

# File I

## Implementation

### 1 l3draw implementation

```

1 <*package>
2 <@@=draw>
3 \ProvidesExplPackage{l3draw}{2026-06-18}{}
4 {L3 Experimental core drawing support}

```

#### 1.1 Internal auxiliaries

`\s__draw_mark` Internal scan marks.

```

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

```

*(End of definition for \s\_\_draw\_mark and \s\_\_draw\_stop.)*

`\q__draw_recursion_tail` Internal recursion quarks.

```

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

```

*(End of definition for \q\_\_draw\_recursion\_tail and \q\_\_draw\_recursion\_stop.)*

`\__draw_if_recursion_tail_stop_do:Nn` Functions to query recursion quarks.

```

9 \__kernel_quark_new_test:N \__draw_if_recursion_tail_stop_do:Nn

```

*(End of definition for \\_\_draw\_if\_recursion\_tail\_stop\_do:Nn.)*

Everything else is in the sub-files!

```

10 </package>

```

### 2 l3draw-boxes implementation

```

11 <*package>

```

```

12 <@@=draw>

```

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

`\l__draw_tmp_box`

```

13 \box_new:N \l__draw_tmp_box

```

*(End of definition for \l\_\_draw\_tmp\_box.)*

`\draw_box_use:N`

`\draw_box_use:Nn`

`\__draw_box_use:nNnnnnn`

`\__draw_box_use:Nnnnn`

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

```

14 \cs_new_protected:Npn \draw_box_use:N #1

```

```

15 {

```

```

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

```

```

70     \group_end:
71 }

```

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

<pre> \draw_coffin_use:Nnn \draw_coffin_use:Nnnn \__draw_coffin_use:nNnn </pre>	<p>Slightly more than a shortcut: we have to allow for the fact that coffins have no apparent width before the reference point.</p> <pre> 72 \cs_new_protected:Npn \draw_coffin_use:Nnn #1#2#3 73 { 74     \__draw_coffin_use:nNnn { \__draw_box_use:Nnnnnnn } 75     #1 {#2} {#3} 76 } 77 \cs_new_protected:Npn \draw_coffin_use:Nnnn #1#2#3#4 78 { 79     \__draw_coffin_use:nNnn { \__draw_box_use:nNnnnn {#4} } 80     #1 {#2} {#3} 81 } 82 \cs_new_protected:Npn \__draw_coffin_use:nNnn #1#2#3#4 83 { 84     \group_begin: 85     \hbox_set:Nn \l__draw_tmp_box 86     { \coffin_typeset:Nnnnn #2 {#3} {#4} { Opt } { Opt } } 87     #1 \l__draw_tmp_box 88     { \box_wd:N \l__draw_tmp_box - \coffin_wd:N #2 } 89     { -\box_dp:N \l__draw_tmp_box } 90     { \box_wd:N \l__draw_tmp_box } 91     { \box_ht:N \l__draw_tmp_box } 92     \group_end: 93 } </pre>
---	--

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

```

94 \</package>

```

### 3 l3draw-layers implementation

```

95 \<*package>
96 \<@@=draw>

```

#### 3.1 User interface

<pre> \l__draw_layer_tl </pre>	<p>The name of the current layer: we start off with <code>main</code>.</p> <pre> 97 \tl_new:N \l__draw_layer_tl 98 \tl_set:Nn \l__draw_layer_tl { main } </pre> <p>(End of definition for <code>\l__draw_layer_tl</code>.)</p>
<pre> \l__draw_layer_close_bool </pre>	<p>Used to track if a layer needs to be closed.</p> <pre> 99 \bool_new:N \l__draw_layer_close_bool </pre> <p>(End of definition for <code>\l__draw_layer_close_bool</code>.)</p>

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

```

\g__draw_layers_clist
100 \clist_new:N \l_draw_layers_clist
101 \clist_set:Nn \l_draw_layers_clist { main }
102 \clist_new:N \g__draw_layers_clist

```

*(End of definition for \l\_draw\_layers\_clist and \g\_\_draw\_layers\_clist. This variable is documented on page ??.)*

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

```

\__draw_layer_begin:n
\draw_layer_end:
103 \cs_new_protected:Npn \draw_layer_begin:n #1
104 {
105   \clist_if_in:NnTF \l_draw_layers_clist {#1}
106   {
107     \bool_lazy_or:nnF
108     { \str_if_eq_p:nn {#1} { main } }
109     { \box_if_exist_p:c { g__draw_layer_ #1 _box } }
110     {
111       \box_new:c { g__draw_layer_ #1 _box }
112       \box_new:c { l__draw_layer_ #1 _box }
113     }
114     \str_if_eq:nnTF {#1} { main }
115     { \msg_error:nnn { draw } { main-layer } }
116     { \__draw_layer_begin:n {#1} }
117   }
118   {
119     \bool_lazy_any:nF
120     {
121       { \str_if_eq_p:nn {#1} { main } }
122       { \str_if_eq_p:nn {#1} { discard } }
123       { \box_if_exist_p:c { g__draw_layer_ #1 _box } }
124     }
125     { \msg_error:nnn { draw } { unknown-layer } {#1} }
126   }
127 }
128 \cs_new_protected:Npn \__draw_layer_begin:n #1
129 {
130   \group_begin:
131   \str_if_eq:VnTF \l__draw_layer_tl {#1}
132   { \bool_set_false:N \l__draw_layer_close_bool }
133   {
134     \bool_set_true:N \l__draw_layer_close_bool
135     \tl_set:Nn \l__draw_layer_tl {#1}
136     \box_gset_wd:cn { g__draw_layer_ #1 _box } { Opt }
137     \hbox_gset_cw { g__draw_layer_ #1 _box }
138     \box_use_drop:c { g__draw_layer_ #1 _box }
139     \group_begin:
140   }
141   \draw_set_linewidth:n { \l_draw_default_linewidth_dim }
142 }
143 \cs_new_protected:Npn \draw_layer_end:
144 {
145   \bool_if:NT \l__draw_layer_close_bool

```

```

146         {
147             \group_end:
148             \hbox_gset_end:
149         }
150     \group_end:
151 }

```

(End of definition for \draw\_layer\_begin:n, \\_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.

```

152 \cs_new_protected:Npn \_draw_layers_insert:
153 {
154     \clist_map_inline:Nn \l_draw_layers_clist
155     {
156         \str_if_eq:nnTF {##1} { main }
157         {
158             \box_set_wd:Nn \l__draw_layer_main_box { Opt }
159             \box_use_drop:N \l__draw_layer_main_box
160         }
161         {
162             \__draw_backend_scope_begin:
163             \box_gset_wd:cn { g__draw_layer_ ##1 _box } { Opt }
164             \box_use_drop:c { g__draw_layer_ ##1 _box }
165             \__draw_backend_scope_end:
166         }
167     }
168 }

```

(End of definition for \\_draw\_layers\_insert:.)

\\_draw\_layers\_save: Simple save/restore functions. After creating a local copy of the layer box, we must clear the global one so nested drawings do not inherit the parent's layers.

```

169 \cs_new_protected:Npn \_draw_layers_save:
170 {
171     \clist_map_inline:Nn \l_draw_layers_clist
172     {
173         \str_if_eq:nnF {##1} { main }
174         {
175             \box_set_eq:cc { l__draw_layer_ ##1 _box }
176             { g__draw_layer_ ##1 _box }
177             \box_gclear:c { g__draw_layer_ ##1 _box }
178         }
179     }
180 }
181 \cs_new_protected:Npn \_draw_layers_restore:
182 {
183     \clist_map_inline:Nn \l_draw_layers_clist
184     {
185         \str_if_eq:nnF {##1} { main }
186         {
187             \box_gset_eq:cc { g__draw_layer_ ##1 _box }

```

```

188         { l__draw_layer_ ##1 _box }
189     }
190 }
191 }

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

192 \msg_new:nnnn { draw } { main-layer }
193 { Material~cannot~be~added~to~'main'~layer. }
194 { The~main~layer~may~only~be~accessed~at~the~top~level. }
195 \msg_new:nnnn { draw } { unknown-layer }
196 { Layer~'#1'~has~not~been~created. }
197 { You~have~tried~to~use~layer~'#1',~but~it~was~never~set~up. }
198 % \end{macrocode}
199 %
200 % \begin{macrocode}
201 \end{macrocode}

```

## 4 l3draw-paths implementation

```

202 \*package
203 \@@=draw

```

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

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

`\l__draw_path_tmp_tl` Scratch space.

```

\l__draw_path_tmpa_fp 204 \tl_new:N \l__draw_path_tmp_tl
\l__draw_path_tmpb_fp 205 \fp_new:N \l__draw_path_tmpa_fp
206 \fp_new:N \l__draw_path_tmpb_fp

```

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

### 4.1 Tracking paths

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

```

\g__draw_path_lasty_dim 207 \dim_new:N \g__draw_path_lastx_dim
208 \dim_new:N \g__draw_path_lasty_dim

```

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

$\backslash g\_draw\_path\_xmax\_dim$ $\backslash g\_draw\_path\_xmin\_dim$ $\backslash g\_draw\_path\_ymax\_dim$ $\backslash g\_draw\_path\_ymin\_dim$	<p>The limiting size of a path.</p> <pre> 209 \dim_new:N \g__draw_path_xmax_dim 210 \dim_new:N \g__draw_path_xmin_dim 211 \dim_new:N \g__draw_path_ymax_dim 212 \dim_new:N \g__draw_path_ymin_dim </pre> <p><i>(End of definition for <math>\backslash g\_draw\_path\_xmax\_dim</math> and others.)</i></p>
$\backslash\_draw\_path\_update\_limits:nn$ $\backslash\_draw\_path\_reset\_limits:$	<p>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.)</p> <pre> 213 \cs_new_protected:Npn \__draw_path_update_limits:nn #1#2 214 { 215   \dim_gset:Nn \g__draw_path_xmax_dim 216     { \dim_max:nn \g__draw_path_xmax_dim {#1} } 217   \dim_gset:Nn \g__draw_path_xmin_dim 218     { \dim_min:nn \g__draw_path_xmin_dim {#1} } 219   \dim_gset:Nn \g__draw_path_ymax_dim 220     { \dim_max:nn \g__draw_path_ymax_dim {#2} } 221   \dim_gset:Nn \g__draw_path_ymin_dim 222     { \dim_min:nn \g__draw_path_ymin_dim {#2} } 223   \bool_if:NT \l_draw_bb_update_bool 224   { 225     \dim_gset:Nn \g_draw_bb_xmax_dim 226       { \dim_max:nn \g_draw_bb_xmax_dim {#1} } 227     \dim_gset:Nn \g_draw_bb_xmin_dim 228       { \dim_min:nn \g_draw_bb_xmin_dim {#1} } 229     \dim_gset:Nn \g_draw_bb_ymax_dim 230       { \dim_max:nn \g_draw_bb_ymax_dim {#2} } 231     \dim_gset:Nn \g_draw_bb_ymin_dim 232       { \dim_min:nn \g_draw_bb_ymin_dim {#2} } 233   } 234 } 235 \cs_new_protected:Npn \__draw_path_reset_limits: 236 { 237   \dim_gset:Nn \g__draw_path_xmax_dim { -\c_max_dim } 238   \dim_gset:Nn \g__draw_path_xmin_dim { \c_max_dim } 239   \dim_gset:Nn \g__draw_path_ymax_dim { -\c_max_dim } 240   \dim_gset:Nn \g__draw_path_ymin_dim { \c_max_dim } 241 } </pre> <p><i>(End of definition for <math>\backslash\_draw\_path\_update\_limits:nn</math> and <math>\backslash\_draw\_path\_reset\_limits:.</math>)</i></p>
$\backslash\_draw\_path\_update\_last:nn$	<p>A simple auxiliary to avoid repetition.</p> <pre> 242 \cs_new_protected:Npn \__draw_path_update_last:nn #1#2 243 { 244   \dim_gset:Nn \g__draw_path_lastx_dim {#1} 245   \dim_gset:Nn \g__draw_path_lasty_dim {#2} 246 } </pre> <p><i>(End of definition for <math>\backslash\_draw\_path\_update\_last:nn</math>.)</i></p>

## 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 247 \dim_new:N \l__draw_corner_xarc_dim
248 \dim_new:N \l__draw_corner_yarc_dim

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

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

(End of definition for \l__draw_corner_arc_bool.)

