tmp_tl` Scratch space.

```

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

```

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

4.1 Tracking paths

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

```

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

```

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

`\g__draw_path_xmax_dim` The limiting size of a path.

```

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

```

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

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

```

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

```

```

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

```

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

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

```

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

```

(End definition for `__draw_path_update_last:nn`.)

4.2 Corner arcs

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

`\l__draw_corner_xarc_dim` The two arcs in use.

```

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

```

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

`\l__draw_corner_arc_bool` A flag to speed up the repeated checks.

```

220 \bool_new:N \l__draw_corner_arc_bool

```

(End definition for `\l__draw_corner_arc_bool`.)

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

```

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

```

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

```

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

```

(End definition for `__draw_path_mark_corner:.`)

4.3 Basic path constructions

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

```

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

```



```

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

```

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

`\draw_path_close:` A simple wrapper.

```

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

```

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

4.4 Canvas path constructions

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

```

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

```

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

4.5 Computed curves

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

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

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

`\c__draw_path_curveto_a_fp`
`\c__draw_path_curveto_b_fp`

and

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

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

```

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

```

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

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

```

\draw_path_arc:nnnn
\__draw_path_arc:nnNnn
\__draw_path_arc_auxi:nnnnNnn
\__draw_path_arc_auxi:fnnnNnn
\__draw_path_arc_auxi:fnfnNnn
\__draw_path_arc_auxii:nnnNnnnn
\__draw_path_arc_auxiii:nn
\__draw_path_arc_auxiv:nnnn
\__draw_path_arc_auxv:nn
\__draw_path_arc_auxvi:nn
\__draw_path_arc_add:nnnn
\l__draw_path_arc_delta_fp
\l__draw_path_arc_start_fp
\c__draw_path_arc_90_fp
\c__draw_path_arc_60_fp

```

```

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

```

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

```

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

```

```

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

```

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

```

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

```

```

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

```

The first control point.

```

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

```

The end point: simple arithmetic.

```

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

```

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

```

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

```

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

`\draw_path_arc_axes:nmmm` A simple wrapper.

```

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

```

(End definition for \draw_path_arc_axes:nmmm. 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
\__draw_path_ellipse:nnnnnn same constant. We need to deal with the ellipse in four parts and also deal with moving
  \_draw_path_ellipse_arci:nnnnnn to the right place, closing it and ending up back at the center. That is handled on a
  \_draw_path_ellipse_arcii:nnnnnn per-arc basis, each in a separate auxiliary for readability.
  \_draw_path_ellipse_arciiii:nnnnnn
  \_draw_path_ellipse_arciiv:nnnnnn
\c__draw_path_ellipse_fp
489 \cs_new_protected:Npn \draw_path_ellipse:nnn #1#2#3
490 {
491   \__draw_point_process:nnnn
492   { \_draw_path_ellipse:nnnnnn }
493   { \draw_point_transform:n {#1} }
494   { \__draw_point_transform_noshift:n {#2} }
495   { \__draw_point_transform_noshift:n {#3} }
496 }
497 \cs_new_protected:Npn \__draw_path_ellipse:nnnnnn #1#2#3#4#5#6
498 {
499   \use:x
500   {
501     \__draw_path_moveto:nn
502     { \fp_to_dim:n { #1 + #3 } } { \fp_to_dim:n { #2 + #4 } }
503     \__draw_path_ellipse_arci:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
504     \__draw_path_ellipse_arcii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
505     \__draw_path_ellipse_arciiii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
506     \__draw_path_ellipse_arciiv:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
507   }
508   \__draw_softpath_closepath:
509   \__draw_path_moveto:nn {#1} {#2}
510 }
511 \cs_new:Npn \__draw_path_ellipse_arci:nnnnnn #1#2#3#4#5#6
512 {
513   \__draw_path_curveto:nnnnnn
514   { \fp_to_dim:n { #1 + #3 + #5 * \c__draw_path_ellipse_fp } }
515   { \fp_to_dim:n { #2 + #4 + #6 * \c__draw_path_ellipse_fp } }
516   { \fp_to_dim:n { #1 + #3 * \c__draw_path_ellipse_fp + #5 } }
517   { \fp_to_dim:n { #2 + #4 * \c__draw_path_ellipse_fp + #6 } }
518   { \fp_to_dim:n { #1 + #5 } }
519   { \fp_to_dim:n { #2 + #6 } }
520 }
521 \cs_new:Npn \__draw_path_ellipse_arcii:nnnnnn #1#2#3#4#5#6
522 {
523   \__draw_path_curveto:nnnnnn
524   { \fp_to_dim:n { #1 - #3 * \c__draw_path_ellipse_fp + #5 } }
525   { \fp_to_dim:n { #2 - #4 * \c__draw_path_ellipse_fp + #6 } }
526   { \fp_to_dim:n { #1 - #3 + #5 * \c__draw_path_ellipse_fp } }
527   { \fp_to_dim:n { #2 - #4 + #6 * \c__draw_path_ellipse_fp } }
528   { \fp_to_dim:n { #1 - #3 } }
529   { \fp_to_dim:n { #2 - #4 } }
530 }
531 \cs_new:Npn \__draw_path_ellipse_arciiii:nnnnnn #1#2#3#4#5#6
532 {
533   \__draw_path_curveto:nnnnnn
534   { \fp_to_dim:n { #1 - #3 - #5 * \c__draw_path_ellipse_fp } }
535   { \fp_to_dim:n { #2 - #4 - #6 * \c__draw_path_ellipse_fp } }
536   { \fp_to_dim:n { #1 - #3 * \c__draw_path_ellipse_fp - #5 } }
537   { \fp_to_dim:n { #2 - #4 * \c__draw_path_ellipse_fp - #6 } }
538   { \fp_to_dim:n { #1 - #5 } }

```

```

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

```

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

`\draw_path_circle:nn` A shortcut.

```

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

```

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

4.6 Rectangles

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

```

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

```

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

```

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

```

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

4.7 Grids

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

```

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

```



```

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

```

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

4.8 Using paths

```

\l__draw_path_use_clip_bool
\l__draw_path_use_fill_bool
  \l__draw_path_use_stroke_bool

```

Actions to pass to the driver.

```

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

```

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

```

\l__draw_path_use_bb_bool
\l__draw_path_use_clear_bool

```

Actions handled at the macro layer.

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

```

713     \bool_set_true:N \l__draw_path_use_stroke_bool
714 }

```

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

```

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

```

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

4.9 Scoping paths

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

```

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

```

(End definition for \l__draw_path_lastx_dim and others.)

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

```

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

```

```

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

```

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

```

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

```

5 l3draw-points implementation

```

788 <*package>
789 <@@=draw>

```

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

- `\pgfpointorigin`: Can be given explicitly as `0pt,0pt`.
- `\pgfpointadd`, `\pgfpointdiff`, `\pgfpointscale`: Can be given explicitly.


```

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

```

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

```

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

```

5.2 Polar co-ordinates

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

```

\draw_point_polar:nn
\draw_point_polar:nnn
\__draw_draw_polar:nnn 847 \cs_new:Npn \draw_point_polar:nn #1#2
\__draw_draw_polar:fnn 848 { \draw_point_polar:nnn {#1} {#1} {#2} }
849 \cs_new:Npn \draw_point_polar:nnn #1#2#3
850 { \__draw_draw_polar:fnn { \fp_eval:n {#3} } {#1} {#2} }
851 \cs_new:Npn \__draw_draw_polar:nnn #1#2#3
852 { \__draw_point_to_dim:n { cosd(#1) * (#2) , sind(#1) * (#3) } }
853 \cs_generate_variant:Nn \__draw_draw_polar:nnn { f }

```

5.3 Point expression arithmetic

These functions all take point expressions as arguments.

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

```

\draw_point_unit_vector:n
\__draw_point_unit_vector:nn
\__draw_point_unit_vector:nnm

```

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

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

```

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

```

5.4 Intersection calculations

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

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

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.)

```

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

```

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

```

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

```

#8 y_4

so now just have to do all of the calculation.

```

877 \cs_new:Npn \__draw_point_intersect_lines:nnnnnnnn #1#2#3#4#5#6#7#8
878 {
879   \__draw_point_intersect_lines_aux:fffff
880   { \fp_eval:n { #1 * #4 - #2 * #3 } }
881   { \fp_eval:n { #5 * #8 - #6 * #7 } }
882   { \fp_eval:n { #1 - #3 } }
883   { \fp_eval:n { #5 - #7 } }
884   { \fp_eval:n { #2 - #4 } }
885   { \fp_eval:n { #6 - #8 } }
886 }
887 \cs_new:Npn \__draw_point_intersect_lines_aux:nnnnnn #1#2#3#4#5#6
888 {
889   \__draw_point_to_dim:n
890   {
891     ( #2 * #3 - #1 * #4 , #2 * #5 - #1 * #6 )
892     / ( #4 * #5 - #6 * #3 )
893   }
894 }
895 \cs_generate_variant:Nn \__draw_point_intersect_lines_aux:nnnnnn { fffff }

```

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

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

in either

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

or

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

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

```

896 \cs_new:Npn \draw_point_intersect_circles:nnnnn #1#2#3#4#5
897 {
898   \__draw_point_process:nnn
899   { \__draw_point_intersect_circles_auxi:nnnnnnn {#2} {#4} {#5} }

```



```

900     {#1} {#3}
901   }
902 \cs_new:Npn \__draw_point_intersect_circles_auxi:nnnnnnn #1#2#3#4#5#6#7
903   {
904     \__draw_point_intersect_circles_auxii:ffnnnnn
905     { \fp_eval:n {#1} } { \fp_eval:n {#2} } {#4} {#5} {#6} {#7} {#3}
906   }

```

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

```

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

```

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

```

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

```

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

```

922 \cs_new:Npn \__draw_point_intersect_circles_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
923   {
924     \__draw_point_intersect_circles_auxv:ffnnnnnnnn
925     { \fp_eval:n { 1 / #1 } }
926     { \fp_eval:n { #4 * #4 } }
927     {#1} {#2} {#3} {#5} {#6} {#7} {#8}
928   }
929 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxiv:nnnnnnnn { f }
930 \cs_new:Npn \__draw_point_intersect_circles_auxv:nnnnnnnnn #1#2#3#4#5#6#7#8#9
931   {
932     \__draw_point_intersect_circles_auxvi:fnnnnnnnn
933     { \fp_eval:n { 0.5 * #1 * ( #2 + #3 * #3 - #6 * #6 ) } }

```

```

934     {#1} {#2} {#4} {#5} {#7} {#8} {#9}
935   }
936 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxv:nnnnnnnn { ff }

```

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

```

#1  $k$ 
#2  $1/p$ 
#3  $r^2$ 
#4  $e$ 
#5  $f$ 
#6  $a$ 
#7  $b$ 
#8  $n$ 

```

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

```

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

```

5.5 Interpolation on a line (vector) or arc

Simple maths after expansion.

```

\draw_point_interpolate_line:nnn 952 \cs_new:Npn \draw_point_interpolate_line:nnn #1#2#3
\_draw_point_interpolate_line_aux:nnnnn 953 {
\_draw_point_interpolate_line_aux:fnnnn 954   \__draw_point_process:nnn
\_draw_point_interpolate_line_aux:nnnnnn 955   { \__draw_point_interpolate_line_aux:fnnnn { \fp_eval:n {#1} } }
\_draw_point_interpolate_line_aux:fnnnnnn 956   {#2} {#3}
957 }
958 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnnn #1#2#3#4#5
959 {
960   \__draw_point_interpolate_line_aux:fnnnnnn { \fp_eval:n { 1 - #1 } }
961   {#1} {#2} {#3} {#4} {#5}
962 }
963 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnnn { f }

```

```

964 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnnn #1#2#3#4#5#6
965 { \__draw_point_to_dim:n { #2 * #3 + #1 * #5 , #2 * #4 + #1 * #6 } }
966 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnnn { f }

```

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

```

\draw_point_interpolate_distance:nnn
\__draw_point_interpolate_distance:nnnnn
\__draw_point_interpolate_distance:nnnnnn
\__draw_point_interpolate_distance:fnnnnn

```

```

967 \cs_new:Npn \draw_point_interpolate_distance:nnn #1#2#3
968 {
969   \__draw_point_process:nn
970   { \__draw_point_interpolate_distance:nnnn {#1} {#3} }
971   {#2}
972 }
973 \cs_new:Npn \__draw_point_interpolate_distance:nnnn #1#2#3#4
974 {
975   \__draw_point_process:nn
976   {
977     \__draw_point_interpolate_distance:fnnnn
978     { \fp_eval:n {#1} } {#3} {#4}
979   }
980   { \draw_point_unit_vector:n { ( #2 ) - ( #3 , #4 ) } }
981 }
982 \cs_new:Npn \__draw_point_interpolate_distance:nnnnn #1#2#3#4#5
983 { \__draw_point_to_dim:n { #2 + #1 * #4 , #3 + #1 * #5 } }
984 \cs_generate_variant:Nn \__draw_point_interpolate_distance:nnnnn { f }

```

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

```

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

```

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.

```

985 \cs_new:Npn \draw_point_interpolate_arcaxes:nnnnnn #1#2#3#4#5#6
986 {
987   \__draw_point_process:nnnn
988   { \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnn {#1} {#5} {#6} }
989   {#2} {#3} {#4}
990 }
991 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnn #1#2#3#4#5#6#7#8#9
992 {
993   \__draw_point_interpolate_arcaxes_auxiii:fnnnnnnnn
994   { \fp_eval:n {#1} } {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
995 }

```

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

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

#7 y_{a1}

#8 x_{a2}

#9 y_{a2}

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

```
996 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnn #1#2#3#4#5#6#7#8#9
997 {
998   \__draw_point_interpolate_arcaxes_auxiii:fnnnnnnn
999   { \fp_eval:n { #1 * (#3) + ( 1 - #1 ) * (#2) } }
1000   {#4} {#5} {#6} {#7} {#8} {#9}
1001 }
1002 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnn { f }
1003 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnn #1#2#3#4#5#6#7
1004 {
1005   \__draw_point_interpolate_arcaxes_auxiv:ffnnnnnnn
1006   { \fp_eval:n { cosd (#1) } }
1007   { \fp_eval:n { sind (#1) } }
1008   {#2} {#3} {#4} {#5} {#6} {#7}
1009 }
1010 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnn { f }
1011 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnnn #1#2#3#4#5#6#7#8
1012 {
1013   \__draw_point_to_dim:n
1014   { #3 + #1 * #5 + #2 * #7 , #4 + #1 * #6 + #2 * #8 }
1015 }
1016 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnnn { ff }
```

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

```
\draw_point_interpolate_curve:nnnnn
draw_point_interpolate_curve_auxi:nnnnnnnnn
draw_point_interpolate_curve_auxii:nnnnnnnnn
draw_point_interpolate_curve_auxiii:fnnnnnnnnn
\draw_point_interpolate_curve_auxiiii:nnnnnnn
\draw_point_interpolate_curve_auxiiiii:fnnnnnnn
\draw_point_interpolate_curve_auxiv:nnnnnnn
\draw_point_interpolate_curve_auxv:nnw
\draw_point_interpolate_curve_auxv:ffw
\draw_point_interpolate_curve_auxvi:n
draw_point_interpolate_curve_auxvii:nnnnnnnnn
draw_point_interpolate_curve_auxviii:nnnnnnn
draw_point_interpolate_curve_auxviiii:ffnnnnn
```

Here we start with a proportion of the curve (p) and four points

1. The initial point (x_1, y_1)
2. The first control point (x_2, y_2)
3. The second control point (x_3, y_3)
4. The final point (x_4, y_4)

The first phase is to expand out all of these values.

```
1017 \cs_new:Npn \draw_point_interpolate_curve:nnnnnn #1#2#3#4#5
1018 {
1019   \__draw_point_process:nnnnnn
1020   { \__draw_point_interpolate_curve_auxi:nnnnnnnnn {#1} }
1021   {#2} {#3} {#4} {#5}
1022 }
1023 \cs_new:Npn \__draw_point_interpolate_curve_auxi:nnnnnnnnn #1#2#3#4#5#6#7#8#9
1024 {
1025   \__draw_point_interpolate_curve_auxii:fnnnnnnnnn
1026   { \fp_eval:n {#1} }
1027   {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1028 }
```

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!

```

1029 \cs_new:Npn \__draw_point_interpolate_curve_auxii:nnnnnnnn
1030   #1#2#3#4#5#6#7#8#9
1031   {
1032     \__draw_point_interpolate_curve_auxiii:fnnnnn
1033     { \fp_eval:n { 1 - #1 } }
1034     {#1}
1035     { {#2} {#3} } { {#4} {#5} } { {#6} {#7} } { {#8} {#9} }
1036   }
1037 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxii:nnnnnnnn { f }
1038 % \begin{macrocode}
1039 % We need to do the first cycle, but haven't got enough arguments to keep
1040 % everything in play at once. So here we use a bit of argument re-ordering
1041 % and a single auxiliary to get the job done.
1042 % \begin{macrocode}
1043 \cs_new:Npn \__draw_point_interpolate_curve_auxiii:nnnnnn #1#2#3#4#5#6
1044   {
1045     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #3 #4
1046     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #4 #5
1047     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #5 #6
1048     \prg_do_nothing:
1049     \__draw_point_interpolate_curve_auxvi:n { {#1} {#2} }
1050   }
1051 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxiii:nnnnnn { f }
1052 \cs_new:Npn \__draw_point_interpolate_curve_auxiv:nnnnnn #1#2#3#4#5#6
1053   {
1054     \__draw_point_interpolate_curve_auxv:ffw
1055     { \fp_eval:n { #1 * #3 + #2 * #5 } }

```

```

1056     { \fp_eval:n { #1 * #4 + #2 * #6 } }
1057   }
1058 \cs_new:Npn \__draw_point_interpolate_curve_auxv:nnw
1059   #1#2#3 \prg_do_nothing: #4#5
1060   {
1061     #3
1062     \prg_do_nothing:
1063     #4 { #5 {#1} {#2} }
1064   }
1065 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxv:nnw { ff }
1066 %   \begin{macrocode}
1067 %   Get the arguments back into the right places and to the second and
1068 %   third cycles directly.
1069 %   \begin{macrocode}
1070 \cs_new:Npn \__draw_point_interpolate_curve_auxvi:n #1
1071   { \__draw_point_interpolate_curve_auxvii:nnnnnnn #1 }
1072 \cs_new:Npn \__draw_point_interpolate_curve_auxvii:nnnnnnn #1#2#3#4#5#6#7#8
1073   {
1074     \__draw_point_interpolate_curve_auxviii:ffffnn
1075     { \fp_eval:n { #1 * #5 + #2 * #3 } }
1076     { \fp_eval:n { #1 * #6 + #2 * #4 } }
1077     { \fp_eval:n { #1 * #7 + #2 * #5 } }
1078     { \fp_eval:n { #1 * #8 + #2 * #6 } }
1079     {#1} {#2}
1080   }
1081 \cs_new:Npn \__draw_point_interpolate_curve_auxviii:nnnnnn #1#2#3#4#5#6
1082   {
1083     \__draw_point_to_dim:n
1084     { #5 * #3 + #6 * #1 , #5 * #4 + #6 * #2 }
1085   }
1086 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxviii:nnnnnn { ffff }

```

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

5.6 Vector support

As well as co-ordinates relative to the drawing

```

\l__draw_xvec_x_dim Base vectors to map to the underlying two-dimensional drawing space.
\l__draw_xvec_y_dim 1087 \dim_new:N \l__draw_xvec_x_dim
\l__draw_yvec_x_dim 1088 \dim_new:N \l__draw_xvec_y_dim
\l__draw_yvec_y_dim 1089 \dim_new:N \l__draw_yvec_x_dim
\l__draw_zvec_x_dim 1090 \dim_new:N \l__draw_yvec_y_dim
\l__draw_zvec_y_dim 1091 \dim_new:N \l__draw_zvec_x_dim
1092 \dim_new:N \l__draw_zvec_y_dim

```

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

```

\draw_xvec:n Calculate the underlying position and store it.
\draw_yvec:n 1093 \cs_new_protected:Npn \draw_xvec:n #1
\draw_zvec:n 1094 { \__draw_vec:nn { x } {#1} }
\__draw_vec:nn 1095 \cs_new_protected:Npn \draw_yvec:n #1
\__draw_vec:nnn 1096 { \__draw_vec:nn { y } {#1} }

```

```

1097 \cs_new_protected:Npn \draw_zvec:n #1
1098 { \__draw_vec:nn { z } {#1} }
1099 \cs_new_protected:Npn \__draw_vec:nn #1#2
1100 {
1101   \__draw_point_process:nn { \__draw_vec:nnn {#1} } {#2}
1102 }
1103 \cs_new_protected:Npn \__draw_vec:nnn #1#2#3
1104 {
1105   \dim_set:cn { l__draw_ #1 vec_x_dim } {#2}
1106   \dim_set:cn { l__draw_ #1 vec_y_dim } {#3}
1107 }

```

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

Initialise the vectors.

```

1108 \draw_xvec:n { 1cm , 0cm }
1109 \draw_yvec:n { 0cm , 1cm }
1110 \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 1111 \cs_new:Npn \draw_point_vec:nn #1#2
\__draw_point_vec:ff 1112 { \__draw_point_vec:ff { \fp_eval:n {#1} } { \fp_eval:n {#2} } }
\draw_point_vec:nnn 1113 \cs_new:Npn \__draw_point_vec:nn #1#2
\__draw_point_vec:nnn 1114 {
\__draw_point_vec:fff 1115   \__draw_point_to_dim:n
1116   {
1117     #1 * \l__draw_xvec_x_dim + #2 * \l__draw_yvec_x_dim ,
1118     #1 * \l__draw_xvec_y_dim + #2 * \l__draw_yvec_y_dim
1119   }
1120 }
1121 \cs_generate_variant:Nn \__draw_point_vec:nn { ff }
1122 \cs_new:Npn \draw_point_vec:nnn #1#2#3
1123 {
1124   \__draw_point_vec:fff
1125   { \fp_eval:n {#1} } { \fp_eval:n {#2} } { \fp_eval:n {#3} }
1126 }
1127 \cs_new:Npn \__draw_point_vec:nnn #1#2#3
1128 {
1129   \__draw_point_to_dim:n
1130   {
1131     #1 * \l__draw_xvec_x_dim
1132     + #2 * \l__draw_yvec_x_dim
1133     + #3 * \l__draw_zvec_x_dim
1134     ,
1135     #1 * \l__draw_xvec_y_dim
1136     + #2 * \l__draw_yvec_y_dim
1137     + #3 * \l__draw_zvec_y_dim
1138   }
1139 }
1140 \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 1141 \cs_new:Npn \draw_point_vec_polar:nn #1#2
\__draw_point_vec_polar:nnn
\__draw_point_vec_polar:fnn

```

```

1142 { \draw_point_vec_polar:nnn {#1} {#1} {#2} }
1143 \cs_new:Npn \draw_point_vec_polar:nnn #1#2#3
1144 { \__draw_draw_vec_polar:fnn { \fp_eval:n {#3} } {#1} {#2} }
1145 \cs_new:Npn \__draw_draw_vec_polar:nnn #1#2#3
1146 {
1147   \__draw_point_to_dim:n
1148   {
1149     cosd(#1) * (#2) * \l__draw_xvec_x_dim ,
1150     sind(#1) * (#3) * \l__draw_yvec_y_dim
1151   }
1152 }
1153 \cs_generate_variant:Nn \__draw_draw_vec_polar:nnn { f }

```

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

5.7 Transformations

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

```

1154 \cs_new:Npn \draw_point_transform:n #1
1155 {
1156   \__draw_point_process:nn
1157   { \__draw_point_transform:nn } {#1}
1158 }
1159 \cs_new:Npn \__draw_point_transform:nn #1#2
1160 {
1161   \bool_if:NTF \l__draw_matrix_active_bool
1162   {
1163     \__draw_point_to_dim:n
1164     {
1165       (
1166         \l__draw_matrix_a_fp * #1
1167         + \l__draw_matrix_c_fp * #2
1168         + \l__draw_xshift_dim
1169       )
1170     ,
1171     (
1172       \l__draw_matrix_b_fp * #1
1173       + \l__draw_matrix_d_fp * #2
1174       + \l__draw_yshift_dim
1175     )
1176   }
1177 }
1178 {
1179   \__draw_point_to_dim:n
1180   {
1181     (#1, #2)
1182     + ( \l__draw_xshift_dim , \l__draw_yshift_dim )
1183   }
1184 }
1185 }

```

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


```

\__draw_point_transform_noshift:n A version with no shift: used for internal purposes.
\__draw_point_transform_noshift:nn
1186 \cs_new:Npn \__draw_point_transform_noshift:n #1
1187 {
1188   \__draw_point_process:nn
1189   { \__draw_point_transform_noshift:nn } {#1}
1190 }
1191 \cs_new:Npn \__draw_point_transform_noshift:nn #1#2
1192 {
1193   \bool_if:NTF \l__draw_matrix_active_bool
1194   {
1195     \__draw_point_to_dim:n
1196     {
1197       (
1198         \l__draw_matrix_a_fp * #1
1199         + \l__draw_matrix_c_fp * #2
1200       )
1201       ,
1202       (
1203         \l__draw_matrix_b_fp * #1
1204         + \l__draw_matrix_d_fp * #2
1205       )
1206     }
1207   }
1208   { \__draw_point_to_dim:n { (#1, #2) } }
1209 }

```

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

```
1210 </package>
```

6 l3draw-scopes implementation

```
1211 <*package>
```

```
1212 <@@=draw>
```

6.1 Drawing environment

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

```

\g__draw_xmin_dim
\g__draw_ymax_dim
\g__draw_ymin_dim
1213 \dim_new:N \g__draw_xmax_dim
1214 \dim_new:N \g__draw_xmin_dim
1215 \dim_new:N \g__draw_ymax_dim
1216 \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.

```
1217 \bool_new:N \l_draw_bb_update_bool
```

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

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

```

1218 \box_new:N \l__draw_main_box
1219 \box_new:N \l__draw_layer_main_box

```

(End definition for \l__draw_layer_main_box.)