\draw_path_corner_arc:nn Calculate the arcs, check they are non-zero.
250 \cs_new_protected:Npn \draw_path_corner_arc:nn #1#2
251 {
252   \dim_set:Nn \l__draw_corner_xarc_dim { \fp_to_dim:n {#1} }
253   \dim_set:Nn \l__draw_corner_yarc_dim { \fp_to_dim:n {#2} }
254   \bool_lazy_and:nnTF
255     { \dim_compare_p:nNn \l__draw_corner_xarc_dim = { 0pt } }
256     { \dim_compare_p:nNn \l__draw_corner_yarc_dim = { 0pt } }
257     { \bool_set_false:N \l__draw_corner_arc_bool }
258     { \bool_set_true:N \l__draw_corner_arc_bool }
259 }

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

\__draw_path_mark_corner: Mark up corners for arc post-processing.
260 \cs_new_protected:Npn \__draw_path_mark_corner:
261 {
262   \bool_if:NT \l__draw_corner_arc_bool
263   {
264     \__draw_softpath_roundpoint:VV
265       \l__draw_corner_xarc_dim
266       \l__draw_corner_yarc_dim
267   }
268 }

(End of definition for \__draw_path_mark_corner:.)
```

## 4.3 Basic path constructions

```
\draw_path_moveto:n At present, stick to purely linear transformation support and skip the soft path business:
\draw_path_lineto:n that will likely need to be revisited later.
\__draw_path_moveto:nn 269 \cs_new_protected:Npn \draw_path_moveto:n #1
\__draw_path_lineto:nn 270 {
\draw_path_curveto:nnn 271   \__draw_point_process:nn
\__draw_path_curveto:nnnnnn 272   { \__draw_path_moveto:nn }
273   { \draw_point_transform:n {#1} }
```



```

274 }
275 \cs_new_protected:Npn \__draw_path_moveto:nn #1#2
276 {
277   \__draw_path_update_limits:nn {#1} {#2}
278   \__draw_softpath_moveto:nn {#1} {#2}
279   \__draw_path_update_last:nn {#1} {#2}
280 }
281 \cs_new_protected:Npn \draw_path_lineto:n #1
282 {
283   \__draw_point_process:nn
284   { \__draw_path_lineto:nn }
285   { \draw_point_transform:n {#1} }
286 }
287 \cs_new_protected:Npn \__draw_path_lineto:nn #1#2
288 {
289   \__draw_path_mark_corner:
290   \__draw_path_update_limits:nn {#1} {#2}
291   \__draw_softpath_lineto:nn {#1} {#2}
292   \__draw_path_update_last:nn {#1} {#2}
293 }
294 \cs_new_protected:Npn \draw_path_curveto:nnn #1#2#3
295 {
296   \__draw_point_process:nnnn
297   {
298     \__draw_path_mark_corner:
299     \__draw_path_curveto:nnnnnn
300   }
301   { \draw_point_transform:n {#1} }
302   { \draw_point_transform:n {#2} }
303   { \draw_point_transform:n {#3} }
304 }
305 \cs_new_protected:Npn \__draw_path_curveto:nnnnnn #1#2#3#4#5#6
306 {
307   \__draw_path_update_limits:nn {#1} {#2}
308   \__draw_path_update_limits:nn {#3} {#4}
309   \__draw_path_update_limits:nn {#5} {#6}
310   \__draw_softpath_curveto:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
311   \__draw_path_update_last:nn {#5} {#6}
312 }

```

*(End of definition for \draw\_path\_moveto:n and others. These functions are documented on page ??.)*

`\draw_path_close:` A simple wrapper.

```

313 \cs_new_protected:Npn \draw_path_close:
314 {
315   \__draw_path_mark_corner:
316   \__draw_softpath_closepath:
317 }

```

*(End of definition for \draw\_path\_close:. This function is documented on page ??.)*

## 4.4 Accessing path information

`\draw_path_lastx:` Wrappers so the underlying data are read-only for users.

`\draw_path_lasty:`

```

318 \cs_new:Npn \draw_path_lastx: { \dim_use:N \g__draw_path_lastx_dim }
319 \cs_new:Npn \draw_path_lasty: { \dim_use:N \g__draw_path_lasty_dim }

```

(End of definition for \draw\_path\_lastx: and \draw\_path\_lasty:. These functions are documented on page ??.)

## 4.5 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
320 \cs_new_protected:Npn \draw_path_canvas_moveto:n #1
321 { \__draw_point_process:nn { \__draw_path_moveto:nn } { \draw_point:n {#1} } }
322 \cs_new_protected:Npn \draw_path_canvas_lineto:n #1
323 { \__draw_point_process:nn { \__draw_path_lineto:nn } { \draw_point:n {#1} } }
324 \cs_new_protected:Npn \draw_path_canvas_curveto:nnn #1#2#3
325 {
326   \__draw_point_process:nnnn
327   {
328     \__draw_path_mark_corner:
329     \__draw_path_curveto:nnnnnn
330   }
331   { \draw_point:n {#1} }
332   { \draw_point:n {#2} }
333   { \draw_point:n {#3} }
334 }

```

(End of definition for \draw\_path\_canvas\_moveto:n, \draw\_path\_canvas\_lineto:n, and \draw\_path\_canvas\_curveto:nnn. These functions are documented on page ??.)

## 4.6 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
\__draw_path_curveto:nnnn then
\c__draw_path_curveto_a_fp
\c__draw_path_curveto_b_fp

```

and

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

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

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

```

335 \cs_new_protected:Npn \draw_path_curveto:nn #1#2
336 {
337   \__draw_point_process:nnn
338   { \__draw_path_curveto:nnnn }
339   { \draw_point_transform:n {#1} }
340   { \draw_point_transform:n {#2} }
341 }
342 \cs_new_protected:Npn \__draw_path_curveto:nnnn #1#2#3#4
343 {
344   \fp_set:Nn \l__draw_path_tmpa_fp { \c__draw_path_curveto_b_fp * #1 }
345   \fp_set:Nn \l__draw_path_tmpb_fp { \c__draw_path_curveto_b_fp * #2 }
346   \use:e
347   {

```

```

348 \__draw_path_mark_corner:
349 \__draw_path_curveto:nnnnnn
350 {
351   \fp_to_dim:n
352   {
353     \c__draw_path_curveto_a_fp * \g__draw_path_lastx_dim
354     + \l__draw_path_tmpa_fp
355   }
356 }
357 {
358   \fp_to_dim:n
359   {
360     \c__draw_path_curveto_a_fp * \g__draw_path_lasty_dim
361     + \l__draw_path_tmpb_fp
362   }
363 }
364 {
365   \fp_to_dim:n
366   { \c__draw_path_curveto_a_fp * #3 + \l__draw_path_tmpa_fp }
367 }
368 {
369   \fp_to_dim:n
370   { \c__draw_path_curveto_a_fp * #4 + \l__draw_path_tmpb_fp }
371 }
372 {#3}
373 {#4}
374 }
375 }
376 \fp_const:Nn \c__draw_path_curveto_a_fp { 1 / 3 }
377 \fp_const:Nn \c__draw_path_curveto_b_fp { 2 / 3 }

```

(End of definition for \draw\_path\_curveto:nn and others. This function is documented on page ??.)

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

```

\draw_path_arc:nnnn
\__draw_path_arc:nnnn
\__draw_path_arc:nnNnn
  \__draw_path_arc_auxi:nnnnNnn
  \__draw_path_arc_auxi:enenNnn
  \__draw_path_arc_auxi:eennNnn
  \__draw_path_arc_auxii:nnnNnnnn
\__draw_path_arc_auxiii:nn
\__draw_path_arc_auxiv:nnnn
  \__draw_path_arc_auxv:nn
  \__draw_path_arc_auxvi:nn
  \__draw_path_arc_add:nnnn
\l__draw_path_arc_delta_fp
\l__draw_path_arc_start_fp
  \c__draw_path_arc_90_fp
  \c__draw_path_arc_60_fp
378 \cs_new_protected:Npn \draw_path_arc:nnn #1#2#3
379   { \draw_path_arc:nnnn {#1} {#1} {#2} {#3} }
380 \cs_new_protected:Npn \draw_path_arc:nnnn #1#2#3#4
381   {
382     \use:e
383     {
384       \__draw_path_arc:nnnn
385       { \fp_eval:n {#3} }
386       { \fp_eval:n {#4} }
387       { \fp_to_dim:n {#1} }
388       { \fp_to_dim:n {#2} }
389     }
390   }
391 \cs_new_protected:Npn \__draw_path_arc:nnnn #1#2#3#4
392   {
393     \fp_compare:nNnTF {#1} > {#2}
394     { \__draw_path_arc:nnNnn {#1} {#2} - {#3} {#4} }
395     { \__draw_path_arc:nnNnn {#1} {#2} + {#3} {#4} }

```

```

396 }
397 \cs_new_protected:Npn \__draw_path_arc:nnNnn #1#2#3#4#5
398 {
399   \fp_set:Nn \l__draw_path_arc_start_fp {#1}
400   \fp_set:Nn \l__draw_path_arc_delta_fp { abs( #1 - #2 ) }
401   \fp_while_do:nNnn { \l__draw_path_arc_delta_fp } > { 90 }
402   {
403     \fp_compare:nNnTF \l__draw_path_arc_delta_fp > { 115 }
404     {
405       \__draw_path_arc_auxi:eennNnn
406       { \fp_to_decimal:N \l__draw_path_arc_start_fp }
407       { \fp_eval:n { \l__draw_path_arc_start_fp #3 90 } }
408       { 90 } {#2}
409       #3 {#4} {#5}
410     }
411     {
412       \__draw_path_arc_auxi:eennNnn
413       { \fp_to_decimal:N \l__draw_path_arc_start_fp }
414       { \fp_eval:n { \l__draw_path_arc_start_fp #3 60 } }
415       { 60 } {#2}
416       #3 {#4} {#5}
417     }
418   }
419   \__draw_path_mark_corner:
420   \__draw_path_arc_auxi:enenNnn
421   { \fp_to_decimal:N \l__draw_path_arc_start_fp }
422   {#2}
423   { \fp_eval:n { abs( \l__draw_path_arc_start_fp - #2 ) } }
424   {#2}
425   #3 {#4} {#5}
426 }

```

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  $60^\circ$  pre-calculated for speed.