\g__draw_id_int The drawing number.

```
1220 \int_new:N \g__draw_id_int
```

(End definition for \g__draw_id_int.)

__draw_reset_bb: A simple auxiliary.

```
1221 \cs_new_protected:Npn \__draw_reset_bb:
1222 {
1223   \dim_gset:Nn \g__draw_xmax_dim { -\c_max_dim }
1224   \dim_gset:Nn \g__draw_xmin_dim { \c_max_dim }
1225   \dim_gset:Nn \g__draw_ymax_dim { -\c_max_dim }
1226   \dim_gset:Nn \g__draw_ymin_dim { \c_max_dim }
1227 }
```

(End definition for __draw_reset_bb:.)

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

```
1228 \cs_new_protected:Npn \draw_begin:
1229 {
1230   \group_begin:
1231     \int_gincr:N \g__draw_id_int
1232     \hbox_set:Nw \l__draw_main_box
1233     \__draw_backend_begin:
1234     \__draw_reset_bb:
1235     \__draw_path_reset_limits:
1236     \bool_set_true:N \l_draw_bb_update_bool
1237     \draw_transform_matrix_reset:
1238     \draw_transform_shift_reset:
1239     \__draw_softpath_clear:
1240     \draw_linewidth:n { \l_draw_default_linewidth_dim }
1241     \color_select:n { . }
1242     \draw_nonzero_rule:
1243     \draw_cap_but:
1244     \draw_join_miter:
1245     \draw_miterlimit:n { 10 }
1246     \draw_dash_pattern:nn { } { 0cm }
1247     \hbox_set:Nw \l__draw_layer_main_box
1248   }
1249 \cs_new_protected:Npn \draw_end:
1250 {
1251   \exp_args:NNNV \hbox_set_end:
1252   \clist_set:Nn \l_draw_layers_clist \l_draw_layers_clist
1253   \__draw_layers_insert:
1254   \__draw_backend_end:
1255   \hbox_set_end:
1256   \dim_compare:nNnT \g__draw_xmin_dim = \c_max_dim
1257   {
```

```

1258         \dim_gzero:N \g__draw_xmax_dim
1259         \dim_gzero:N \g__draw_xmin_dim
1260         \dim_gzero:N \g__draw_ymax_dim
1261         \dim_gzero:N \g__draw_ymin_dim
1262     }
1263     \hbox_set:Nn \l__draw_main_box
1264     {
1265         \skip_horizontal:n { -\g__draw_xmin_dim }
1266         \box_move_down:nn { \g__draw_ymin_dim }
1267         { \box_use_drop:N \l__draw_main_box }
1268     }
1269     \box_set_ht:Nn \l__draw_main_box
1270     { \g__draw_ymax_dim - \g__draw_ymin_dim }
1271     \box_set_dp:Nn \l__draw_main_box { Opt }
1272     \box_set_wd:Nn \l__draw_main_box
1273     { \g__draw_xmax_dim - \g__draw_xmin_dim }
1274     \mode_leave_vertical:
1275     \box_use_drop:N \l__draw_main_box
1276     \group_end:
1277 }

```

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

6.2 Scopes

```

\l__draw_linewidth_dim Storage for local variables.
\l__draw_fill_color_tl 1278 \dim_new:N \l__draw_linewidth_dim
\l__draw_stroke_color_tl 1279 \tl_new:N \l__draw_fill_color_tl
1280 \tl_new:N \l__draw_stroke_color_tl

```

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

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

```

\draw_scope_begin: 1281 \cs_new_protected:Npn \draw_scope_begin:
1282 {
1283     \__draw_backend_scope_begin:
1284     \group_begin:
1285         \dim_set_eq:NN \l__draw_linewidth_dim \g__draw_linewidth_dim
1286         \draw_path_scope_begin:
1287     }
1288 \cs_new_protected:Npn \draw_scope_end:
1289 {
1290     \draw_path_scope_end:
1291     \dim_gset_eq:NN \g__draw_linewidth_dim \l__draw_linewidth_dim
1292     \group_end:
1293     \__draw_backend_scope_end:
1294 }

```

(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 1295 \dim_new:N \l__draw_xmax_dim
\l__draw_ymax_dim 1296 \dim_new:N \l__draw_xmin_dim
\l__draw_ymin_dim 1297 \dim_new:N \l__draw_ymax_dim
1298 \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: 1299 \cs_new_protected:Npn \__draw_scope_bb_begin:
1300 {
1301   \group_begin:
1302     \dim_set_eq:NN \l__draw_xmax_dim \g__draw_xmax_dim
1303     \dim_set_eq:NN \l__draw_xmin_dim \g__draw_xmin_dim
1304     \dim_set_eq:NN \l__draw_ymax_dim \g__draw_ymax_dim
1305     \dim_set_eq:NN \l__draw_ymin_dim \g__draw_ymin_dim
1306     \__draw_reset_bb:
1307   }
1308 \cs_new_protected:Npn \__draw_scope_bb_end:
1309 {
1310   \dim_gset_eq:NN \g__draw_xmax_dim \l__draw_xmax_dim
1311   \dim_gset_eq:NN \g__draw_xmin_dim \l__draw_xmin_dim
1312   \dim_gset_eq:NN \g__draw_ymax_dim \l__draw_ymax_dim
1313   \dim_gset_eq:NN \g__draw_ymin_dim \l__draw_ymin_dim
1314   \group_end:
1315 }
```

(End definition for __draw_scope_bb_begin: and __draw_scope_bb_end:.)

\draw_suspend_begin: Suspend all parts of a drawing.

```
\draw_suspend_end: 1316 \cs_new_protected:Npn \draw_suspend_begin:
1317 {
1318   \__draw_scope_bb_begin:
1319   \draw_path_scope_begin:
1320   \draw_transform_matrix_reset:
1321   \draw_transform_shift_reset:
1322   \__draw_layers_save:
1323 }
1324 \cs_new_protected:Npn \draw_suspend_end:
1325 {
1326   \__draw_layers_restore:
1327   \draw_path_scope_end:
1328   \__draw_scope_bb_end:
1329 }
```

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

```
1330 </package>
```

7 I3draw-softpath implementation

```
1331 <*package>
```

```
1332 <@@=draw>
```

7.1 Managing soft paths

There are two linked aims in the code here. The most significant is to provide a way to modify paths, for example to shorten the ends or round the corners. This means that the path cannot be written piecemeal as specials, but rather needs to be held in macros. The

second aspect that follows from this is performance: simply adding to a single macro a piece at a time will have poor performance as the list gets long so we use `\tl_build_...` functions.

Each marker (operation) token takes two arguments, which makes processing more straight-forward. As such, some operations have dummy arguments, whilst others have to be split over several tokens. As the code here is at a low level, all dimension arguments are assumed to be explicit and fully-expanded.

```

\g__draw_softpath_main_tl The soft path itself.
1333 \tl_new:N \g__draw_softpath_main_tl
(End definition for \g__draw_softpath_main_tl.)

\l__draw_softpath_internal_tl The soft path itself.
1334 \tl_new:N \l__draw_softpath_internal_tl
(End definition for \l__draw_softpath_internal_tl.)

\g__draw_softpath_corners_bool Allow for optimised path use.
1335 \bool_new:N \g__draw_softpath_corners_bool
(End definition for \g__draw_softpath_corners_bool.)

\__draw_softpath_add:n
\__draw_softpath_add:o 1336 \cs_new_protected:Npn \__draw_softpath_add:n
\__draw_softpath_add:x 1337 { \tl_build_gput_right:Nn \g__draw_softpath_main_tl }
1338 \cs_generate_variant:Nn \__draw_softpath_add:n { o, x }
(End definition for \__draw_softpath_add:n.)

\__draw_softpath_use: Using and clearing is trivial.
\__draw_softpath_clear: 1339 \cs_new_protected:Npn \__draw_softpath_use:
1340 {
1341   \tl_build_get:NN \g__draw_softpath_main_tl \l__draw_softpath_internal_tl
1342   \l__draw_softpath_internal_tl
1343 }
1344 \cs_new_protected:Npn \__draw_softpath_clear:
1345 {
1346   \tl_build_gclear:N \g__draw_softpath_main_tl
1347   \bool_gset_false:N \g__draw_softpath_corners_bool
1348 }
(End definition for \__draw_softpath_use: and \__draw_softpath_clear:.)

\g__draw_softpath_lastx_dim For tracking the end of the path (to close it).
\g__draw_softpath_lasty_dim 1349 \dim_new:N \g__draw_softpath_lastx_dim
1350 \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.
1351 \bool_new:N \g__draw_softpath_move_bool
1352 \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        1353 \cs_new_protected:Npn \_draw_softpath_closepath:
\__draw_softpath_moveto:nn       1354 {
    \_draw_softpath_rectangle:nnnn 1355   \_draw_softpath_add:x
    \_draw_softpath_roundpoint:nn  1356   {
    \_draw_softpath_roundpoint:VV   1357     \_draw_softpath_close_op:nn
    \_draw_softpath_roundpoint:VV   1358     { \dim_use:N \g__draw_softpath_lastx_dim }
    \_draw_softpath_roundpoint:VV   1359     { \dim_use:N \g__draw_softpath_lasty_dim }
    \_draw_softpath_roundpoint:VV   1360   }
    \_draw_softpath_roundpoint:VV   1361 }
\cs_new_protected:Npn \_draw_softpath_curveto:nnnnnn #1#2#3#4#5#6
1362 {
1363   \_draw_softpath_add:n
1364   {
1365     \_draw_softpath_curveto_opi:nn {#1} {#2}
1366     \_draw_softpath_curveto_opii:nn {#3} {#4}
1367     \_draw_softpath_curveto_opiii:nn {#5} {#6}
1368   }
1369 }
1370 }
1371 \cs_new_protected:Npn \_draw_softpath_lineto:nn #1#2
1372 {
1373   \_draw_softpath_add:n
1374   { \_draw_softpath_lineto_op:nn {#1} {#2} }
1375 }
1376 \cs_new_protected:Npn \_draw_softpath_moveto:nn #1#2
1377 {
1378   \_draw_softpath_add:n
1379   { \_draw_softpath_moveto_op:nn {#1} {#2} }
1380   \bool_if:NT \g__draw_softpath_move_bool
1381   {
1382     \dim_gset:Nn \g__draw_softpath_lastx_dim {#1}
1383     \dim_gset:Nn \g__draw_softpath_lasty_dim {#2}
1384   }
1385 }
1386 \cs_new_protected:Npn \_draw_softpath_rectangle:nnnn #1#2#3#4
1387 {
1388   \_draw_softpath_add:n
1389   {
1390     \_draw_softpath_rectangle_opi:nn {#1} {#2}
1391     \_draw_softpath_rectangle_opii:nn {#3} {#4}
1392   }
1393 }
1394 \cs_new_protected:Npn \_draw_softpath_roundpoint:nn #1#2
1395 {
1396   \_draw_softpath_add:n
1397   { \_draw_softpath_roundpoint_op:nn {#1} {#2} }
1398   \bool_gset_true:N \g__draw_softpath_corners_bool
1399 }
1400 \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
  \_draw_softpath_curveto_opi:nn
  \_draw_softpath_curveto_opii:nn
  \_draw_softpath_curveto_opiii:nn
  \_draw_softpath_lineto_op:nn
  \_draw_softpath_moveto_op:nn
  \_draw_softpath_roundpoint_op:nn
  \_draw_softpath_rectangle_opi:nn
  \_draw_softpath_rectangle_opii:nn
\_draw_softpath_curveto_opi:nnNnnNnn
\_draw_softpath_rectangle_opi:nnNnn

```

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