```

427 \cs_new_protected:Npn \__draw_path_arc_auxi:nnnnNnn #1#2#3#4#5#6#7
428 {
429   \use:e
430   {
431     \__draw_path_arc_auxii:nnnnNnnnn
432     {#1} {#2} {#4} #5 {#6} {#7}
433     {
434       \fp_to_dim:n
435       {
436         \cs_if_exist_use:cF
437         { c__draw_path_arc_ #3 _fp }
438         { 4/3 * tand( 0.25 * #3 ) }
439         * #6
440       }
441     }
442     {
443       \fp_to_dim:n
444       {

```

```

445         \cs_if_exist_use:cF
446         { c__draw_path_arc_ #3 _fp }
447         { 4/3 * tand( 0.25 * #3 ) }
448         * #7
449     }
450 }
451 }
452 }
453 \cs_generate_variant:Nn \__draw_path_arc_auxi:nnnnNnn { ene , ee }

```

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

```

454 \cs_new_protected:Npn \__draw_path_arc_auxii:nnnNnnnn #1#2#3#4#5#6#7#8
455 {
456     \tl_clear:N \l__draw_path_tmp_tl
457     \__draw_point_process:nn
458     { \__draw_path_arc_auxiii:nn }
459     {
460         \__draw_point_transform_noshift:n
461         { \draw_point_polar:nnn {#7} {#8} { #1 #4 90 } }
462     }
463     \__draw_point_process:nnn
464     { \__draw_path_arc_auxiv:nnnn }
465     {
466         \draw_point_transform:n
467         { \draw_point_polar:nnn {#5} {#6} {#1} }
468     }
469     {
470         \draw_point_transform:n
471         { \draw_point_polar:nnn {#5} {#6} {#2} }
472     }
473     \__draw_point_process:nn
474     { \__draw_path_arc_auxv:nn }
475     {
476         \__draw_point_transform_noshift:n
477         { \draw_point_polar:nnn {#7} {#8} { #2 #4 -90 } }
478     }
479     \exp_after:wN \__draw_path_curveto:nnnnnn \l__draw_path_tmp_tl
480     \fp_set:Nn \l__draw_path_arc_delta_fp { abs ( #2 - #3 ) }
481     \fp_set:Nn \l__draw_path_arc_start_fp {#2}
482 }

```

The first control point.

```

483 \cs_new_protected:Npn \__draw_path_arc_auxiii:nn #1#2
484 {
485     \__draw_path_arc_aux_add:nn
486     { \g__draw_path_lastx_dim + #1 }
487     { \g__draw_path_lasty_dim + #2 }
488 }

```

The end point: simple arithmetic.

```

489 \cs_new_protected:Npn \__draw_path_arc_auxiv:nnnn #1#2#3#4

```

```

490 {
491   \__draw_path_arc_aux_add:nn
492   { \g__draw_path_lastx_dim - #1 + #3 }
493   { \g__draw_path_lasty_dim - #2 + #4 }
494 }

```

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

```

495 \cs_new_protected:Npn \__draw_path_arc_auxv:nn #1#2
496 {
497   \exp_after:wN \__draw_path_arc_auxvi:nn
498   \l__draw_path_tmp_tl {#1} {#2}
499 }
500 \cs_new_protected:Npn \__draw_path_arc_auxvi:nn #1#2#3#4#5#6
501 {
502   \tl_set:Nn \l__draw_path_tmp_tl { {#1} {#2} }
503   \__draw_path_arc_aux_add:nn
504   { #5 + #3 }
505   { #6 + #4 }
506   \tl_put_right:Nn \l__draw_path_tmp_tl { {#3} {#4} }
507 }
508 \cs_new_protected:Npn \__draw_path_arc_aux_add:nn #1#2
509 {
510   \tl_put_right:Ne \l__draw_path_tmp_tl
511   { { \fp_to_dim:n {#1} } { \fp_to_dim:n {#2} } }
512 }
513 \fp_new:N \l__draw_path_arc_delta_fp
514 \fp_new:N \l__draw_path_arc_start_fp
515 \fp_const:cn { c__draw_path_arc_90_fp } { 4/3 * (sqrt(2) - 1) }
516 \fp_const:cn { c__draw_path_arc_60_fp } { 4/3 * tand(15) }

```

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

\draw\_path\_arc\_axes:nnnn A simple wrapper.

```

517 \cs_new_protected:Npn \draw_path_arc_axes:nnnn #1#2#3#4
518 {
519   \group_begin:
520   \draw_transform_triangle:nnn { 0cm , 0cm } {#1} {#2}
521   \draw_path_arc:nnn { 1pt } {#3} {#4}
522   \group_end:
523 }

```

(End of definition for \draw\_path\_arc\_axes:nnnn. This function is documented on page ??.)

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

```

\__draw_path_ellipse:nnnnnn
\__draw_path_ellipse_arci:nnnnnn
\__draw_path_ellipse_arci:nnnnnn
\__draw_path_ellipse_arci:nnnnnn
\__draw_path_ellipse_arci:nnnnnn
524 \cs_new_protected:Npn \draw_path_ellipse:nnn #1#2#3
525 {
526   \__draw_point_process:nnnn
527   { \__draw_path_ellipse:nnnnnn }
528   { \draw_point_transform:n {#1} }
529   { \__draw_point_transform_noshift:n {#2} }
530   { \__draw_point_transform_noshift:n {#3} }
531 }

```

```

532 \cs_new_protected:Npn \__draw_path_ellipse:nnnnnn #1#2#3#4#5#6
533 {
534   \use:e
535   {
536     \__draw_path_moveto:nn
537     { \fp_to_dim:n { #1 + #3 } } { \fp_to_dim:n { #2 + #4 } }
538     \__draw_path_ellipse_arci:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
539     \__draw_path_ellipse_arcii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
540     \__draw_path_ellipse_arciiii:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
541     \__draw_path_ellipse_arciv:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
542   }
543   \__draw_softpath_closepath:
544   \__draw_path_moveto:nn {#1} {#2}
545 }
546 \cs_new:Npn \__draw_path_ellipse_arci:nnnnnn #1#2#3#4#5#6
547 {
548   \__draw_path_curveto:nnnnnn
549   { \fp_to_dim:n { #1 + #3 + #5 * \c__draw_path_ellipse_fp } }
550   { \fp_to_dim:n { #2 + #4 + #6 * \c__draw_path_ellipse_fp } }
551   { \fp_to_dim:n { #1 + #3 * \c__draw_path_ellipse_fp + #5 } }
552   { \fp_to_dim:n { #2 + #4 * \c__draw_path_ellipse_fp + #6 } }
553   { \fp_to_dim:n { #1 + #5 } }
554   { \fp_to_dim:n { #2 + #6 } }
555 }
556 \cs_new:Npn \__draw_path_ellipse_arcii:nnnnnn #1#2#3#4#5#6
557 {
558   \__draw_path_curveto:nnnnnn
559   { \fp_to_dim:n { #1 - #3 * \c__draw_path_ellipse_fp + #5 } }
560   { \fp_to_dim:n { #2 - #4 * \c__draw_path_ellipse_fp + #6 } }
561   { \fp_to_dim:n { #1 - #3 + #5 * \c__draw_path_ellipse_fp } }
562   { \fp_to_dim:n { #2 - #4 + #6 * \c__draw_path_ellipse_fp } }
563   { \fp_to_dim:n { #1 - #3 } }
564   { \fp_to_dim:n { #2 - #4 } }
565 }
566 \cs_new:Npn \__draw_path_ellipse_arciiii:nnnnnn #1#2#3#4#5#6
567 {
568   \__draw_path_curveto:nnnnnn
569   { \fp_to_dim:n { #1 - #3 - #5 * \c__draw_path_ellipse_fp } }
570   { \fp_to_dim:n { #2 - #4 - #6 * \c__draw_path_ellipse_fp } }
571   { \fp_to_dim:n { #1 - #3 * \c__draw_path_ellipse_fp - #5 } }
572   { \fp_to_dim:n { #2 - #4 * \c__draw_path_ellipse_fp - #6 } }
573   { \fp_to_dim:n { #1 - #5 } }
574   { \fp_to_dim:n { #2 - #6 } }
575 }
576 \cs_new:Npn \__draw_path_ellipse_arciv:nnnnnn #1#2#3#4#5#6
577 {
578   \__draw_path_curveto:nnnnnn
579   { \fp_to_dim:n { #1 + #3 * \c__draw_path_ellipse_fp - #5 } }
580   { \fp_to_dim:n { #2 + #4 * \c__draw_path_ellipse_fp - #6 } }
581   { \fp_to_dim:n { #1 + #3 - #5 * \c__draw_path_ellipse_fp } }
582   { \fp_to_dim:n { #2 + #4 - #6 * \c__draw_path_ellipse_fp } }
583   { \fp_to_dim:n { #1 + #3 } }
584   { \fp_to_dim:n { #2 + #4 } }
585 }

```

```
586 \fp_const:Nn \c__draw_path_ellipse_fp { \fp_use:c { c__draw_path_arc_90_fp } }
```

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

\draw\_path\_circle:nn A shortcut.

```
587 \cs_new_protected:Npn \draw_path_circle:nn #1#2
588 { \draw_path_ellipse:nnn {#1} { #2 , Opt } { Opt , #2 } }
```

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

## 4.7 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
589 \cs_new_protected:Npn \draw_path_rectangle:nn #1#2
590 {
591   \bool_lazy_or:nnTF
592     { \l__draw_corner_arc_bool }
593     { \l__draw_matrix_active_bool }
594   {
595     \__draw_point_process:nnn \__draw_path_rectangle_rounded:nnnn
596     { \draw_point:n {#1} }
597     { \draw_point:n {#2} }
598   }
599   {
600     \__draw_point_process:nnn \__draw_path_rectangle:nnnn
601     { \draw_point:n { (#1) + ( \l__draw_xshift_dim , \l__draw_yshift_dim ) } }
602     { \draw_point:n {#2} }
603   }
604 }
605 \cs_new_protected:Npn \__draw_path_rectangle:nnnn #1#2#3#4
606 {
607   \__draw_path_update_limits:nn {#1} {#2}
608   \__draw_path_update_limits:nn { #1 + #3 } { #2 + #4 }
609   \__draw_softpath_rectangle:nnnn {#1} {#2} {#3} {#4}
610   \__draw_path_update_last:nn {#1} {#2}
611 }
612 \cs_new_protected:Npn \__draw_path_rectangle_rounded:nnnn #1#2#3#4
613 {
614   \draw_path_moveto:n { #1 + #3 , #2 + #4 }
615   \draw_path_lineto:n { #1 , #2 + #4 }
616   \draw_path_lineto:n { #1 , #2 }
617   \draw_path_lineto:n { #1 + #3 , #2 }
618   \draw_path_close:
619   \draw_path_moveto:n { #1 , #2 }
620 }
```

(End of definition for \draw\_path\_rectangle:nn, \\_\_draw\_path\_rectangle:nnnn, and \\_\_draw\_path\_rectangle\_rounded:nnnn. This function is documented on page ??.)

\draw\_path\_rectangle\_corners:nn Another shortcut wrapper.

```
\__draw_path_rectangle_corners:nnnn
621 \cs_new_protected:Npn \draw_path_rectangle_corners:nn #1#2
622 {
623   \__draw_point_process:nnn
```



```

624     { \__draw_path_rectangle_corners:nnnnn {#1} }
625     { \draw_point:n {#1} }
626     { \draw_point:n {#2} }
627   }
628   \cs_new_protected:Npn \__draw_path_rectangle_corners:nnnnn #1#2#3#4#5
629     { \draw_path_rectangle:nn {#1} { #4 - #2 , #5 - #3 } }

```

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

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

```

```

670     }
671     \dim_step_inline:nnnn
672     {#2}
673     {#4}
674     {#8}
675     {
676         \draw_path_moveto:n { #5 , ##1 }
677         \draw_path_lineto:n { #7 , ##1 }
678     }
679 }
680 \cs_generate_variant:Nn \__draw_path_grid_auxiv:nnnnnnnn { ee }

```

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

## 4.9 Using paths

Actions to pass to the driver.

```

\l__draw_path_use_clip_bool
\l__draw_path_use_fill_bool
\l__draw_path_use_stroke_bool
681 \bool_new:N \l__draw_path_use_clip_bool
682 \bool_new:N \l__draw_path_use_fill_bool
683 \bool_new:N \l__draw_path_use_stroke_bool

```

(End of definition for \l\_\_draw\_path\_use\_clip\_bool, \l\_\_draw\_path\_use\_fill\_bool, and \l\_\_draw\_path\_use\_stroke\_bool.)

Actions handled at the macro layer.

```

684 \bool_new:N \l__draw_path_use_clear_bool

```

(End of definition for \l\_\_draw\_path\_use\_clear\_bool.)

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

```

\draw_path_use:n
\draw_path_use_clear:n
\draw_path_replace_bb:
\__draw_path_replace_bb:NnN
\__draw_path_use:n
\__draw_path_use_action_draw:
\__draw_path_use_action_fillstroke:
\__draw_path_use_stroke_bb:
\__draw_path_use_bb:NnN
685 \cs_new_protected:Npn \draw_path_use:n #1
686 {
687     \tl_if_blank:nF {#1}
688     { \__draw_path_use:n {#1} }
689 }
690 \cs_new_protected:Npn \draw_path_use_clear:n #1
691 {
692     \bool_lazy_or:nnTF
693     { \tl_if_blank_p:n {#1} }
694     { \str_if_eq_p:nn {#1} { clear } }
695     {
696         \__draw_softpath_clear:
697         \__draw_path_reset_limits:
698     }
699     { \__draw_path_use:n { #1 , clear } }
700 }
701 \cs_new_protected:Npn \draw_path_replace_bb:
702 {
703     \__draw_path_replace_bb:NnN x { max } +
704     \__draw_path_replace_bb:NnN y { max } +
705     \__draw_path_replace_bb:NnN x { min } -
706     \__draw_path_replace_bb:NnN y { min } -

```

```

707     \__draw_softpath_clear:
708     \__draw_path_reset_limits:
709 }
710 \cs_new_protected:Npn \__draw_path_replace_bb:NnN #1#2#3
711 {
712     \dim_gset:cn { g_draw_bb_ #1#2 _dim }
713     {
714         \dim_use:c { g__draw_path_ #1#2 _dim }
715         #3 0.5 \g__draw_linewidth_dim
716     }
717 }

```

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).

```

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

```

```

757     }
758   }
759   \bool_if:NT \l__draw_path_use_clear_bool
760   {
761     \__draw_softpath_clear:
762     \__draw_path_reset_limits:
763   }
764 }
765 \cs_new_protected:Npn \__draw_path_use_action_draw:
766 {
767   \bool_set_true:N \l__draw_path_use_stroke_bool
768 }
769 \cs_new_protected:Npn \__draw_path_use_action_fillstroke:
770 {
771   \bool_set_true:N \l__draw_path_use_fill_bool
772   \bool_set_true:N \l__draw_path_use_stroke_bool
773 }

```

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.

```

774 \cs_new_protected:Npn \__draw_path_use_stroke_bb:
775 {
776   \__draw_path_use_bb:NnN x { max } +
777   \__draw_path_use_bb:NnN y { max } +
778   \__draw_path_use_bb:NnN x { min } -
779   \__draw_path_use_bb:NnN y { min } -
780 }
781 \cs_new_protected:Npn \__draw_path_use_bb:NnN #1#2#3
782 {
783   \dim_compare:nNnF { \dim_use:c { g_draw_bb_ #1#2 _dim } } = { #3 -\c_max_dim }
784   {
785     \dim_gset:cn { g_draw_bb_ #1#2 _dim }
786     {
787       \use:c { dim_ #2 :nn }
788       { \dim_use:c { g_draw_bb_ #1#2 _dim } }
789       {
790         \dim_use:c { g__draw_path_ #1#2 _dim }
791         #3 0.5 \g__draw_linewidth_dim
792       }
793     }
794   }
795 }

```

*(End of definition for \draw\_path\_use:n and others. These functions are documented on page ??.)*

## 4.10 Scoping paths

Local storage for global data. There is already a \l\_\_draw\_softpath\_main\_tl for path manipulation, so we can reuse that (it is always grouped when the path is being reconstructed).

```

\l__draw_path_lastx_dim
\l__draw_path_lasty_dim
\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

```

```

796 \dim_new:N \l__draw_path_lastx_dim
797 \dim_new:N \l__draw_path_lasty_dim
798 \dim_new:N \l__draw_path_xmax_dim
799 \dim_new:N \l__draw_path_xmin_dim

```

```

800 \dim_new:N \l__draw_path_ymax_dim
801 \dim_new:N \l__draw_path_ymin_dim
802 \dim_new:N \l__draw_softpath_lastx_dim
803 \dim_new:N \l__draw_softpath_lasty_dim
804 \bool_new:N \l__draw_softpath_corners_bool

```

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

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

```

805 \cs_new_protected:Npn \draw_path_scope_begin:
806 {
807   \group_begin:
808     \dim_set_eq:NN \l__draw_path_lastx_dim \g__draw_path_lastx_dim
809     \dim_set_eq:NN \l__draw_path_lasty_dim \g__draw_path_lasty_dim
810     \dim_set_eq:NN \l__draw_path_xmax_dim \g__draw_path_xmax_dim
811     \dim_set_eq:NN \l__draw_path_xmin_dim \g__draw_path_xmin_dim
812     \dim_set_eq:NN \l__draw_path_ymax_dim \g__draw_path_ymax_dim
813     \dim_set_eq:NN \l__draw_path_ymin_dim \g__draw_path_ymin_dim
814     \dim_set_eq:NN \l__draw_softpath_lastx_dim \g__draw_softpath_lastx_dim
815     \dim_set_eq:NN \l__draw_softpath_lasty_dim \g__draw_softpath_lasty_dim
816     \__draw_path_reset_limits:
817     \__draw_softpath_save:
818   }
819 \cs_new_protected:Npn \draw_path_scope_end:
820 {
821   \__draw_softpath_restore:
822   \dim_gset_eq:NN \g__draw_softpath_lastx_dim \l__draw_softpath_lastx_dim
823   \dim_gset_eq:NN \g__draw_softpath_lasty_dim \l__draw_softpath_lasty_dim
824   \dim_gset_eq:NN \g__draw_path_xmax_dim \l__draw_path_xmax_dim
825   \dim_gset_eq:NN \g__draw_path_xmin_dim \l__draw_path_xmin_dim
826   \dim_gset_eq:NN \g__draw_path_ymax_dim \l__draw_path_ymax_dim
827   \dim_gset_eq:NN \g__draw_path_ymin_dim \l__draw_path_ymin_dim
828   \dim_gset_eq:NN \g__draw_path_lastx_dim \l__draw_path_lastx_dim
829   \dim_gset_eq:NN \g__draw_path_lasty_dim \l__draw_path_lasty_dim
830   \group_end:
831 }

```

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

## 4.11 Messages

```

832 \msg_new:nnnn { draw } { invalid-path-action }
833 { Invalid~action~'#1'~for~path. }
834 { Paths~can~be~used~with~actions~'draw',~'clip',~'fill'~or~'stroke'. }
835 % \end{macrocode}
836 %
837 % \begin{macrocode}
838 \end{package}

```

## 5 l3draw-points implementation

```

839 \begin{package}

```

840 `<@=draw>`

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

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

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

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

## 5.1 Support functions

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

```

\__draw_point_process:nn
  \__draw_point_process_auxi:nn
    \__draw_point_process_auxi:en
      \__draw_point_process_auxii:nw
\__draw_point_process:nnn
  \__draw_point_process_auxiii:nnn
    \__draw_point_process_auxiii:een
      \__draw_point_process_auxiv:nw
\__draw_point_process:nnnn
  \__draw_point_process_auxv:nnnn
    \__draw_point_process_auxv:eeen
      \__draw_point_process_auxvi:nw
\__draw_point_process:nnnnn
  \__draw_point_process_auxvii:nnnnn
    \__draw_point_process_auxvii:eeeen
      \__draw_point_process_auxviii:nw
841 \cs_new:Npn \__draw_point_process:nn #1#2
842   { \__draw_point_process_auxi:en {#2} {#1} }
843 \cs_new:Npn \__draw_point_process_auxi:nn #1#2
844   { \__draw_point_process_auxii:nw {#2} #1 \s__draw_stop }
845 \cs_generate_variant:Nn \__draw_point_process_auxi:nn { e }
846 \cs_new:Npn \__draw_point_process_auxii:nw #1 #2 , #3 \s__draw_stop
847   { #1 {#2} {#3} }
848 \cs_new:Npn \__draw_point_process:nnn #1#2#3
849   { \__draw_point_process_auxiii:een {#2} {#3} {#1} }
850 \cs_new:Npn \__draw_point_process_auxiii:nnn #1#2#3
851   { \__draw_point_process_auxiv:nw {#3} #1 \s__draw_mark #2 \s__draw_stop }
852 \cs_generate_variant:Nn \__draw_point_process_auxiii:nnn { ee }
853 \cs_new:Npn \__draw_point_process_auxiv:nw #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_stop
854   { #1 {#2} {#3} {#4} {#5} }
855 \cs_new:Npn \__draw_point_process:nnnn #1#2#3#4
856   { \__draw_point_process_auxv:eeen {#2} {#3} {#4} {#1} }
857 \cs_new:Npn \__draw_point_process_auxv:nnnn #1#2#3#4
858   { \__draw_point_process_auxvi:nw {#4} #1 \s__draw_mark #2 \s__draw_mark #3 \s__draw_stop }
859 \cs_generate_variant:Nn \__draw_point_process_auxv:nnnn { eee }

```

```

860 \cs_new:Npn \__draw_point_process_auxvi:nw
861   #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_mark #6 , #7 \s__draw_stop
862   { #1 {#2} {#3} {#4} {#5} {#6} {#7} }
863 \cs_new:Npn \__draw_point_process_nnnnn #1#2#3#4#5
864   { \__draw_point_process_auxvii:eeeeen {#2} {#3} {#4} {#5} {#1} }
865 \cs_new:Npn \__draw_point_process_auxvii:nnnnn #1#2#3#4#5
866   {
867     \__draw_point_process_auxviii:nw
868     {#5} #1 \s__draw_mark #2 \s__draw_mark #3 \s__draw_mark #4 \s__draw_stop
869   }
870 \cs_generate_variant:Nn \__draw_point_process_auxvii:nnnnn { eeee }
871 \cs_new:Npn \__draw_point_process_auxviii:nw
872   #1 #2 , #3 \s__draw_mark #4 , #5 \s__draw_mark #6 , #7 \s__draw_mark #8 , #9 \s__draw_stop
873   { #1 {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9} }

```

(End of definition for \\_\_draw\_point\_process:nn and others.)

## 5.2 Basic points

\draw\_point:n Coordinates are always returned as two dimensions.

```

\__draw_point_to_dim:n 874 \cs_new:Npn \draw_point:n #1
\__draw_point_to_dim:e 875   { \__draw_point_to_dim:e { \fp_eval:n {#1} } }
\__draw_point_to_dim:w 876 \cs_new:Npn \__draw_point_to_dim:n #1
                        877   { \__draw_point_to_dim:w #1 }
                        878 \cs_generate_variant:Nn \__draw_point_to_dim:n { e }
                        879 \cs_new:Npn \__draw_point_to_dim:w ( #1 , ~ #2 ) { #1pt , #2pt }

```

## 5.3 Polar coordinates

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

```

\draw_point_polar:nn 880 \cs_new:Npn \draw_point_polar:nn #1#2
\draw_point_polar:nnn 881   { \draw_point_polar:nnn {#1} {#1} {#2} }
\__draw_draw_polar:nnn 882 \cs_new:Npn \draw_point_polar:nnn #1#2#3
\__draw_draw_polar:enn 883   { \__draw_draw_polar:enn { \fp_eval:n {#3} } {#1} {#2} }
                        884 \cs_new:Npn \__draw_draw_polar:nnn #1#2#3
                        885   { \draw_point:n { cosd(#1) * (#2) , sind(#1) * (#3) } }
                        886 \cs_generate_variant:Nn \__draw_draw_polar:nnn { e }

```

## 5.4 Point expression arithmetic

These functions all take point expressions as arguments.

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

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

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

```

887 \cs_new:Npn \draw_point_unit_vector:n #1
888   { \__draw_point_process:nn { \__draw_point_unit_vector:nn } { \draw_point:n {#1} } }
889 \cs_new:Npn \__draw_point_unit_vector:nn #1#2

```

```

890 {
891   \_draw_point_unit_vector:nnn
892   { \fp_eval:n { (sqrt(#1 * #1 + #2 * #2)) } }
893   {#1} {#2}
894 }
895 \cs_new:Npn \_draw_point_unit_vector:nnn #1#2#3
896 {
897   \fp_compare:nNnTF {#1} = \c_zero_fp
898   { Opt, 1pt }
899   {
900     \draw_point:n
901     { ( #2 , #3 ) / #1 }
902   }
903 }
904 \cs_generate_variant:Nn \_draw_point_unit_vector:nnn { e }

```

## 5.5 Intersection calculations

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

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

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.)