```

1401 \cs_new_protected:Npn \__draw_softpath_close_op:nn #1#2
1402   { \__draw_backend_closepath: }
1403 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nn #1#2
1404   { \__draw_softpath_curveto_opi:nnNnnNnn {#1} {#2} }
1405 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nnNnnNnn #1#2#3#4#5#6#7#8
1406   { \__draw_backend_curveto:nnnnnn {#1} {#2} {#4} {#5} {#7} {#8} }
1407 \cs_new_protected:Npn \__draw_softpath_curveto_opii:nn #1#2
1408   { \__draw_softpath_curveto_opii:nn }
1409 \cs_new_protected:Npn \__draw_softpath_curveto_opiii:nn #1#2
1410   { \__draw_softpath_curveto_opiii:nn }
1411 \cs_new_protected:Npn \__draw_softpath_lineto_op:nn #1#2
1412   { \__draw_backend_lineto:nn {#1} {#2} }
1413 \cs_new_protected:Npn \__draw_softpath_moveto_op:nn #1#2
1414   { \__draw_backend_moveto:nn {#1} {#2} }
1415 \cs_new_protected:Npn \__draw_softpath_roundpoint_op:nn #1#2 { }
1416 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nn #1#2
1417   { \__draw_softpath_rectangle_opi:nnNnn {#1} {#2} }
1418 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nnNnn #1#2#3#4#5
1419   { \__draw_backend_rectangle:nnnn {#1} {#2} {#4} {#5} }
1420 \cs_new_protected:Npn \__draw_softpath_rectangle_opii:nn #1#2 { }

```

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

7.2 Rounding soft path corners

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

`\l__draw_softpath_main_tl` For constructing the updated path.

```
1421 \tl_new:N \l__draw_softpath_main_tl
```

(End definition for `\l__draw_softpath_main_tl`.)

`\l__draw_softpath_part_tl` Data structures.

```
1422 \tl_new:N \l__draw_softpath_part_tl
1423 \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
  \l__draw_softpath_corneri_dim
  \l__draw_softpath_cornerii_dim
\l__draw_softpath_first_tl
  \l__draw_softpath_move_tl
1424 \fp_new:N \l__draw_softpath_lastx_fp
1425 \fp_new:N \l__draw_softpath_lasty_fp
1426 \dim_new:N \l__draw_softpath_corneri_dim
1427 \dim_new:N \l__draw_softpath_cornerii_dim
1428 \tl_new:N \l__draw_softpath_first_tl
1429 \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.

```
1430 \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
\__draw_softpath_round_loop:Nnn such, we avoid this entirely if we know there are no corners to deal with. Assuming there
\__draw_softpath_round_action:nn is work to do, we recover the existing path and start a loop.
\__draw_softpath_round_action:Nnn 1431 \cs_new_protected:Npn \__draw_softpath_round_corners:
\__draw_softpath_round_action_curveto:NnnNnn 1432 {
\__draw_softpath_round_action_close: 1433   \bool_if:NT \g__draw_softpath_corners_bool
\__draw_softpath_round_lookahead:NnnNnn 1434   {
\__draw_softpath_round_roundpoint:NnnNnnNnn 1435     \group_begin:
\__draw_softpath_round_calc:NnnNnn 1436       \tl_clear:N \l__draw_softpath_main_tl
\__draw_softpath_round_calc:nnnnnn 1437       \tl_clear:N \l__draw_softpath_part_tl
\__draw_softpath_round_calc:fVnnnn 1438       \fp_zero:N \l__draw_softpath_lastx_fp
\__draw_softpath_round_calc:nnnnw 1439       \fp_zero:N \l__draw_softpath_lasty_fp
\__draw_softpath_round_close:nn 1440       \tl_clear:N \l__draw_softpath_first_tl
\__draw_softpath_round_close:w 1441       \tl_clear:N \l__draw_softpath_move_tl
\__draw_softpath_round_end: 1442       \tl_build_get:NN \g__draw_softpath_main_tl \l__draw_softpath_internal_tl
1443       \exp_after:wN \__draw_softpath_round_loop:Nnn
1444       \l__draw_softpath_internal_tl
1445       \q__draw_recursion_tail ? ?
1446       \q__draw_recursion_stop
1447     \group_end:
1448   }
1449   \bool_gset_false:N \g__draw_softpath_corners_bool
1450 }

```

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.

```

1451 \cs_new_protected:Npn \__draw_softpath_round_loop:Nnn #1#2#3
1452 {
1453   \__draw_if_recursion_tail_stop_do:Nn #1 { \__draw_softpath_round_end: }
1454   \token_if_eq_meaning:NNTF #1 \__draw_softpath_roundpoint_op:nn
1455   { \__draw_softpath_round_action:nn {#2} {#3} }
1456   {
1457     \tl_if_empty:NT \l__draw_softpath_first_tl
1458     { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1459     \fp_set:Nn \l__draw_softpath_lastx_fp {#2}
1460     \fp_set:Nn \l__draw_softpath_lasty_fp {#3}
1461     \token_if_eq_meaning:NNTF #1 \__draw_softpath_moveto_op:nn
1462     {
1463       \tl_put_right:No \l__draw_softpath_main_tl
1464       \l__draw_softpath_move_tl
1465       \tl_put_right:No \l__draw_softpath_main_tl
1466       \l__draw_softpath_part_tl
1467       \tl_set:Nn \l__draw_softpath_move_tl { #1 {#2} {#3} }
1468       \tl_clear:N \l__draw_softpath_first_tl
1469       \tl_clear:N \l__draw_softpath_part_tl
1470     }
1471     { \tl_put_right:Nn \l__draw_softpath_part_tl { #1 {#2} {#3} } }
1472     \__draw_softpath_round_loop:Nnn
1473   }

```



```

1474 }
1475 \cs_new_protected:Npn \__draw_softpath_round_action:nn #1#2
1476 {
1477   \dim_set:Nn \l__draw_softpath_corneri_dim {#1}
1478   \dim_set:Nn \l__draw_softpath_cornerii_dim {#2}
1479   \bool_lazy_and:nnTF
1480     { \dim_compare_p:nNn \l__draw_softpath_corneri_dim = { Opt } }
1481     { \dim_compare_p:nNn \l__draw_softpath_cornerii_dim = { Opt } }
1482     { \__draw_softpath_round_loop:Nnn }
1483     { \__draw_softpath_round_action:Nnn }
1484 }

```

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.

```

1485 \cs_new_protected:Npn \__draw_softpath_round_action:Nnn #1#2#3
1486 {
1487   \tl_if_empty:NT \l__draw_softpath_first_tl
1488     { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1489   \token_if_eq_meaning:NNTF #1 \__draw_softpath_curveto_opi:nn
1490     { \__draw_softpath_round_action_curveto:NnnNnn }
1491     {
1492       \token_if_eq_meaning:NNTF #1 \__draw_softpath_close_op:nn
1493         { \__draw_softpath_round_action_close: }
1494         {
1495           \token_if_eq_meaning:NNTF #1 \__draw_softpath_lineto_op:nn
1496             { \__draw_softpath_round_lookahead:NnnNnn }
1497             { \__draw_softpath_round_loop:Nnn }
1498         }
1499     }
1500   #1 {#2} {#3}
1501 }

```

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.

```

1502 \cs_new_protected:Npn \__draw_softpath_round_action_curveto:NnnNnn
1503   #1#2#3#4#5#6
1504 {
1505   \tl_put_right:Nn \l__draw_softpath_part_tl
1506     { #1 {#2} {#3} #4 {#5} {#6} }
1507   \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1508   \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1509   \__draw_softpath_round_lookahead:NnnNnn
1510 }
1511 \cs_new_protected:Npn \__draw_softpath_round_action_close:
1512 {
1513   \bool_lazy_and:nnTF
1514     { ! \tl_if_empty_p:N \l__draw_softpath_first_tl }
1515     { ! \tl_if_empty_p:N \l__draw_softpath_move_tl }
1516     {
1517       \exp_after:wN \__draw_softpath_round_close:nn
1518         \l__draw_softpath_first_tl
1519     }
1520   { \__draw_softpath_round_loop:Nnn }
1521 }

```

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.

```

1522 \cs_new_protected:Npn \__draw_softpath_round_lookahead:NnnNnn #1#2#3#4#5#6
1523 {
1524   \bool_lazy_any:nTF
1525     {
1526       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_lineto_op:nn }
1527       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_curveto_opi:nn }
1528       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_close_op:nn }
1529     }
1530     {
1531       \__draw_softpath_round_calc:NnnNnn
1532       \__draw_softpath_round_loop:Nnn
1533       {#5} {#6}
1534     }
1535     {
1536       \token_if_eq_meaning:NNTF #4 \__draw_softpath_roundpoint_op:nn
1537       { \__draw_softpath_round_roundpoint:NnnNnnNnn }
1538       { \__draw_softpath_round_loop:Nnn }
1539     }
1540     #1 {#2} {#3}
1541     #4 {#5} {#6}
1542   }
1543 \cs_new_protected:Npn \__draw_softpath_round_roundpoint:NnnNnnNnn
1544   #1#2#3#4#5#6#7#8#9
1545   {
1546     \__draw_softpath_round_calc:NnnNnn
1547     \__draw_softpath_round_loop:Nnn
1548     {#8} {#9}
1549     #1 {#2} {#3}
1550     #4 {#5} {#6} #7 {#8} {#9}
1551   }

```

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

```

1552 \cs_new_protected:Npn \__draw_softpath_round_calc:NnnNnn #1#2#3#4#5#6
1553 {
1554   \tl_set:Nx \l__draw_softpath_curve_end_tl
1555     {
1556       \draw_point_interpolate_distance:nnn
1557       \l__draw_softpath_cornerii_dim
1558       { #5 , #6 } { #2 , #3 }
1559     }
1560   \tl_put_right:Nx \l__draw_softpath_part_tl
1561     {
1562       \exp_not:N #4

```

```

1563     \__draw_softpath_round_calc:fVnnnn
1564     {
1565         \draw_point_interpolate_distance:nnn
1566         \l__draw_softpath_corneri_dim
1567         { #5 , #6 }
1568         {
1569             \l__draw_softpath_lastx_fp ,
1570             \l__draw_softpath_lasty_fp
1571         }
1572     }
1573     \l__draw_softpath_curve_end_t1
1574     {#5} {#6} {#2} {#3}
1575 }
1576 \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1577 \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1578 #1
1579 }

```

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).

```

1580 \cs_new:Npn \__draw_softpath_round_calc:nnnnn #1#2#3#4#5#6
1581 {
1582     \__draw_softpath_round_calc:nnnw {#3} {#4} {#5} {#6}
1583     #1 \s__draw_mark #2 \s__draw_stop
1584 }
1585 \cs_generate_variant:Nn \__draw_softpath_round_calc:nnnnn { fV }

```

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

```

1586 \cs_new:Npn \__draw_softpath_round_calc:nnnw
1587 #1#2#3#4 #5 , #6 \s__draw_mark #7 , #8 \s__draw_stop
1588 {
1589     {#5} {#6}
1590     \exp_not:N \__draw_softpath_curveto_opi:nn
1591     {
1592         \fp_to_dim:n
1593         { #5 + \c__draw_softpath_arc_fp * ( #1 - #5 ) }
1594     }
1595     {
1596         \fp_to_dim:n
1597         { #6 + \c__draw_softpath_arc_fp * ( #2 - #6 ) }
1598     }
1599     \exp_not:N \__draw_softpath_curveto_opii:nn
1600     {
1601         \fp_to_dim:n
1602         { #7 + \c__draw_softpath_arc_fp * ( #1 - #7 ) }
1603     }
1604     {
1605         \fp_to_dim:n
1606         { #8 + \c__draw_softpath_arc_fp * ( #2 - #8 ) }
1607     }
1608     \exp_not:N \__draw_softpath_curveto_opiii:nn
1609     {#7} {#8}
1610 }

```

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.

```

1611 \cs_new_protected:Npn \__draw_softpath_round_close:nn #1#2
1612 {
1613   \use:x
1614   {
1615     \__draw_softpath_round_calc:NnnNnn
1616     {
1617       \tl_set:Nx \exp_not:N \l__draw_softpath_move_tl
1618       {
1619         \__draw_softpath_moveto_op:nn
1620         \exp_not:N \exp_after:wN
1621         \exp_not:N \__draw_softpath_round_close:w
1622         \exp_not:N \l__draw_softpath_curve_end_tl
1623         \s__draw_stop
1624       }
1625       \use:x
1626       {
1627         \exp_not:N \exp_not:N \exp_not:N \use_i:nnnn
1628         {
1629           \__draw_softpath_round_loop:Nnn
1630           \__draw_softpath_close_op:nn
1631           \exp_not:N \exp_after:wN
1632           \exp_not:N \__draw_softpath_round_close:w
1633           \exp_not:N \l__draw_softpath_curve_end_tl
1634           \s__draw_stop
1635         }
1636       }
1637     }
1638     {#1} {#2}
1639     \__draw_softpath_lineto_op:nn
1640     \exp_after:wN \use_none:n \l__draw_softpath_move_tl
1641   }
1642 }
1643 \cs_new:Npn \__draw_softpath_round_close:w #1 , #2 \s__draw_stop { {#1} {#2} }

```

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

```

1644 \cs_new_protected:Npn \__draw_softpath_round_end:
1645 {
1646   \tl_put_right:No \l__draw_softpath_main_tl
1647   \l__draw_softpath_move_tl
1648   \tl_put_right:No \l__draw_softpath_main_tl
1649   \l__draw_softpath_part_tl
1650   \tl_build_gclear:N \g__draw_softpath_main_tl
1651   \__draw_softpath_add:o \l__draw_softpath_main_tl
1652 }

```

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

```

1653 </package>

```

8 I3draw-state implementation

1654 `*package)`

1655 `\@@=draw)`

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

- `\pgfsetinnerlinewidth`, `\pgfinnerlinewidth`, `\pgfsetinnerstrokecolor`, `\pgfsetinnerstrokecolor`

Likely to be added on further work is done on paths/stroking.