```

905 \cs_new:Npn \draw_point_intersect_lines:nnnn #1#2#3#4
906 {
907   \_draw_point_process:nnnnn
908   { \_draw_point_intersect_lines:nnnnnnnn }
909   { \draw_point:n {#1} }
910   { \draw_point:n {#2} }
911   { \draw_point:n {#3} }
912   { \draw_point:n {#4} }
913 }

```

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.

```

914 \cs_new:Npn \__draw_point_intersect_lines:nnnnnnnn #1#2#3#4#5#6#7#8
915 {
916   \__draw_point_intersect_lines_aux:eeeeeee
917   { \fp_eval:n { #1 * #4 - #2 * #3 } }
918   { \fp_eval:n { #5 * #8 - #6 * #7 } }
919   { \fp_eval:n { #1 - #3 } }
920   { \fp_eval:n { #5 - #7 } }
921   { \fp_eval:n { #2 - #4 } }
922   { \fp_eval:n { #6 - #8 } }
923 }
924 \cs_new:Npn \__draw_point_intersect_lines_aux:nnnnnn #1#2#3#4#5#6
925 {
926   \draw_point:n
927   {
928     ( #2 * #3 - #1 * #4 , #2 * #5 - #1 * #6 )
929     / ( #4 * #5 - #6 * #3 )
930   }
931 }
932 \cs_generate_variant:Nn \__draw_point_intersect_lines_aux:nnnnnn { eeeeeee }

```

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.

```

933 \cs_new:Npn \draw_point_intersect_circles:nnnnn #1#2#3#4#5
934 {
935   \__draw_point_process:nnn
936   { \__draw_point_intersect_circles_auxi:nnnnnnn {#2} {#4} {#5} }

```

```

937     { \draw_point:n {#1} }
938     { \draw_point:n {#3} }
939   }
940 \cs_new:Npn \__draw_point_intersect_circles_auxi:nnnnnnn #1#2#3#4#5#6#7
941 {
942   \__draw_point_intersect_circles_auxii:eennnnnn
943   { \fp_eval:n {#1} } { \fp_eval:n {#2} } {#4} {#5} {#6} {#7} {#3}
944 }

```

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

```

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

```

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

```

945 \cs_new:Npn \__draw_point_intersect_circles_auxii:nnnnnnn #1#2#3#4#5#6#7
946 {
947   \__draw_point_intersect_circles_auxiii:eennnnnn
948   { \fp_eval:n { #5 - #3 } }
949   { \fp_eval:n { #6 - #4 } }
950   {#1} {#2} {#3} {#4} {#7}
951 }
952 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxii:nnnnnnn { ee }
953 \cs_new:Npn \__draw_point_intersect_circles_auxiii:nnnnnnn #1#2#3#4#5#6#7
954 {
955   \__draw_point_intersect_circles_auxiv:ennnnnnn
956   { \fp_eval:n { sqrt( #1 * #1 + #2 * #2 ) } }
957   {#1} {#2} {#3} {#4} {#5} {#6} {#7}
958 }
959 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxiii:nnnnnnn { ee }

```

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

```

960 \cs_new:Npn \__draw_point_intersect_circles_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
961 {
962   \__draw_point_intersect_circles_auxv:eennnnnnnn
963   { \fp_eval:n { 1 / #1 } }
964   { \fp_eval:n { #4 * #4 } }
965   {#1} {#2} {#3} {#5} {#6} {#7} {#8}
966 }
967 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxiv:nnnnnnnn { e }
968 \cs_new:Npn \__draw_point_intersect_circles_auxv:nnnnnnnnn #1#2#3#4#5#6#7#8#9
969 {
970   \__draw_point_intersect_circles_auxvi:ennnnnnnnn

```

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

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.

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

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

in either

$$P_{1x} = x_1 + \mu_1 \times (x_2 - x_1)$$

$$P_{1y} = y_1 + \mu_1 \times (y_2 - y_1)$$

or

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

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

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

```

990 \cs_new:Npn \draw_point_intersect_line_circle:nnnnn #1#2#3#4#5
991 {
992   \__draw_point_process:nnnn
993   { \__draw_point_intersect_line_circle_auxi:nnnnnnnn {#4} {#5} }
994   { \draw_point:n {#1} }
995   { \draw_point:n {#2} }
996   { \draw_point:n {#3} }
997 }
998 \cs_new:Npn \__draw_point_intersect_line_circle_auxi:nnnnnnnn #1#2#3#4#5#6#7#8
999 {
1000   \__draw_point_intersect_line_circle_auxii:ennnnnnnn
1001   { \fp_eval:n {#1} } {#3} {#4} {#5} {#6} {#7} {#8} {#2}
1002 }

```

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

```

#1 r
#2 x_1
#3 y_1
#4 x_2
#5 y_2
#6 x_3
#7 y_3
#8 n

```

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

```

1003 \cs_new:Npn \__draw_point_intersect_line_circle_auxii:nnnnnnnn #1#2#3#4#5#6#7#8
1004 {
1005   \__draw_point_intersect_line_circle_auxiii:eeennnnnn
1006   { \fp_eval:n { (#4-#2)*(#4-#2)+(#5-#3)*(#5-#3) } }
1007   { \fp_eval:n { 2*((#4-#2)*(#2-#6)+(#5-#3)*(#3-#7)) } }
1008   { \fp_eval:n { (#6*#6+#7*#7)+(#2*#2+#3*#3)-(2*(#6*#2+#7*#3))-(#1*#1) } }
1009   {#2} {#3} {#4} {#5} {#8}
1010 }
1011 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxii:nnnnnnnn { e }

```

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

```

1012 \cs_new:Npn \__draw_point_intersect_line_circle_auxiii:nnnnnnnn #1#2#3#4#5#6#7#8
1013 {
1014   \__draw_point_intersect_line_circle_auxiv:eennnnnnn
1015   { \fp_eval:n { #2 * #2 - 4 * #1 * #3 } }
1016   { \int_if_odd:nTF {#8} { 1 } { -1 } }
1017   {#1} {#2} {#4} {#5} {#6} {#7}
1018 }
1019 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxiii:nnnnnnnn { eee }

```

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

```

#1 a
#2 b
#3 c
#4 d
#5  $\pm$ (the usage of  $n$ )
#6  $x_1$ 
#7  $y_1$ 
#8  $x_2$ 
#9  $y_2$ 

```

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

```

1020 \cs_new:Npn \__draw_point_intersect_line_circle_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
1021 {
1022   \__draw_point_intersect_line_circle_auxv:ennnn
1023   { \fp_eval:n { (-1 * #4 + #2 * sqrt(#1)) / (2 * #3) } }
1024   {#5} {#6} {#7} {#8}
1025 }
1026 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxiv:nnnnnnnn { ee }
1027 \cs_new:Npn \__draw_point_intersect_line_circle_auxv:nnnnn #1#2#3#4#5
1028 {
1029   \draw_point:n
1030   { #2 + #1 * (#4 - #2), #3 + #1 * (#5 - #3) }
1031 }
1032 \cs_generate_variant:Nn \__draw_point_intersect_line_circle_auxv:nnnnn { e }

```

## 5.6 Interpolation on a line (vector) or arc

Simple maths after expansion.

```

\draw_point_interpolate_line:nnn 1033 \cs_new:Npn \draw_point_interpolate_line:nnn #1#2#3
\_draw_point_interpolate_line_aux:nnnnn 1034 {
\_draw_point_interpolate_line_aux:ennnn 1035   \__draw_point_process:nnn
\_draw_point_interpolate_line_aux:nnnnnn 1036   { \__draw_point_interpolate_line_aux:ennnn { \fp_eval:n {#1} } }
\_draw_point_interpolate_line_aux:ennnnn 1037   { \draw_point:n {#2} }
\_draw_point_interpolate_line_aux:ennnnn 1038   { \draw_point:n {#3} }
1039 }

```

```

1040 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnn #1#2#3#4#5
1041 {
1042   \__draw_point_interpolate_line_aux:ennnnn { \fp_eval:n { 1 - #1 } }
1043   {#1} {#2} {#3} {#4} {#5}
1044 }
1045 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnn { e }
1046 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnnn #1#2#3#4#5#6
1047 { \draw_point:n { #1 * #3 + #2 * #5 , #1 * #4 + #2 * #6 } }
1048 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnnn { e }

```

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

```

\draw_point_interpolate_distance:nnn
\_draw_point_interpolate_distance:nnnnn
\_draw_point_interpolate_distance:nnnnnn
\_draw_point_interpolate_distance:ennnnn

```

```

1049 \cs_new:Npn \draw_point_interpolate_distance:nnn #1#2#3
1050 {
1051   \__draw_point_process:nn
1052   { \__draw_point_interpolate_distance:nnnn {#1} {#3} }
1053   {#2}
1054 }
1055 \cs_new:Npn \__draw_point_interpolate_distance:nnnn #1#2#3#4
1056 {
1057   \__draw_point_process:nn
1058   {
1059     \__draw_point_interpolate_distance:ennnn
1060     { \fp_eval:n {#1} } {#3} {#4}
1061   }
1062   { \draw_point_unit_vector:n { ( #2 ) - ( #3 , #4 ) } }
1063 }
1064 \cs_new:Npn \__draw_point_interpolate_distance:nnnnn #1#2#3#4#5
1065 { \draw_point:n { #2 + #1 * #4 , #3 + #1 * #5 } }
1066 \cs_generate_variant:Nn \__draw_point_interpolate_distance:nnnnn { e }

```

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

```

\draw_point_interpolate_arc_axes:nnnnnn
aw_point_interpolate_arcaxes_auxi:nnnnnnnnnn
w_point_interpolate_arcaxes_auxii:nnnnnnnnnn
w_point_interpolate_arcaxes_auxii:ennnnnnnnn
aw_point_interpolate_arcaxes_auxiii:nnnnnnnn
aw_point_interpolate_arcaxes_auxiii:ennnnnnn
aw_point_interpolate_arcaxes_auxiv:nnnnnnnnnn
aw_point_interpolate_arcaxes_auxiv:ennnnnnnnn

```

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

```

1067 \cs_new:Npn \draw_point_interpolate_arc_axes:nnnnnn #1#2#3#4#5#6
1068 {
1069   \__draw_point_process:nnnn
1070   { \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn {#1} {#5} {#6} }
1071   { \draw_point:n {#2} }
1072   { \draw_point:n {#3} }
1073   { \draw_point:n {#4} }
1074 }
1075 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
1076 {
1077   \__draw_point_interpolate_arcaxes_auxii:ennnnnnnnn
1078   { \fp_eval:n {#1} } {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1079 }

```

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

#1  $p$

#2  $\theta_1$

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

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

```

1080 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnn #1#2#3#4#5#6#7#8#9
1081 {
1082   \__draw_point_interpolate_arcaxes_auxiii:ennnnnnn
1083   { \fp_eval:n { #1 * (#3) + ( 1 - #1 ) * (#2) } }
1084   {#4} {#5} {#6} {#7} {#8} {#9}
1085 }
1086 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnn { e }
1087 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnn #1#2#3#4#5#6#7
1088 {
1089   \__draw_point_interpolate_arcaxes_auxiv:eennnnnnn
1090   { \fp_eval:n { cosd (#1) } }
1091   { \fp_eval:n { sind (#1) } }
1092   {#2} {#3} {#4} {#5} {#6} {#7}
1093 }
1094 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnn { e }
1095 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnnn #1#2#3#4#5#6#7#8
1096 {
1097   \draw_point:n
1098   { #3 + #1 * #5 + #2 * #7 , #4 + #1 * #6 + #2 * #8 }
1099 }
1100 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnnn { ee }

```

(End of definition for \draw\_point\_interpolate\_arc\_axes:nnnnnn and others. This function is documented on page ??.)

\draw\_point\_interpolate\_curve:nnnn  
 w\_point\_interpolate\_curve\_quad\_auxi:nnnnnnnn  
 \_point\_interpolate\_curve\_quad\_auxii:nnnnnnnn  
 \_point\_interpolate\_curve\_quad\_auxiii:ennnnnnnn  
 int\_interpolate\_curve\_quad\_auxiiii:nnnnnnnnnn  
 int\_interpolate\_curve\_quad\_auxiiiii:eennnnnnnnn

Interpolation along a quadratic Bézier curve uses the relation

$$P(t) = (1 - t)^2 P_0 + 2(1 - t)tP_1 + t^2 P_2$$

where  $P_0$  is the start point,  $P_1$  is the single control point, and  $P_2$  is the end point. To maximize performance and maintain full expandability, the time parameter  $t$  is evaluated once up front. The three polynomial coefficients are then pre-calculated via an  $e$ -type variant before being mapped against the coordinate components processed by \\_\_draw\_point\_process:nnnn.

```

1101 \cs_new:Npn \draw_point_interpolate_curve:nnnn #1#2#3#4
1102 {
1103   \__draw_point_process:nnnn
1104   { \__draw_point_interpolate_curve_quad_auxi:nnnnnnnn {#1} }
1105   { \draw_point:n {#2} }

```

```

1106     { \draw_point:n {#3} }
1107     { \draw_point:n {#4} }
1108   }
1109   \cs_new:Npn \__draw_point_interpolate_curve_quad_auxi:nnnnnnn #1#2#3#4#5#6#7
1110   {
1111     \__draw_point_interpolate_curve_quad_auxii:ennnnnnn
1112     { \fp_eval:n {#1} }
1113     {#2} {#3} {#4} {#5} {#6} {#7}
1114   }
1115   \cs_new:Npn \__draw_point_interpolate_curve_quad_auxii:nnnnnnn #1#2#3#4#5#6#7
1116   {
1117     \__draw_point_interpolate_curve_quad_auxiii:eeennnnnnn
1118     { \fp_eval:n { (1 - #1)^2 } }
1119     { \fp_eval:n { 2 * (1 - #1) * #1 } }
1120     { \fp_eval:n { #1^2 } }
1121     {#2} {#3} {#4} {#5} {#6} {#7}
1122   }
1123   \cs_generate_variant:Nn \__draw_point_interpolate_curve_quad_auxii:nnnnnnn { e }
1124   \cs_new:Npn \__draw_point_interpolate_curve_quad_auxiii:nnnnnnnnn #1#2#3#4#5#6#7#8#9
1125   {
1126     \draw_point:n
1127     {
1128       #1 * #4 + #2 * #6 + #3 * #8 ,
1129       #1 * #5 + #2 * #7 + #3 * #9
1130     }
1131   }
1132   \cs_generate_variant:Nn \__draw_point_interpolate_curve_quad_auxiii:nnnnnnnnn { eee }

```

(End of definition for \draw\_point\_interpolate\_curve:nnnn 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:ennnnnnnnn
draw_point_interpolate_curve_auxiiii:nnnnnnn
draw_point_interpolate_curve_auxv:ennnnnnn
draw_point_interpolate_curve_auxvi:nnnnnnn
draw_point_interpolate_curve_auxvii:nnnnnnnnn
draw_point_interpolate_curve_auxviii:ennnnnnn

```

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.

```

1133   \cs_new:Npn \draw_point_interpolate_curve:nnnnn #1#2#3#4#5
1134   {
1135     \__draw_point_process:nnnnn
1136     { \__draw_point_interpolate_curve_auxi:nnnnnnnnn {#1} }
1137     { \draw_point:n {#2} }
1138     { \draw_point:n {#3} }
1139     { \draw_point:n {#4} }
1140     { \draw_point:n {#5} }
1141   }
1142   \cs_new:Npn \__draw_point_interpolate_curve_auxi:nnnnnnnnn #1#2#3#4#5#6#7#8#9
1143   {
1144     \__draw_point_interpolate_curve_auxii:ennnnnnnnn
1145     { \fp_eval:n {#1} }
1146     {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1147   }

```



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

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

In the second stage, we can drop the final point

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

and for the final stage only need one set of calculations

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

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

```

1148 \cs_new:Npn \__draw_point_interpolate_curve_auxii:nnnnnnnnn
1149   #1#2#3#4#5#6#7#8#9
1150   {
1151     \__draw_point_interpolate_curve_auxiii:ennnnnn
1152     { \fp_eval:n { 1 - #1 } }
1153     {#1}
1154     { {#2} {#3} } { {#4} {#5} } { {#6} {#7} } { {#8} {#9} }
1155   }
1156 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxii:nnnnnnnnn { e }
1157 % \begin{macrocode}
1158 % We need to do the first cycle, but haven't got enough arguments to keep
1159 % everything in play at once. So here we use a bit of argument re-ordering
1160 % and a single auxiliary to get the job done.
1161 % \begin{macrocode}
1162 \cs_new:Npn \__draw_point_interpolate_curve_auxiii:nnnnnn #1#2#3#4#5#6
1163   {
1164     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #3 #4
1165     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #4 #5
1166     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #5 #6
1167     \prg_do_nothing:
1168     \__draw_point_interpolate_curve_auxvi:n { {#1} {#2} }
1169   }
1170 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxiii:nnnnnn { e }
1171 \cs_new:Npn \__draw_point_interpolate_curve_auxiv:nnnnnn #1#2#3#4#5#6
1172   {
1173     \__draw_point_interpolate_curve_auxv:eev
1174     { \fp_eval:n { #1 * #3 + #2 * #5 } }

```

```

1175     { \fp_eval:n { #1 * #4 + #2 * #6 } }
1176   }
1177   \cs_new:Npn \__draw_point_interpolate_curve_auxv:nnw
1178     #1#2#3 \prg_do_nothing: #4#5
1179   {
1180     #3
1181     \prg_do_nothing:
1182     #4 { #5 {#1} {#2} }
1183   }
1184   \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxv:nnw { ee }
1185   % \begin{macrocode}
1186   % Get the arguments back into the right places and to the second and
1187   % third cycles directly.
1188   % \begin{macrocode}
1189   \cs_new:Npn \__draw_point_interpolate_curve_auxvi:n #1
1190     { \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1 }
1191   \cs_new:Npn \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1#2#3#4#5#6#7#8
1192     {
1193       \__draw_point_interpolate_curve_auxviii:eeeeenn
1194       { \fp_eval:n { #1 * #3 + #2 * #5 } }
1195       { \fp_eval:n { #1 * #4 + #2 * #6 } }
1196       { \fp_eval:n { #1 * #5 + #2 * #7 } }
1197       { \fp_eval:n { #1 * #6 + #2 * #8 } }
1198       {#1} {#2}
1199     }
1200   \cs_new:Npn \__draw_point_interpolate_curve_auxviii:nnnnnn #1#2#3#4#5#6
1201     {
1202       \draw_point:n
1203       { #5 * #1 + #6 * #3 , #5 * #2 + #6 * #4 }
1204     }
1205   \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxviii:nnnnnn { eeee }

```

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

## 5.7 Vector support

As well as coordinates relative to the drawing

```

\l__draw_xvec_x_dim Base vectors to map to the underlying two-dimensional drawing space.
\l__draw_xvec_y_dim 1206 \dim_new:N \l__draw_xvec_x_dim
\l__draw_yvec_x_dim 1207 \dim_new:N \l__draw_xvec_y_dim
\l__draw_yvec_y_dim 1208 \dim_new:N \l__draw_yvec_x_dim
\l__draw_zvec_x_dim 1209 \dim_new:N \l__draw_yvec_y_dim
\l__draw_zvec_y_dim 1210 \dim_new:N \l__draw_zvec_x_dim
1211 \dim_new:N \l__draw_zvec_y_dim

```

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

```

\draw_xvec:n Calculate the underlying position and store it.
\draw_yvec:n 1212 \cs_new_protected:Npn \draw_xvec:n #1
\draw_zvec:n 1213 { \__draw_vec:nn { x } {#1} }
\__draw_vec:nn 1214 \cs_new_protected:Npn \draw_yvec:n #1
\__draw_vec:nnn 1215 { \__draw_vec:nn { y } {#1} }

```

```

1216 \cs_new_protected:Npn \draw_zvec:n #1
1217 { \__draw_vec:nn { z } {#1} }
1218 \cs_new_protected:Npn \__draw_vec:nn #1#2
1219 { \__draw_point_process:nn { \__draw_vec:nnn {#1} } { \draw_point:n {#2} } }
1220 \cs_new_protected:Npn \__draw_vec:nnn #1#2#3
1221 {
1222   \dim_set:cn { l__draw_ #1 vec_x_dim } {#2}
1223   \dim_set:cn { l__draw_ #1 vec_y_dim } {#3}
1224 }

```

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

Initialize the vectors.

```

1225 \draw_xvec:n { 1cm , 0cm }
1226 \draw_yvec:n { 0cm , 1cm }
1227 \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 1228 \cs_new:Npn \draw_point_vec:nn #1#2
\__draw_point_vec:ee 1229 { \__draw_point_vec:ee { \fp_eval:n {#1} } { \fp_eval:n {#2} } }
\draw_point_vec:nnn 1230 \cs_new:Npn \__draw_point_vec:nn #1#2
\__draw_point_vec:nnn 1231 {
\__draw_point_vec:eee 1232   \draw_point:n
1233   {
1234     #1 * \l__draw_xvec_x_dim + #2 * \l__draw_yvec_x_dim ,
1235     #1 * \l__draw_xvec_y_dim + #2 * \l__draw_yvec_y_dim
1236   }
1237 }
1238 \cs_generate_variant:Nn \__draw_point_vec:nn { ee }
1239 \cs_new:Npn \draw_point_vec:nnn #1#2#3
1240 {
1241   \__draw_point_vec:eee
1242   { \fp_eval:n {#1} } { \fp_eval:n {#2} } { \fp_eval:n {#3} }
1243 }
1244 \cs_new:Npn \__draw_point_vec:nnn #1#2#3
1245 {
1246   \draw_point:n
1247   {
1248     #1 * \l__draw_xvec_x_dim
1249     + #2 * \l__draw_yvec_x_dim
1250     + #3 * \l__draw_zvec_x_dim
1251   ,
1252     #1 * \l__draw_xvec_y_dim
1253     + #2 * \l__draw_yvec_y_dim
1254     + #3 * \l__draw_zvec_y_dim
1255   }
1256 }
1257 \cs_generate_variant:Nn \__draw_point_vec:nnn { eee }

```

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

\draw\_point\_vec\_polar:nn Much the same as the core polar approach.

```

\draw_point_vec_polar:nnn 1258 \cs_new:Npn \draw_point_vec_polar:nn #1#2
\__draw_point_vec_polar:nnn 1259 { \draw_point_vec_polar:nnn {#1} {#1} {#2} }
\__draw_point_vec_polar:enn 1260 \cs_new:Npn \draw_point_vec_polar:nnn #1#2#3
\__draw_point_vec_polar:nnnn
\__draw_point_vec_polar:eeenn