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

1656 `\dim_new:N \g__draw_linewidth_dim`

(End definition for \g__draw_linewidth_dim.)

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

1657 `\dim_new:N \l_draw_default_linewidth_dim`

1658 `\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.

1659 `\cs_new_protected:Npn \draw_linewidth:n #1`

1660 `{`

1661 `\dim_gset:Nn \g__draw_linewidth_dim { \fp_to_dim:n {#1} }`

1662 `__draw_backend_linewidth:n \g__draw_linewidth_dim`

1663 `}`

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

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

`\l__draw_tmp_seq` 1664 `\cs_new_protected:Npn \draw_dash_pattern:nn #1#2`

1665 `{`

1666 `\group_begin:`

1667 `\seq_set_from_clist:Nn \l__draw_tmp_seq {#1}`

1668 `\seq_set_map:NNn \l__draw_tmp_seq \l__draw_tmp_seq`

1669 `{ \fp_to_dim:n {##1} }`

1670 `\use:x`

1671 `{`

1672 `__draw_backend_dash_pattern:nn`

1673 `{ \seq_use:Nn \l__draw_tmp_seq { , } }`

1674 `{ \fp_to_dim:n {#2} }`

1675 `}`

1676 `\group_end:`

1677 `}`

1678 `\seq_new:N \l__draw_tmp_seq`

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

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

1679 `\cs_new_protected:Npn \draw_miterlimit:n #1`

1680 `{ \exp_args:Nx __draw_backend_miterlimit:n { \fp_eval:n {#1} } }`

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

```
\draw_cap_but: All straight wrappers.
\draw_cap_rectangle: 1681 \cs_new_protected:Npn \draw_cap_but: { \__draw_backend_cap_but: }
\draw_cap_round: 1682 \cs_new_protected:Npn \draw_cap_rectangle: { \__draw_backend_cap_rectangle: }
\draw_evenodd_rule: 1683 \cs_new_protected:Npn \draw_cap_round: { \__draw_backend_cap_round: }
\draw_nonzero_rule: 1684 \cs_new_protected:Npn \draw_evenodd_rule: { \__draw_backend_evenodd_rule: }
\draw_join_bevel: 1685 \cs_new_protected:Npn \draw_nonzero_rule: { \__draw_backend_nonzero_rule: }
\draw_join_miter: 1686 \cs_new_protected:Npn \draw_join_bevel: { \__draw_backend_join_bevel: }
\draw_join_round: 1687 \cs_new_protected:Npn \draw_join_miter: { \__draw_backend_join_miter: }
1688 \cs_new_protected:Npn \draw_join_round: { \__draw_backend_join_round: }
```

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

```
1689 </package>
```

9 I3draw-transforms implementation

```
1690 <*package>
```

```
1691 <@@=draw>
```

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

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

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

```
1692 \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 1693 \fp_new:N \l__draw_matrix_a_fp
\l__draw_matrix_c_fp 1694 \fp_new:N \l__draw_matrix_b_fp
\l__draw_xshift_dim 1695 \fp_new:N \l__draw_matrix_c_fp
\l__draw_yshift_dim 1696 \fp_new:N \l__draw_matrix_d_fp
1697 \dim_new:N \l__draw_xshift_dim
1698 \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: 1699 \cs_new_protected:Npn \draw_transform_matrix_reset:
1700 {
1701   \fp_set:Nn \l__draw_matrix_a_fp { 1 }
1702   \fp_zero:N \l__draw_matrix_b_fp
1703   \fp_zero:N \l__draw_matrix_c_fp
1704   \fp_set:Nn \l__draw_matrix_d_fp { 1 }
1705 }
1706 \cs_new_protected:Npn \draw_transform_shift_reset:
1707 {
1708   \dim_zero:N \l__draw_xshift_dim
1709   \dim_zero:N \l__draw_yshift_dim
1710 }
1711 \draw_transform_matrix_reset:
1712 \draw_transform_shift_reset:
```

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

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

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

```
\_draw_transform_shift_absolute:nn 1713 \cs_new_protected:Npn \draw_transform_matrix_absolute:nmmn #1#2#3#4
1714 {
1715   \fp_set:Nn \l__draw_matrix_a_fp {#1}
1716   \fp_set:Nn \l__draw_matrix_b_fp {#2}
1717   \fp_set:Nn \l__draw_matrix_c_fp {#3}
1718   \fp_set:Nn \l__draw_matrix_d_fp {#4}
1719   \bool_lazy_all:nTF
1720   {
1721     { \fp_compare_p:nNn \l__draw_matrix_a_fp = \c_one_fp }
1722     { \fp_compare_p:nNn \l__draw_matrix_b_fp = \c_zero_fp }
1723     { \fp_compare_p:nNn \l__draw_matrix_c_fp = \c_zero_fp }
1724     { \fp_compare_p:nNn \l__draw_matrix_d_fp = \c_one_fp }
1725   }
1726   { \bool_set_false:N \l__draw_matrix_active_bool }
1727   { \bool_set_true:N \l__draw_matrix_active_bool }
1728 }
1729 \cs_new_protected:Npn \draw_transform_shift_absolute:n #1
1730 {
1731   \__draw_point_process:nn
1732   { \__draw_transform_shift_absolute:nn } {#1}
1733 }
1734 \cs_new_protected:Npn \__draw_transform_shift_absolute:nn #1#2
1735 {
1736   \dim_set:Nn \l__draw_xshift_dim {#1}
1737   \dim_set:Nn \l__draw_yshift_dim {#2}
1738 }
```

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

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

`__draw_transform:nmmn`

`\draw_transform_shift:n`

```
1739 \cs_new_protected:Npn \draw_transform_matrix:nmmn #1#2#3#4
```

`__draw_transform_shift:nn`

```

1740 {
1741   \use:x
1742   {
1743     \__draw_transform:nmmm
1744     { \fp_eval:n {#1} }
1745     { \fp_eval:n {#2} }
1746     { \fp_eval:n {#3} }
1747     { \fp_eval:n {#4} }
1748   }
1749 }
1750 \cs_new_protected:Npn \__draw_transform:nmmm #1#2#3#4
1751 {
1752   \use:x
1753   {
1754     \draw_transform_matrix_absolute:nmmm
1755     { #1 * \l__draw_matrix_a_fp + #2 * \l__draw_matrix_c_fp }
1756     { #1 * \l__draw_matrix_b_fp + #2 * \l__draw_matrix_d_fp }
1757     { #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_c_fp }
1758     { #3 * \l__draw_matrix_b_fp + #4 * \l__draw_matrix_d_fp }
1759   }
1760 }
1761 \cs_new_protected:Npn \draw_transform_shift:n #1
1762 {
1763   \__draw_point_process:nn
1764   { \__draw_transform_shift:nn } {#1}
1765 }
1766 \cs_new_protected:Npn \__draw_transform_shift:nn #1#2
1767 {
1768   \dim_set:Nn \l__draw_xshift_dim { \l__draw_xshift_dim + #1 }
1769   \dim_set:Nn \l__draw_yshift_dim { \l__draw_yshift_dim + #2 }
1770 }

```

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

```

\draw_transform_matrix_invert: Standard mathematics: calculate the inverse matrix and use that, then undo the shifts.
\__draw_transform_invert:n 1771 \cs_new_protected:Npn \draw_transform_matrix_invert:
\__draw_transform_invert:f 1772 {
\draw_transform_shift_invert: 1773   \bool_if:NT \l__draw_matrix_active_bool
1774   {
1775     \__draw_transform_invert:f
1776     {
1777       \fp_eval:n
1778       {
1779         1 /
1780         (
1781           \l__draw_matrix_a_fp * \l__draw_matrix_d_fp
1782           - \l__draw_matrix_b_fp * \l__draw_matrix_c_fp
1783         )
1784       }
1785     }
1786   }
1787 }
1788 \cs_new_protected:Npn \__draw_transform_invert:n #1

```



```

1789 {
1790   \fp_set:Nn \l__draw_matrix_a_fp
1791     { \l__draw_matrix_d_fp * #1 }
1792   \fp_set:Nn \l__draw_matrix_b_fp
1793     { -\l__draw_matrix_b_fp * #1 }
1794   \fp_set:Nn \l__draw_matrix_c_fp
1795     { -\l__draw_matrix_c_fp * #1 }
1796   \fp_set:Nn \l__draw_matrix_d_fp
1797     { \l__draw_matrix_a_fp * #1 }
1798 }
1799 \cs_generate_variant:Nn \__draw_transform_invert:n { f }
1800 \cs_new_protected:Npn \draw_transform_shift_invert:
1801 {
1802   \dim_set:Nn \l__draw_xshift_dim { -\l__draw_xshift_dim }
1803   \dim_set:Nn \l__draw_yshift_dim { -\l__draw_yshift_dim }
1804 }

```

(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.

```

1805 \cs_new_protected:Npn \draw_transform_triangle:nnn #1#2#3
1806 {
1807   \__draw_point_process:nnn
1808   {
1809     \__draw_point_process:nn
1810     { \__draw_tranform_triangle:nnnnnn }
1811     {#1}
1812   }
1813   {#2} {#3}
1814 }
1815 \cs_new_protected:Npn \__draw_tranform_triangle:nnnnnn #1#2#3#4#5#6
1816 {
1817   \use:x
1818   {
1819     \draw_transform_matrix_absolute:nnnn
1820     { #3 - #1 }
1821     { #4 - #2 }
1822     { #5 - #1 }
1823     { #6 - #2 }
1824     \draw_transform_shift_absolute:n { #1 , #2 }
1825   }
1826 }

```

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

`\draw_transform_scale:n` Lots of shortcuts.

```

\draw_transform_xscale:n 1827 \cs_new_protected:Npn \draw_transform_scale:n #1
\draw_transform_yscale:n 1828 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { #1 } }
\draw_transform_xshift:n 1829 \cs_new_protected:Npn \draw_transform_xscale:n #1
\draw_transform_yshift:n 1830 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { #1 } }
\draw_transform_xslant:n 1831 \cs_new_protected:Npn \draw_transform_yscale:n #1
\draw_transform_yslant:n 1832 { \draw_transform_matrix:nnnn { 1 } { 0 } { 0 } { #1 } }
1833 \cs_new_protected:Npn \draw_transform_xshift:n #1
1834 { \draw_transform_shift:n { #1 , Opt } }

```

```

1835 \cs_new_protected:Npn \draw_transform_yshift:n #1
1836   { \draw_transform_shift:n { Opt , #1 } }
1837 \cs_new_protected:Npn \draw_transform_xslant:n #1
1838   { \draw_transform_matrix:nmmm { 1 } { 0 } { #1 } { 1 } }
1839 \cs_new_protected:Npn \draw_transform_yslant:n #1
1840   { \draw_transform_matrix:nmmm { 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 1841 \cs_new_protected:Npn \draw_transform_rotate:n #1
\__draw_transform_rotate:f 1842   { \__draw_transform_rotate:f { \fp_eval:n {#1} } }
\__draw_transform_rotate:nn 1843 \cs_new_protected:Npn \__draw_transform_rotate:n #1
\__draw_transform_rotate:ff 1844   {
1845     \__draw_transform_rotate:ff
1846     { \fp_eval:n { cosd(#1) } }
1847     { \fp_eval:n { sind(#1) } }
1848   }
1849 \cs_generate_variant:Nn \__draw_transform_rotate:n { f }
1850 \cs_new_protected:Npn \__draw_transform_rotate:nn #1#2
1851   { \draw_transform_matrix:nmmm {#1} {#2} { -#2 } { #1 } }
1852 \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 ??.)