```

```

1261 { \__draw_point_vec_polar:enn { \fp_eval:n {#3} } {#1} {#2} }
1262 \cs_new:Npn \__draw_point_vec_polar:nnn #1#2#3
1263 {
1264   \__draw_point_vec_polar:een
1265   { \fp_eval:n { sind(#1) } }
1266   { \fp_eval:n { cosd(#1) } }
1267   {#2} {#3}
1268 }
1269 \cs_generate_variant:Nn \__draw_point_vec_polar:nnn { e }
1270 \cs_new:Npn \__draw_point_vec_polar:nnnn #1#2#3#4
1271 {
1272   \draw_point:n
1273   {
1274     #2 * (#3) * \l__draw_xvec_x_dim
1275     + #1 * (#4) * \l__draw_yvec_x_dim
1276     ,
1277     #2 * (#3) * \l__draw_xvec_y_dim
1278     + #1 * (#4) * \l__draw_yvec_y_dim
1279   }
1280 }
1281 \cs_generate_variant:Nn \__draw_point_vec_polar:nnnn { ee }

```

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

## 5.8 Transformations

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

```

1282 \cs_new:Npn \draw_point_transform:n #1
1283 {
1284   \__draw_point_process:nn
1285   { \__draw_point_transform:nn } { \draw_point:n {#1} }
1286 }
1287 \cs_new:Npn \__draw_point_transform:nn #1#2
1288 {
1289   \bool_if:NTF \l__draw_matrix_active_bool
1290   {
1291     \draw_point:n
1292     {
1293       (
1294         \l__draw_matrix_a_fp * #1
1295         + \l__draw_matrix_c_fp * #2
1296         + \l__draw_xshift_dim
1297       )
1298       ,
1299       (
1300         \l__draw_matrix_b_fp * #1
1301         + \l__draw_matrix_d_fp * #2
1302         + \l__draw_yshift_dim
1303       )
1304     }
1305   }
1306 }

```

```

1307     \draw_point:n
1308     {
1309         (#1, #2)
1310         + ( \l__draw_xshift_dim , \l__draw_yshift_dim )
1311     }
1312 }
1313 }

```

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

`\__draw_point_transform_noshift:n`  
`\__draw_point_transform_noshift:nn`

A version with no shift: used for internal purposes.

```

1314 \cs_new:Npn \__draw_point_transform_noshift:n #1
1315 {
1316     \__draw_point_process:nn
1317     { \__draw_point_transform_noshift:nn }
1318     { \draw_point:n {#1} }
1319 }
1320 \cs_new:Npn \__draw_point_transform_noshift:nn #1#2
1321 {
1322     \bool_if:NTF \l__draw_matrix_active_bool
1323     {
1324         \draw_point:n
1325         {
1326             (
1327                 \l__draw_matrix_a_fp * #1
1328                 + \l__draw_matrix_c_fp * #2
1329             )
1330             ,
1331             (
1332                 \l__draw_matrix_b_fp * #1
1333                 + \l__draw_matrix_d_fp * #2
1334             )
1335         }
1336     }
1337     { \draw_point:n { (#1, #2) } }
1338 }

```

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

```

1339 \endpackage

```

## 6 l3draw-scopes implementation

```

1340 \begin{package}
1341 \begin{@@=draw}

```

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

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

## 6.1 Drawing environment

<code>\g_draw_bb_xmax_dim</code>	Used to track the overall (official) size of the image created: may not actually be the natural size of the content.
<code>\g_draw_bb_xmin_dim</code>	
<code>\g_draw_bb_ymax_dim</code>	1342 <code>\dim_new:N \g_draw_bb_xmax_dim</code>
<code>\g_draw_bb_ymin_dim</code>	1343 <code>\dim_new:N \g_draw_bb_xmin_dim</code>
	1344 <code>\dim_new:N \g_draw_bb_ymax_dim</code>
	1345 <code>\dim_new:N \g_draw_bb_ymin_dim</code>
	(End of definition for <code>\g_draw_bb_xmax_dim</code> and others. These variables are documented on page ??.)
<code>\l_draw_bb_update_bool</code>	Flag to indicate that a path (or similar) should update the bounding box of the drawing.
	1346 <code>\bool_new:N \l_draw_bb_update_bool</code>
	(End of definition for <code>\l_draw_bb_update_bool</code> . This variable is documented on page ??.)
<code>\l__draw_layer_main_box</code>	Box for setting the drawing itself and the top-level layer.
	1347 <code>\box_new:N \l__draw_main_box</code>
	1348 <code>\box_new:N \l__draw_layer_main_box</code>
	(End of definition for <code>\l__draw_layer_main_box</code> .)
<code>\g_draw_id_int</code>	The drawing number.
	1349 <code>\int_new:N \g_draw_id_int</code>
	(End of definition for <code>\g_draw_id_int</code> . This variable is documented on page ??.)
<code>\__draw_reset_bb:</code>	A simple auxiliary.
	1350 <code>\cs_new_protected:Npn \__draw_reset_bb:</code>
	1351 <code>{</code>
	1352 <code>\dim_gset:Nn \g_draw_bb_xmax_dim { -\c_max_dim }</code>
	1353 <code>\dim_gset:Nn \g_draw_bb_xmin_dim { \c_max_dim }</code>
	1354 <code>\dim_gset:Nn \g_draw_bb_ymax_dim { -\c_max_dim }</code>
	1355 <code>\dim_gset:Nn \g_draw_bb_ymin_dim { \c_max_dim }</code>
	1356 <code>}</code>
	(End of definition for <code>\__draw_reset_bb:</code> .)
<code>\draw_begin:</code>	Drawings are created by setting them into a box, then adjusting the box before inserting
<code>\draw_end:</code>	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 <code>hbox</code> . To allow for layers, there is some box nesting: notice that we
	1357 <code>\cs_new_protected:Npn \draw_begin:</code>
	1358 <code>{</code>
	1359 <code>\group_begin:</code>
	1360 <code>\int_gincr:N \g_draw_id_int</code>
	1361 <code>\hbox_set:Nw \l__draw_main_box</code>
	1362 <code>\__draw_backend_begin:</code>
	1363 <code>\__draw_reset_bb:</code>
	1364 <code>\__draw_path_reset_limits:</code>
	1365 <code>\bool_set_true:N \l_draw_bb_update_bool</code>
	1366 <code>\draw_transform_matrix_reset:</code>
	1367 <code>\draw_transform_shift_reset:</code>

```

1368     \__draw_software_clear:
1369     \draw_set_linewidth:n { \l_draw_default_linewidth_dim }
1370     \color_ensure_current:
1371     \draw_set_nonzero_rule:
1372     \draw_set_cap_but:
1373     \draw_set_join_miter:
1374     \draw_set_miterlimit:n { 10 }
1375     \draw_set_dash_pattern:nn { } { 0cm }
1376     \hbox_set:Nw \l__draw_layer_main_box
1377     \__draw_record_origin:
1378 }
1379 \cs_new_protected:Npn \draw_end:
1380 {
1381     \__draw_baseline_finalize:w
1382     \exp_args:NNNV \hbox_set_end:
1383     \clist_set:Nn \l_draw_layers_clist \l_draw_layers_clist
1384     \__draw_layers_insert:
1385     \__draw_backend_end:
1386     \hbox_set_end:
1387     \dim_compare:nNnT \g_draw_bb_xmin_dim = \c_max_dim
1388     {
1389         \dim_gzero:N \g_draw_bb_xmax_dim
1390         \dim_gzero:N \g_draw_bb_xmin_dim
1391         \dim_gzero:N \g_draw_bb_ymax_dim
1392         \dim_gzero:N \g_draw_bb_ymin_dim
1393     }
1394     \__draw_finalize:
1395     \box_set_wd:Nn \l__draw_main_box
1396     { \g_draw_bb_xmax_dim - \g_draw_bb_xmin_dim }
1397     \mode_leave_vertical:
1398     \box_use_drop:N \l__draw_main_box
1399 \group_end:
1400 }

```

(End of definition for \draw\_begin: and \draw\_end:. These functions are documented on page ??.)

`\__draw_record_origin:` Used to log the absolute location of a drawing. Ideally this would not need two `\savepos:` we need to sort an “always left-to-right” box. At present, this functionality is only available in L<sup>A</sup>T<sub>E</sub>X.

```

1401 \cs_new_protected:Npe \__draw_record_origin:
1402 {
1403     \hbox_to_wd:nn { Opt }
1404     {
1405         \tex_savepos:D
1406         \cs_if_exist:NT \@expl@finalise@setup@@
1407         {
1408             \exp_not:N \property_record:en
1409             { draw . \exp_not:N \int_use:N \exp_not:N \g_draw_id_int }
1410             { xpos , ypos , abspage }
1411         }
1412         \tex_savepos:D
1413     }
1414 }
1415 \cs_generate_variant:Nn \property_record:nn { e }

```

(End of definition for `\__draw_record_origin:`.)

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

```

1416 \cs_new_protected:Npn \__draw_finalize:
1417 {
1418   \hbox_set:Nn \l__draw_main_box
1419   {
1420     \skip_horizontal:n { -\g_draw_bb_xmin_dim }
1421     \box_move_down:nn
1422     { \g_draw_bb_ymin_dim }
1423     { \box_use_drop:N \l__draw_main_box }
1424   }
1425   \box_set_dp:Nn \l__draw_main_box { Opt }
1426   \box_set_ht:Nn \l__draw_main_box
1427   { \g_draw_bb_ymax_dim - \g_draw_bb_ymin_dim }
1428 }
1429 \cs_new_protected:Npn \__draw_finalize_baseline:n #1
1430 {
1431   \hbox_set:Nn \l__draw_main_box
1432   {
1433     \skip_horizontal:n { -\g_draw_bb_xmin_dim }
1434     \box_move_down:nn
1435     { #1 }
1436     { \box_use_drop:N \l__draw_main_box }
1437   }
1438   \box_set_dp:Nn \l__draw_main_box
1439   {
1440     \dim_max:nn
1441     { #1 - \g_draw_bb_ymin_dim }
1442     { Opt }
1443   }
1444   \box_set_ht:Nn \l__draw_main_box
1445   { \g_draw_bb_ymax_dim - #1 }
1446 }

```

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

## 6.2 Baseline position

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

```

\l__draw_baseline_dim
1447 \bool_new:N \l__draw_baseline_bool
1448 \dim_new:N \l__draw_baseline_dim

```

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

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

```

1449 \cs_new_protected:Npn \draw_set_baseline:n #1
1450 {
1451   \bool_set_true:N \l__draw_baseline_bool
1452   \dim_set:Nn \l__draw_baseline_dim { \fp_to_dim:n { #1 } }
1453 }

```



(End of definition for \draw\_set\_baseline:n. This function is documented on page ??.)

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

```

1454 \cs_new_protected:Npn \_draw\_baseline\_finalize:w #1 \_draw\_finalize:
1455 {
1456   \bool_if:NTF \l\_draw\_baseline\_bool
1457   {
1458     \use:e
1459     {
1460       \exp_not:n {#1}
1461       \_draw\_finalize\_baseline:n { \dim\_use:N \l\_draw\_baseline\_dim }
1462     }
1463   }
1464   { #1 \_draw\_finalize: }
1465 }

```

(End of definition for \\_draw\\_baseline\\_finalize:w.)

### 6.3 Scopes

\l\\_draw\\_linewidth\\_dim Storage for local variables.

```

1466 \dim_new:N \l\_draw\_linewidth\_dim

```

(End of definition for \l\\_draw\\_linewidth\\_dim.)