```

1853 </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> . . .	<u>171</u> , <u>786</u> , <u>1038</u> , <u>1042</u> , <u>1066</u> , <u>1069</u>
bool commands:	
<code>\bool_gset_eq:NN</code>	<u>767</u>
<code>\bool_gset_false:N</code>	<u>1347</u> , <u>1449</u>
<code>\bool_gset_true:N</code>	<u>1352</u> , <u>1398</u>
<code>\bool_if:NTF</code> <u>21</u> , <u>115</u> , <u>194</u> , <u>233</u> , <u>682</u> , <u>698</u> , <u>699</u> , <u>703</u> , <u>1161</u> , <u>1193</u> , <u>1380</u> , <u>1433</u> , <u>1773</u>
<code>\bool_lazy_all:nTF</code>	<u>1719</u>
<code>\bool_lazy_and:nnTF</code> <u>225</u> , <u>677</u> , <u>1479</u> , <u>1513</u>
<code>\bool_lazy_any:nTF</code>	<u>1524</u>
<code>\bool_lazy_or:nnTF</code>	<u>558</u> , <u>653</u> , <u>686</u> , <u>691</u>
<code>\bool_new:N</code>	<u>86</u> , <u>220</u> , <u>641</u> , <u>642</u> , <u>643</u> , <u>644</u> , <u>645</u> , <u>745</u> , <u>1217</u> , <u>1335</u> , <u>1351</u> , <u>1692</u>
<code>\bool_set_eq:NN</code>	<u>759</u>
<code>\bool_set_false:N</code> <u>96</u> , <u>228</u> , <u>664</u> , <u>665</u> , <u>666</u> , <u>685</u> , <u>1726</u>
<code>\bool_set_true:N</code>	<u>98</u> , <u>229</u> , <u>670</u> , <u>708</u> , <u>712</u> , <u>713</u> , <u>1236</u> , <u>1727</u>
box commands:	
<code>\box_dp:N</code>	<u>17</u> , <u>67</u>
<code>\box_gset_eq:NN</code>	<u>156</u>
<code>\box_gset_wd:Nn</code>	<u>100</u> , <u>133</u>
<code>\box_ht:N</code>	<u>17</u> , <u>69</u>
<code>\box_if_exist:NTF</code>	<u>93</u>
<code>\box_move_down:nn</code>	<u>1266</u>
<code>\box_move_up:nn</code>	<u>51</u>
<code>\box_new:N</code>	<u>13</u> , <u>80</u> , <u>81</u> , <u>1218</u> , <u>1219</u>
<code>\box_set_dp:Nn</code>	<u>55</u> , <u>1271</u>
<code>\box_set_eq:NN</code>	<u>145</u>
<code>\box_set_ht:Nn</code>	<u>54</u> , <u>1269</u>
<code>\box_set_wd:Nn</code>	<u>56</u> , <u>128</u> , <u>1272</u>
<code>\box_use_drop:N</code> <u>52</u> , <u>57</u> , <u>102</u> , <u>129</u> , <u>134</u> , <u>1267</u> , <u>1275</u>
<code>\box_wd:N</code>	<u>17</u> , <u>66</u> , <u>68</u>
C	
clist commands:	
<code>\clist_map_inline:Nn</code>	<u>124</u> , <u>141</u> , <u>152</u>
<code>\clist_map_inline:nn</code>	<u>667</u>
<code>\clist_new:N</code>	<u>87</u> , <u>89</u>
<code>\clist_set:Nn</code>	<u>88</u> , <u>1252</u>
coffin commands:	
<code>\coffin_typeset:Nnnnn</code>	<u>64</u>
<code>\coffin_wd:N</code>	<u>66</u>
color commands:	
<code>\color_select:n</code>	<u>1241</u>
cs commands:	
<code>\cs_generate_variant:Nn</code> <u>420</u> , <u>607</u> , <u>640</u> , <u>845</u> , <u>853</u> , <u>895</u> , <u>914</u> , <u>921</u> , <u>929</u> , <u>936</u> , <u>945</u> , <u>951</u> , <u>963</u> , <u>966</u> , <u>984</u> , <u>1002</u> , <u>1010</u> , <u>1016</u> , <u>1037</u> , <u>1051</u> , <u>1065</u> , <u>1086</u> , <u>1121</u> , <u>1140</u> , <u>1153</u> , <u>1338</u> , <u>1400</u> , <u>1585</u> , <u>1799</u> , <u>1849</u> , <u>1852</u>
<code>\cs_if_exist:NTF</code>	<u>669</u>
<code>\cs_if_exist_use:NTF</code>	<u>403</u> , <u>412</u> , <u>672</u>
<code>\cs_new:Npn</code>	<u>511</u> , <u>521</u> , <u>531</u> , <u>541</u> , <u>790</u> , <u>796</u> , <u>798</u> , <u>800</u> , <u>807</u> , <u>809</u> , <u>811</u> , <u>819</u> , <u>821</u> , <u>824</u> , <u>833</u> , <u>838</u> , <u>841</u> , <u>843</u> , <u>846</u> , <u>847</u> , <u>849</u> , <u>851</u> , <u>854</u> , <u>856</u> , <u>862</u> , <u>871</u> , <u>877</u> , <u>887</u> , <u>896</u> , <u>902</u> , <u>907</u> , <u>915</u> , <u>922</u> , <u>930</u> , <u>937</u> , <u>946</u> , <u>952</u> , <u>958</u> , <u>964</u> , <u>967</u> , <u>973</u> , <u>982</u> , <u>985</u> , <u>991</u> , <u>996</u> , <u>1003</u> , <u>1011</u> , <u>1017</u> , <u>1023</u> , <u>1029</u> , <u>1043</u> , <u>1052</u> , <u>1058</u> , <u>1070</u> , <u>1072</u> , <u>1081</u> , <u>1111</u> , <u>1113</u> , <u>1122</u> , <u>1127</u> , <u>1141</u> , <u>1143</u> , <u>1145</u> , <u>1154</u> , <u>1159</u> , <u>1186</u> , <u>1191</u> , <u>1580</u> , <u>1586</u> , <u>1643</u>
<code>\cs_new_protected:Npn</code> <u>14</u> , <u>19</u> , <u>60</u> , <u>75</u> , <u>90</u> , <u>113</u> , <u>122</u> , <u>139</u> , <u>150</u> , <u>184</u> , <u>206</u> , <u>213</u> , <u>221</u> , <u>231</u> , <u>240</u> , <u>246</u> , <u>252</u> , <u>258</u> , <u>265</u> , <u>276</u> , <u>284</u> , <u>289</u> , <u>291</u> , <u>293</u> , <u>302</u> , <u>309</u> , <u>345</u> , <u>347</u> , <u>358</u> , <u>364</u> , <u>394</u> , <u>421</u> , <u>450</u> , <u>456</u> , <u>462</u> , <u>467</u> , <u>475</u> , <u>484</u> , <u>489</u> , <u>497</u> , <u>552</u> , <u>554</u> , <u>567</u> , <u>574</u> , <u>583</u> , <u>589</u> , <u>591</u> , <u>601</u> , <u>608</u> , <u>614</u> , <u>621</u> , <u>646</u> , <u>651</u> , <u>662</u> , <u>706</u> , <u>710</u> , <u>715</u> , <u>722</u> , <u>746</u> , <u>764</u> , <u>1093</u> , <u>1095</u> , <u>1097</u> , <u>1099</u> , <u>1103</u> , <u>1221</u> , <u>1228</u> , <u>1249</u> , <u>1281</u> , <u>1288</u> , <u>1299</u> , <u>1308</u> , <u>1316</u> , <u>1324</u> , <u>1336</u> , <u>1339</u> , <u>1344</u> , <u>1353</u> , <u>1362</u> , <u>1371</u> , <u>1376</u> , <u>1386</u> , <u>1394</u> , <u>1401</u> , <u>1403</u> , <u>1405</u> , <u>1407</u> , <u>1409</u> , <u>1411</u> , <u>1413</u> , <u>1415</u> , <u>1416</u> , <u>1418</u> , <u>1420</u> , <u>1431</u> , <u>1451</u> , <u>1475</u> , <u>1485</u> , <u>1502</u> , <u>1511</u> , <u>1522</u> , <u>1543</u> , <u>1552</u> , <u>1611</u> , <u>1644</u> , <u>1659</u> , <u>1664</u> , <u>1679</u> , <u>1681</u> , <u>1682</u> , <u>1683</u> , <u>1684</u> , <u>1685</u> , <u>1686</u> , <u>1687</u> , <u>1688</u> , <u>1699</u> , <u>1706</u> , <u>1713</u> , <u>1729</u> , <u>1734</u> , <u>1739</u> , <u>1750</u> , <u>1761</u> , <u>1766</u> , <u>1771</u> , <u>1788</u> , <u>1800</u> , <u>1805</u> , <u>1815</u> , <u>1827</u> , <u>1829</u> , <u>1831</u> , <u>1833</u> , <u>1835</u> , <u>1837</u> , <u>1839</u> , <u>1841</u> , <u>1843</u> , <u>1850</u>

D

dim commands:

`\dim_abs:n` 596, 597
`\dim_compare:nNnTF` 603, 610, 724, 1256
`\dim_compare_p:nNn` 226, 227, 1480, 1481
`\dim_eval:n` 596, 597
`\dim_gset:Nn` 186, 188,
 190, 192, 196, 198, 200, 202, 208,
 209, 210, 211, 215, 216, 726, 1223,
 1224, 1225, 1226, 1382, 1383, 1661
`\dim_gset_eq:NN`
 771, 772, 773, 774, 775, 776,
 777, 778, 1291, 1310, 1311, 1312, 1313
`\dim_gzero:N` .. 1258, 1259, 1260, 1261
`\dim_max:nn` 187, 191, 197, 201
`\dim_min:nn` 189, 193, 199, 203
`\dim_new:N` 178,
 179, 180, 181, 182, 183, 218, 219,
 737, 738, 739, 740, 741, 742, 743,
 744, 1087, 1088, 1089, 1090, 1091,
 1092, 1213, 1214, 1215, 1216, 1278,
 1295, 1296, 1297, 1298, 1349, 1350,
 1426, 1427, 1656, 1657, 1697, 1698
`\dim_set:Nn` 223,
 224, 1105, 1106, 1477, 1478, 1658,
 1736, 1737, 1768, 1769, 1802, 1803
`\dim_set_eq:NN`
 749, 750, 751, 752, 753, 754,
 755, 756, 1285, 1302, 1303, 1304, 1305
`\dim_step_inline:nnnn` 623, 631
`\dim_use:N` .. 724, 729, 731, 1358, 1359
`\dim_zero:N` 1708, 1709
`\c_max_dim` 208, 209, 210,
 211, 724, 1223, 1224, 1225, 1226, 1256

draw commands:

`\l_draw_bb_update_bool`
 21, 194, 678, 685, 1217, 1236
`\draw_begin:` 1228
`\draw_box_use:N` 14
`\draw_cap_but:` 1243, 1681
`\draw_cap_rectangle:` 1681
`\draw_cap_round:` 1681
`\draw_coffin_use:Nnn` 60
`\draw_dash_pattern:nn` ... 1246, 1664
`\l_draw_default_linewidth_dim` ...
 105, 1240, 1657
`\draw_end:` 1228
`\draw_evenodd_rule:` 1681
`\draw_join_bevel:` 1681
`\draw_join_miter:` 1244, 1681
`\draw_join_round:` 1681
`\draw_layer_begin:n` 90
`\draw_layer_end:` 90
`\draw_layer_new:n` 75

`\l_draw_layers_clist`
 87, 124, 141, 152, 1252
`\draw_linewidth:n` 105, 1240, 1659
`\draw_miterlimit:n` 1245, 1679
`\draw_nonzero_rule:` 1242, 1681
`\draw_path_arc:nnn` 345, 487
`\draw_path_arc:nnnn` 345
`\draw_path_arc_axes:nnnn` 484
`\draw_path_canvas_curveto:nnn` .. 289
`\draw_path_canvas_lineto:n` 289
`\draw_path_canvas_moveto:n` 289
`\draw_path_circle:nn` 552
`\draw_path_close:` 284, 580
`\draw_path_corner_arc:nn` 221
`\draw_path_curveto:nn` 302
`\draw_path_curveto:nnn` 240
`\draw_path_ellipse:nnn` 489, 553
`\draw_path_grid:nnnn` 591
`\draw_path_lineto:n`
 240, 577, 578, 579, 629, 637
`\draw_path_moveto:n`
 240, 576, 581, 628, 636
`\draw_path_rectangle:nn` 554, 590
`\draw_path_rectangle_corners:nn` 583
`\draw_path_scope_begin:`
 746, 1286, 1319
`\draw_path_scope_end:` 746, 1290, 1327
`\draw_path_use:n` 646
`\draw_path_use_clear:n` 646
`\draw_point_interpolate_arcaxes:nnnnnn`
 985
`\draw_point_interpolate_curve:nnnnn`
 1017
`\draw_point_interpolate_curve:nnnnnn`
 1017
`\draw_point_interpolate_curve_`
`auxi:nnnnnnnn` 1017
`\draw_point_interpolate_curve_`
`auxii:nnnnnnnn` 1017
`\draw_point_interpolate_curve_`
`auxiii:nnnnnn` 1017
`\draw_point_interpolate_curve_`
`auxiv:nnnnnn` 1017
`\draw_point_interpolate_curve_`
`auxv:nnw` 1017
`\draw_point_interpolate_curve_`
`auxvi:n` 1017
`\draw_point_interpolate_curve_`
`auxvii:nnnnnnnn` 1017
`\draw_point_interpolate_curve_`
`auxviii:nnnnnn` 1017
`\draw_point_interpolate_distance:nnn`
 967, 1556, 1565
`\draw_point_interpolate_line:mnn` 952

<code>\draw_point_intersect_circles:nmnn</code>	896	<code>_draw_backend_dash_pattern:nm</code>	1672
<code>\draw_point_intersect_lines:nmnn</code>	871	<code>_draw_backend_discardpath:</code>	689
<code>\draw_point_polar:nn</code>	847	<code>_draw_backend_end:</code>	1254
<code>\draw_point_polar:nmn</code>	428, 434, 438, 444, 847	<code>_draw_backend_evenodd_rule:</code>	1684
<code>\draw_point_transform:n</code>	25, 28, 31, 34, 244, 256, 272, 273, 274, 306, 307, 433, 437, 493, 564, 1154	<code>_draw_backend_join_bevel:</code>	1686
<code>\draw_point_unit_vector:n</code>	854, 980	<code>_draw_backend_join_miter:</code>	1687
<code>\draw_point_vec:nn</code>	1111	<code>_draw_backend_join_round:</code>	1688
<code>\draw_point_vec:nmn</code>	1111	<code>_draw_backend_lineto:nn</code>	1412
<code>\draw_point_vec_polar:nn</code>	1141	<code>_draw_backend_linewidth:n</code>	1662
<code>\draw_point_vec_polar:nmn</code>	1141	<code>_draw_backend_miterlimit:n</code>	1680
<code>\draw_scope_begin:</code>	1281	<code>_draw_backend_moveto:nn</code>	1414
<code>\draw_scope_end:</code>	1288	<code>_draw_backend_nonzero_rule:</code>	1685
<code>\draw_suspend_begin:</code>	1316	<code>_draw_backend_rectangle:nmnn</code>	1419
<code>\draw_suspend_end:</code>	1316	<code>_draw_backend_scope_begin:</code>	132, 1283
<code>\draw_transform_matrix:nmnn</code>	1739, 1828, 1830, 1832, 1838, 1840, 1851	<code>_draw_backend_scope_end:</code>	135, 1293
<code>\draw_transform_matrix_absolute:nmnn</code>	1713, 1754, 1819	<code>_draw_box_use:Nmnnn</code>	14, 65
<code>\draw_transform_matrix_invert:</code>	1771	<code>\l__draw_corner_arc_bool</code>	220, 228, 229, 233, 559
<code>\draw_transform_matrix_reset:</code>	1237, 1320, 1699	<code>\l__draw_corner_xarc_dim</code>	218, 223, 226, 236
<code>\draw_transform_rotate:n</code>	1841	<code>\l__draw_corner_yarc_dim</code>	218, 224, 227, 237
<code>\draw_transform_scale:n</code>	1827	<code>_draw_draw_polar:nmn</code>	847
<code>\draw_transform_shift:n</code>	1739, 1834, 1836	<code>_draw_draw_vec_polar:nmn</code>	1144, 1145, 1153
<code>\draw_transform_shift_absolute:n</code>	1713, 1824	<code>\l__draw_fill_color_tl</code>	1278
<code>\draw_transform_shift_invert:</code>	1771	<code>\g__draw_id_int</code>	1220, 1231
<code>\draw_transform_shift_reset:</code>	1238, 1321, 1699	<code>_draw_if_recursion_tail_stop-do:Nn</code>	9, 1453
<code>\draw_transform_triangle:nmn</code>	486, 1805	<code>\l__draw_layer_close_bool</code>	86, 96, 98, 115
<code>\draw_transform_xscale:n</code>	1827	<code>\l__draw_layer_main_box</code>	128, 129, 1218, 1247
<code>\draw_transform_xshift:n</code>	1827	<code>\l__draw_layer_tl</code>	84, 95, 99
<code>\draw_transform_xslant:n</code>	1827	<code>\g__draw_layers_clist</code>	87
<code>\draw_transform_yscale:n</code>	1827	<code>_draw_layers_insert:</code>	122, 1253
<code>\draw_transform_yshift:n</code>	1827	<code>_draw_layers_restore:</code>	139, 1326
<code>\draw_transform_yslant:n</code>	1827	<code>_draw_layers_save:</code>	139, 1322
<code>\draw_xvec:n</code>	1093, 1108	<code>\g__draw_linewidth_dim</code>	732, 1285, 1291, 1656, 1661, 1662
<code>\draw_yvec:n</code>	1093, 1109	<code>\l__draw_linewidth_dim</code>	1278, 1285, 1291
<code>\draw_zvec:n</code>	1093, 1110	<code>\l__draw_main_box</code>	1218, 1232, 1263, 1267, 1269, 1271, 1272, 1275
draw internal commands:		<code>\l__draw_matrix_a_fp</code>	42, 1166, 1198, 1693, 1701, 1715, 1721, 1755, 1757, 1781, 1790, 1797
<code>_draw_backend_begin:</code>	1233	<code>\l__draw_matrix_active_bool</code>	560, 1161, 1193, 1692, 1726, 1727, 1773
<code>_draw_backend_box_use:Nmnnn</code>	41	<code>\l__draw_matrix_b_fp</code>	43, 1172, 1203, 1693, 1702, 1716, 1722, 1756, 1758, 1782, 1792, 1793
<code>_draw_backend_cap_but:</code>	1681		
<code>_draw_backend_cap_rectangle:</code>	1682		
<code>_draw_backend_cap_round:</code>	1683		
<code>_draw_backend_clip:</code>	684		
<code>_draw_backend_closepath:</code>	1402		
<code>_draw_backend_curveto:nmnnnn</code>	1406		

\l__draw_matrix_c_fp	__draw_path_rectangle_corners:nmmm
. 44, 1167, 1199, 1693, 1703, 1717, 583
1723, 1755, 1757, 1782, 1794, 1795	__draw_path_rectangle_corners:nmmmm
\l__draw_matrix_d_fp 586, 589
. 45, 1173, 1204, 1696, 1704, 1718,	__draw_path_rectangle_rounded:nmmm
1724, 1756, 1758, 1781, 1791, 1796 554
__draw_path_arc:nmmm	__draw_path_reset_limits:
345 184, 658, 757, 1235
__draw_path_arc:nnNmm	\l__draw_path_tmp_tl
345 175, 423, 446, 465, 469, 473, 477
\c__draw_path_arc_60_fp	\l__draw_path_tmpa_fp
345 175, 311, 321, 333
\c__draw_path_arc_90_fp	\l__draw_path_tmpb_fp
345 175, 312, 328, 337
__draw_path_arc_add:nmmmm	__draw_path_update_last:nn
345 213, 250, 263, 282, 572
__draw_path_arc_aux_add:nn	__draw_path_update_limits:nn
452, 458, 470, 475 24, 27, 30, 33,
__draw_path_arc_auxi:nmmmmNmm	184, 248, 261, 278, 279, 280, 569, 570
345, 372, 379	__draw_path_use:n
__draw_path_arc_auxii:nmmmmmmmm 345	646
__draw_path_arc_auxiii:nn	__draw_path_use_action_draw: ..
345	646
__draw_path_arc_auxiv:nmmmm	__draw_path_use_action_fillstroke:
345 646
__draw_path_arc_auxv:nn	\l__draw_path_use_bb_bool
345	644
__draw_path_arc_auxvi:nn	\l__draw_path_use_clear_bool 644, 703
345	\l__draw_path_use_clip_bool
\l__draw_path_arc_delta_fp 641, 664, 682
345	\l__draw_path_use_fill_bool
\l__draw_path_arc_start_fp 641, 665, 687, 692, 698, 712
345	__draw_path_use_stroke_bb: ...
__draw_path_curveto:nmmmm	646
302	__draw_path_use_stroke_bb_-
__draw_path_curveto:nmmmmmm	aux:NmN
240, 298, 316, 446, 513, 523, 533, 543	646
\c__draw_path_curveto_a_fp	\l__draw_path_use_stroke_bool
302	641, 666, 679, 688, 693, 699, 708, 713
\c__draw_path_curveto_b_fp	\g__draw_path_xmax_dim
302 180, 186, 187, 208, 751, 773
__draw_path_ellipse:nmmmmmm	\l__draw_path_xmax_dim .
489	737, 751, 773
__draw_path_ellipse_arci:nmmmmmm 489	\g__draw_path_xmin_dim
__draw_path_ellipse_arcii:nmmmmmm 180, 188, 189, 209, 752, 774
489	\l__draw_path_xmin_dim .
__draw_path_ellipse_arciiii:nmmmmmm	737, 752, 774
489	\g__draw_path_ymax_dim
__draw_path_ellipse_arciv:nmmmmmm 180, 190, 191, 210, 753, 775
489	\l__draw_path_ymax_dim .
\c__draw_path_ellipse_fp	737, 753, 775
489	\g__draw_path_ymin_dim
__draw_path_grid_auxi:nmmmmmm 180, 192, 193, 211, 754, 776
591	\l__draw_path_ymin_dim .
__draw_path_grid_auxii:nmmmmmm .	737, 754, 776
591	__draw_point_interpolate_-
__draw_path_grid_auxiii:nmmmmmm 591	arcaxes_auxi:nmmmmmmmm
__draw_path_grid_auxiiii:nmmmmmm 591 985
__draw_path_grid_auxiv:nmmmmmmmm 591	__draw_point_interpolate_-
\g__draw_path_lastx_dim	arcaxes_auxii:nmmmmmmmm
..... 178, 215, 320, 453, 459, 749, 777 985
\l__draw_path_lastx_dim 737, 749, 777	__draw_point_interpolate_-
\g__draw_path_lasty_dim	arcaxes_auxiii:nmmmmmm
..... 178, 216, 327, 454, 460, 750, 778 985
\l__draw_path_lasty_dim 737, 750, 778	__draw_point_interpolate_-
__draw_path_lineto:nn	arcaxes_auxiv:nmmmmmmmm
240, 292 985
__draw_path_mark_corner:	
..... 231, 260, 269, 286, 297, 315, 386	
__draw_path_moveto:nn	
..... 240, 290, 501, 509	
__draw_path_rectangle:nmmmm	
554	

_draw_point_interpolate_curve_-auxi:nnnnnnnn	1020, 1023	_draw_point_process:nnnnn	790, 873, 1019
_draw_point_interpolate_curve_-auxii:nnnnnnnn	1025, 1029, 1037	_draw_point_process_auxi:nn	790
_draw_point_interpolate_curve_-auxiii:nnnnnn	1032, 1043, 1051	_draw_point_process_auxii:nw	790
_draw_point_interpolate_curve_-auxiv:nnnnnn	1045, 1046, 1047, 1052	_draw_point_process_auxiii:mnn	790
_draw_point_interpolate_curve_-auxv:nnw	1054, 1058, 1065	_draw_point_process_auxiv:nw	790
_draw_point_interpolate_curve_-auxvi:n	1049, 1070	_draw_point_process_auxvi:nw	790
_draw_point_interpolate_curve_-auxvii:nnnnnnnn	1071, 1072	_draw_point_process_auxvii:nnnnn	790
_draw_point_interpolate_curve_-auxviii:nnnnnn	1074, 1081, 1086	_draw_point_process_auxviii:nw	790
_draw_point_interpolate_-distance:nnnn	970, 973	_draw_point_to_dim:n	793, 803, 804, 814, 815, 816, 827, 828, 829, 830, 841, 1013, 1083, 1115, 1129, 1147, 1163, 1179, 1195, 1208
_draw_point_interpolate_-distance:nnnnn	967, 977	_draw_point_to_dim_aux:n	841
_draw_point_interpolate_-distance:nnnnnn	967	_draw_point_to_dim_aux:w	841
_draw_point_interpolate_line_-aux:nnnnn	952	_draw_point_transform:nn	1154
_draw_point_interpolate_line_-aux:nnnnnn	952	_draw_point_transform_noshift:n	427, 443, 494, 495, 1186
_draw_point_intersect_circles_-auxi:nnnnnnnn	896	_draw_point_transform_noshift:nn	1186
_draw_point_intersect_circles_-auxii:nnnnnnnn	896	_draw_point_unit_vector:nn	854
_draw_point_intersect_circles_-auxiii:nnnnnnnn	896	_draw_point_unit_vector:nnn	854
_draw_point_intersect_circles_-auxiv:nnnnnnnnn	896	_draw_point_vec:nn	1111
_draw_point_intersect_circles_-auxvi:nnnnnnnnn	896	_draw_point_vec:nnn	1111
_draw_point_intersect_circles_-auxvii:nnnnnnnn	896	_draw_point_vec_polar:nnn	1141
_draw_point_intersect_lines:nnnnnn	871	_draw_reset_bb:	1221, 1234, 1306
_draw_point_intersect_lines_-aux:nnnnnn	871	_draw_scope_bb_begin:	1299, 1318
_draw_point_process:nn	23, 26, 29, 32, 242, 254, 290, 292, 424, 440, 790, 855, 969, 975, 1101, 1156, 1188, 1731, 1763, 1809	_draw_scope_bb_end:	1299, 1328
_draw_point_process:nnn	304, 430, 556, 585, 593, 790, 898, 954, 1807	_draw_softpath_add:n	770, 1336, 1355, 1364, 1373, 1378, 1388, 1396, 1651
_draw_point_process:nnnn	267, 295, 491, 790, 987	\c_draw_softpath_arc_fp	1430, 1593, 1597, 1602, 1606
		_draw_softpath_clear:	657, 704, 762, 766, 1239, 1339
		_draw_softpath_close_op:nn	1357, 1401, 1492, 1528, 1630
		_draw_softpath_closepath:	287, 508, 1353
		\l_draw_softpath_corneri_dim	1424, 1477, 1480, 1566
		\l_draw_softpath_cornerii_dim	1424, 1478, 1481, 1557
		\g_draw_softpath_corners_bool	761, 768, 1335, 1347, 1398, 1433, 1449
		\l_draw_softpath_corners_bool	737, 760, 769
		\l_draw_softpath_curve_end_tl	1423, 1554, 1573, 1622, 1633
		_draw_softpath_curveto:nnnnnn	281, 1353

_draw_softpath_curveto_opi:nn .	1366, 1401 , 1489 , 1527 , 1590
_draw_softpath_curveto_-	
opi:nnNnnNnn	1401
_draw_softpath_curveto_opii:nn	1367, 1401 , 1599
_draw_softpath_curveto_-	
opiii:nn	1368, 1401 , 1608
\l_draw_softpath_first_tl	1424 , 1440 , 1457 , 1458 , 1468 , 1487 , 1488 , 1514 , 1518
\l_draw_softpath_internal_tl	1334 , 1341 , 1342 , 1442 , 1444
\g_draw_softpath_lastx_dim	755, 771 , 1349 , 1358 , 1382
\l_draw_softpath_lastx_dim	743, 755 , 771
\l_draw_softpath_lastx_fp	1424 , 1438 , 1459 , 1507 , 1569 , 1576
\g_draw_softpath_lasty_dim	756, 772 , 1349 , 1359 , 1383
\l_draw_softpath_lasty_dim	744, 756 , 772
\l_draw_softpath_lasty_fp	1424 , 1439 , 1460 , 1508 , 1570 , 1577
_draw_softpath_lineto:nn	262, 1353
_draw_softpath_lineto_op:nn	1374, 1401 , 1495 , 1526 , 1639
\g_draw_softpath_main_tl	758, 1333 , 1337 , 1341 , 1346 , 1442 , 1650
\l_draw_softpath_main_tl	19, 758 , 770 , 1421 , 1436 , 1463 , 1465 , 1646 , 1648 , 1651
\g_draw_softpath_move_bool	1351, 1380
\l_draw_softpath_move_tl	1424, 1441 , 1464 , 1467 , 1515 , 1617 , 1640 , 1647
_draw_softpath_moveto:nn	249, 1353
_draw_softpath_moveto_op:nn	1379, 1401 , 1461 , 1619
\l_draw_softpath_part_tl	1422, 1437 , 1466 , 1469 , 1471 , 1505 , 1560 , 1649
_draw_softpath_rectangle:nmnn	571, 1353
_draw_softpath_rectangle_-	
opi:nn	1390, 1401
_draw_softpath_rectangle_-	
opi:nnNnn	1401
_draw_softpath_rectangle_-	
opii:nn	1391, 1401
_draw_softpath_round_action:nn	1431
_draw_softpath_round_action:Nnn	1431
_draw_softpath_round_action_-	
close:	1431
_draw_softpath_round_action_-	
curveto:NnnNnn	1431
_draw_softpath_round_calc:NnnNnn	1431
_draw_softpath_round_calc:nmnnnn	1431
_draw_softpath_round_calc:nmnnnw	1431
_draw_softpath_round_close:nn	1431
_draw_softpath_round_close:w	1431
_draw_softpath_round_corners:	676, 1431
_draw_softpath_round_end:	1431
_draw_softpath_round_lookahead:NnnNnn	1431
_draw_softpath_round_loop:Nnn	1431
_draw_softpath_round_roundpoint:NnnNnnNnn	1431
_draw_softpath_roundpoint:nn	235, 1353
_draw_softpath_roundpoint_-	
op:nn	1397, 1401 , 1454 , 1536
_draw_softpath_use:	681, 1339
\l_draw_stroke_color_tl	1278
\l_draw_tmp_box	13, 37, 48, 52, 54, 55, 56, 57, 63, 65, 66, 67, 68, 69
\l_draw_tmp_seq	1664
_draw_transform_triangle:nmnnnn	1810, 1815
_draw_transform:nmnn	1739
_draw_transform_invert:n	1771
_draw_transform_rotate:n	1841
_draw_transform_rotate:nn	1841
_draw_transform_shift:nn	1739
_draw_transform_shift_absolute:nn	1713
_draw_vec:nn	1093
_draw_vec:nmnn	1093
\g_draw_xmax_dim	196, 197, 1213 , 1223 , 1258 , 1273 , 1302 , 1310
\l_draw_xmax_dim	1295, 1302 , 1310
\g_draw_xmin_dim	198, 199, 1213 , 1224 , 1256, 1259, 1265, 1273, 1303, 1311
\l_draw_xmin_dim	1295, 1303 , 1311
\l_draw_xshift_dim	50, 1168, 1182, 1693 , 1708, 1736, 1768, 1802
\l_draw_xvec_x_dim	1087, 1117, 1131, 1149
\l_draw_xvec_y_dim	1087, 1118, 1135

<code>\g__draw_ymax_dim</code>	200,	524, 525, 526, 527, 528, 529, 534,
	201, 1213 , 1225 , 1260 , 1270 , 1304 , 1312	535, 536, 537, 538, 539, 544, 545,
<code>\l__draw_ymax_dim</code>	1295 , 1304 , 1312	546, 547, 548, 549, 617, 618, 1592,
<code>\g__draw_ymin_dim</code>	202, 203, 1213 ,	1596, 1601, 1605, 1661, 1669, 1674
	1226 , 1261 , 1266 , 1270 , 1305 , 1313	<code>\fp_use:N</code>
<code>\l__draw_ymin_dim</code>	1295 , 1305 , 1313	42, 43, 44, 45, 551
<code>\l__draw_yshift_dim</code>	51, 1174 ,	<code>\fp_while_do:nNnn</code>
	1182 , 1693 , 1709 , 1737 , 1769 , 1803	368
<code>\l__draw_yvec_x_dim</code>	1087 , 1117 , 1132	<code>\fp_zero:N</code>
<code>\l__draw_yvec_y_dim</code>		1438, 1439, 1702, 1703
	1087 , 1118 , 1136 , 1150	<code>\c_one_fp</code>
<code>\l__draw_zvec_x_dim</code>	1087 , 1133	1721, 1724
<code>\l__draw_zvec_y_dim</code>	1087 , 1137	<code>\c_zero_fp</code>
		864, 1722, 1723
		G
<code>\end</code>	169, 784	group commands:
exp commands:		<code>\group_begin:</code>
<code>\exp_after:wN</code>		36, 62, 92,
	446, 464, 1443, 1517, 1620, 1631, 1640	103, 748, 1230, 1284, 1301, 1435, 1666
<code>\exp_args:Nf</code>	792, 858	<code>\group_end:</code>
<code>\exp_args:Nff</code>	802	58, 70, 117,
<code>\exp_args:Nfff</code>	813	120, 779, 1276, 1292, 1314, 1447, 1676
<code>\exp_args:Nffff</code>	826	H
<code>\exp_args:NNNV</code>	1251	hbox commands:
<code>\exp_args:Nx</code>	1680	<code>\hbox_gset:Nw</code>
<code>\exp_not:N</code>		101
	1562, 1590, 1599, 1608, 1617, 1620,	<code>\hbox_gset_end:</code>
	1621, 1622, 1627, 1631, 1632, 1633	118
		<code>\hbox_set:Nn</code>
		37, 48, 63, 1263
		<code>\hbox_set:Nw</code>
		1232, 1247
		<code>\hbox_set_end:</code>
		1251, 1255
		I
		int commands:
		<code>\int_gincr:N</code>
		1231
		<code>\int_if_odd:nTF</code>
		941
		<code>\int_new:N</code>
		1220
		K
		kernel internal commands:
		<code>__kernel_kern:n</code>
		50
		<code>__kernel_quark_new_test:N</code>
		9
		M
		mode commands:
		<code>\mode_leave_vertical:</code>
		1274
		msg commands:
		<code>\msg_error:nnn</code>
		78, 109, 110, 673
		<code>\msg_new:nnn</code>
		164
		<code>\msg_new:nnnn</code>
		161, 166, 781
		P
<code>\fp_compare:nNnTF</code>	360, 370, 864	<code>\pgfextractx</code>
<code>\fp_compare_p:nNn</code>		21
	1721, 1722, 1723, 1724	<code>\pgfextracty</code>
<code>\fp_const:Nn</code>		21
	343, 344, 482, 483, 551, 1430	<code>\pgfgetlastxy</code>
<code>\fp_eval:n</code>	352, 353, 374, 381, 390,	46
	842, 850, 859, 880, 881, 882, 883,	<code>\pgfgettransform</code>
	884, 885, 905, 910, 911, 918, 925,	46
	926, 933, 940, 942, 955, 960, 978,	<code>\pgfgettransformentries</code>
	994, 999, 1006, 1007, 1026, 1033,	45
	1055, 1056, 1075, 1076, 1077, 1078,	<code>\pgfinnerlinewidth</code>
	1112, 1125, 1144, 1680, 1744, 1745,	45
	1746, 1747, 1777, 1842, 1846, 1847	<code>\pgflowlevel</code>
<code>\fp_new:N</code>	176, 177, 480,	46
	481, 1424, 1425, 1693, 1694, 1695, 1696	<code>\pgflowlevelsynccm</code>
<code>\fp_set:Nn</code>	311, 312, 366, 367,	46
	447, 448, 1459, 1460, 1507, 1508,	<code>\pgfpatharcto</code>
	1576, 1577, 1701, 1704, 1715, 1716,	6
	1717, 1718, 1790, 1792, 1794, 1796	<code>\pgfpatharctoprecomputed</code>
<code>\fp_to_decimal:N</code>	373, 380, 388	6
<code>\fp_to_dim:n</code>	318, 325,	
	332, 336, 354, 355, 401, 410, 478,	
	502, 514, 515, 516, 517, 518, 519,	