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

```

\draw\_scope\_begin: 1467 \cs_new_protected:Npn \draw\_scope\_begin:
1468 {
1469   \_draw\_backend\_scope\_begin:
1470   \group\_begin:
1471     \dim\_set\_eq:NN \l\_draw\_linewidth\_dim \g\_draw\_linewidth\_dim
1472     \draw\_path\_scope\_begin:
1473   }
1474 \cs_new_protected:Npn \draw\_scope\_end:
1475 {
1476   \draw\_path\_scope\_end:
1477   \dim\_gset\_eq:NN \g\_draw\_linewidth\_dim \l\_draw\_linewidth\_dim
1478   \group\_end:
1479   \_draw\_backend\_scope\_end:
1480 }

```

(End of definition for \draw\\_scope\\_begin:. This function is documented on page ??.)

\l\\_draw\\_xmax\\_dim Storage for the bounding box.

```

\l\_draw\_xmin\_dim 1481 \dim_new:N \l\_draw\_xmax\_dim
\l\_draw\_ymax\_dim 1482 \dim_new:N \l\_draw\_xmin\_dim
\l\_draw\_ymin\_dim 1483 \dim_new:N \l\_draw\_ymax\_dim
1484 \dim_new:N \l\_draw\_ymin\_dim

```

(End of definition for \l\\_draw\\_xmax\\_dim and others.)

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

```

\__draw_scope_bb_end:
1485 \cs_new_protected:Npn \__draw_scope_bb_begin:
1486 {
1487   \group_begin:
1488     \dim_set_eq:NN \l__draw_xmax_dim \g_draw_bb_xmax_dim
1489     \dim_set_eq:NN \l__draw_xmin_dim \g_draw_bb_xmin_dim
1490     \dim_set_eq:NN \l__draw_ymax_dim \g_draw_bb_ymax_dim
1491     \dim_set_eq:NN \l__draw_ymin_dim \g_draw_bb_ymin_dim
1492     \__draw_reset_bb:
1493   }
1494 \cs_new_protected:Npn \__draw_scope_bb_end:
1495 {
1496   \dim_gset_eq:NN \g_draw_bb_xmax_dim \l__draw_xmax_dim
1497   \dim_gset_eq:NN \g_draw_bb_xmin_dim \l__draw_xmin_dim
1498   \dim_gset_eq:NN \g_draw_bb_ymax_dim \l__draw_ymax_dim
1499   \dim_gset_eq:NN \g_draw_bb_ymin_dim \l__draw_ymin_dim
1500   \group_end:
1501 }

```

*(End of definition for \\_\_draw\_scope\_bb\_begin: and \\_\_draw\_scope\_bb\_end:.)*

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

```

\draw_suspend_end:
1502 \cs_new_protected:Npn \draw_suspend_begin:
1503 {
1504   \__draw_scope_bb_begin:
1505   \draw_path_scope_begin:
1506   \draw_transform_matrix_reset:
1507   \draw_transform_shift_reset:
1508   \__draw_layers_save:
1509 }
1510 \cs_new_protected:Npn \draw_suspend_end:
1511 {
1512   \__draw_layers_restore:
1513   \draw_path_scope_end:
1514   \__draw_scope_bb_end:
1515 }

```

*(End of definition for \draw\_suspend\_begin: and \draw\_suspend\_end:.. These functions are documented on page ??.)*

```

1516 </package>

```

## 7 l3draw-softpath implementation

```

1517 <*package>
1518 <@@=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.
1519 \tl_new:N \g__draw_softpath_main_tl
(End of definition for \g__draw_softpath_main_tl.)

\l__draw_softpath_tmp_tl Scratch space.
1520 \tl_new:N \l__draw_softpath_tmp_tl
(End of definition for \l__draw_softpath_tmp_tl.)

\g__draw_softpath_corners_bool Allow for optimized path use.
1521 \bool_new:N \g__draw_softpath_corners_bool
(End of definition for \g__draw_softpath_corners_bool.)

\__draw_softpath_add:n
\__draw_softpath_add:o 1522 \cs_new_protected:Npn \__draw_softpath_add:n
\__draw_softpath_add:e 1523 { \tl_build_gput_right:Nn \g__draw_softpath_main_tl }
1524 \cs_generate_variant:Nn \__draw_softpath_add:n { o, e }
(End of definition for \__draw_softpath_add:n.)

\__draw_softpath_use: Using and clearing is trivial.
\__draw_softpath_clear: 1525 \cs_new_protected:Npn \__draw_softpath_use:
1526 {
1527   \tl_build_get_intermediate:NN
1528   \g__draw_softpath_main_tl
1529   \l__draw_softpath_tmp_tl
1530   \l__draw_softpath_tmp_tl
1531 }
1532 \cs_new_protected:Npn \__draw_softpath_clear:
1533 {
1534   \tl_build_gbegin:N \g__draw_softpath_main_tl
1535   \bool_gset_false:N \g__draw_softpath_corners_bool
1536 }
(End of definition for \__draw_softpath_use: and \__draw_softpath_clear:.)

\__draw_softpath_save: Abstracted ideas to keep variables inside this submodule.
\__draw_softpath_restore: 1537 \cs_new_protected:Npn \__draw_softpath_save:
1538 {
1539   \tl_build_gend:N \g__draw_softpath_main_tl
1540   \tl_set_eq:NN
1541   \l__draw_softpath_main_tl
1542   \g__draw_softpath_main_tl
1543   \bool_set_eq:NN
1544   \l__draw_softpath_corners_bool
1545   \g__draw_softpath_corners_bool

```

```

1546     \__draw_softpath_clear:
1547   }
1548   \cs_new_protected:Npn \__draw_softpath_restore:
1549   {
1550     \__draw_softpath_clear:
1551     \__draw_softpath_add:o \l__draw_softpath_main_tl
1552     \bool_gset_eq:NN
1553       \g__draw_softpath_corners_bool
1554       \l__draw_softpath_corners_bool
1555   }

```

(End of definition for \\_\_draw\_softpath\_save: and \\_\_draw\_softpath\_restore:.)

\g\_\_draw\_softpath\_lastx\_dim For tracking the end of the path (to close it).

```

\g__draw_softpath_lasty_dim 1556 \dim_new:N \g__draw_softpath_lastx_dim
1557 \dim_new:N \g__draw_softpath_lasty_dim

```

(End of definition for \g\_\_draw\_softpath\_lastx\_dim and \g\_\_draw\_softpath\_lasty\_dim.)

\g\_\_draw\_softpath\_move\_bool Track if moving a point should update the close position.

```

1558 \bool_new:N \g__draw_softpath_move_bool
1559 \bool_gset_true:N \g__draw_softpath_move_bool

```

(End of definition for \g\_\_draw\_softpath\_move\_bool.)

\\_\_draw\_softpath\_closepath: The various parts of a path expressed as the appropriate soft path functions.

```

\__draw_softpath_curveto:nnnnnn 1560 \cs_new_protected:Npn \__draw_softpath_closepath:
\__draw_softpath_lineto:nn 1561 {
\__draw_softpath_moveto:nn 1562   \__draw_softpath_add:e
\__draw_softpath_rectangle:nnnn 1563   {
\__draw_softpath_roundpoint:nn 1564     \__draw_softpath_close_op:nn
\__draw_softpath_roundpoint:VV 1565     { \dim_use:N \g__draw_softpath_lastx_dim }
1566     { \dim_use:N \g__draw_softpath_lasty_dim }
1567   }
1568 }
1569 \cs_new_protected:Npn \__draw_softpath_curveto:nnnnnn #1#2#3#4#5#6
1570 {
1571   \__draw_softpath_add:n
1572   {
1573     \__draw_softpath_curveto_opi:nn {#1} {#2}
1574     \__draw_softpath_curveto_opii:nn {#3} {#4}
1575     \__draw_softpath_curveto_opiii:nn {#5} {#6}
1576   }
1577 }
1578 \cs_new_protected:Npn \__draw_softpath_lineto:nn #1#2
1579 {
1580   \__draw_softpath_add:n
1581   { \__draw_softpath_lineto_op:nn {#1} {#2} }
1582 }
1583 \cs_new_protected:Npn \__draw_softpath_moveto:nn #1#2
1584 {
1585   \__draw_softpath_add:n
1586   { \__draw_softpath_moveto_op:nn {#1} {#2} }
1587   \bool_if:NT \g__draw_softpath_move_bool
1588   {

```

```

1589         \dim_gset:Nn \g__draw_softpath_lastx_dim {#1}
1590         \dim_gset:Nn \g__draw_softpath_lasty_dim {#2}
1591     }
1592 }
1593 \cs_new_protected:Npn \__draw_softpath_rectangle:nnnn #1#2#3#4
1594 {
1595     \__draw_softpath_add:n
1596     {
1597         \__draw_softpath_rectangle_opi:nn {#1} {#2}
1598         \__draw_softpath_rectangle_opii:nn {#3} {#4}
1599     }
1600 }
1601 \cs_new_protected:Npn \__draw_softpath_roundpoint:nn #1#2
1602 {
1603     \__draw_softpath_add:n
1604     { \__draw_softpath_roundpoint_op:nn {#1} {#2} }
1605     \bool_gset_true:N \g__draw_softpath_corners_bool
1606 }
1607 \cs_generate_variant:Nn \__draw_softpath_roundpoint:nn { VV }

```

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

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

```

1608 \cs_new_protected:Npn \__draw_softpath_close_op:nn #1#2
1609 { \__draw_backend_closepath: }
1610 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nn #1#2
1611 { \__draw_softpath_curveto_opi:nnNnnNnn {#1} {#2} }
1612 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nnNnnNnn #1#2#3#4#5#6#7#8
1613 { \__draw_backend_curveto:nnnnnn {#1} {#2} {#4} {#5} {#7} {#8} }
1614 \cs_new_protected:Npn \__draw_softpath_curveto_opii:nn #1#2
1615 { \__draw_softpath_curveto_opii:nn }
1616 \cs_new_protected:Npn \__draw_softpath_curveto_opiii:nn #1#2
1617 { \__draw_softpath_curveto_opiii:nn }
1618 \cs_new_protected:Npn \__draw_softpath_lineto_op:nn #1#2
1619 { \__draw_backend_lineto:nn {#1} {#2} }
1620 \cs_new_protected:Npn \__draw_softpath_moveto_op:nn #1#2
1621 { \__draw_backend_moveto:nn {#1} {#2} }
1622 \cs_new_protected:Npn \__draw_softpath_roundpoint_op:nn #1#2
1623 { \__draw_softpath_roundpoint_op:nn }
1624 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nn #1#2
1625 { \__draw_softpath_rectangle_opi:nnNnn {#1} {#2} }
1626 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nnNnn #1#2#3#4#5
1627 { \__draw_backend_rectangle:nnnn {#1} {#2} {#4} {#5} }
1628 \cs_new_protected:Npn \__draw_softpath_rectangle_opii:nn #1#2
1629 { \__draw_softpath_rectangle_opii:nn }

```

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

## 7.2 Rounding soft path corners

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

<code>\l__draw_softpath_main_tl</code>	For constructing the updated path.
--	------------------------------------

```
1630 \tl_new:N \l__draw_softpath_main_tl
```

(End of definition for \l\_\_draw\_softpath\_main\_tl.)

\l\_\_draw\_softpath\_part\_tl Data structures.

```
1631 \tl_new:N \l__draw_softpath_part_tl
```

```
1632 \tl_new:N \l__draw_softpath_curve_end_tl
```

(End of definition for \l\_\_draw\_softpath\_part\_tl.)

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

```
\l__draw_softpath_lasty_fp 1633 \fp_new:N \l__draw_softpath_lastx_fp
```

```
\l__draw_softpath_corneri_dim 1634 \fp_new:N \l__draw_softpath_lasty_fp
```

```
\l__draw_softpath_cornerii_dim 1635 \dim_new:N \l__draw_softpath_corneri_dim
```

```
\l__draw_softpath_first_tl 1636 \dim_new:N \l__draw_softpath_cornerii_dim
```

```
\l__draw_softpath_move_tl 1637 \tl_new:N \l__draw_softpath_first_tl
```

```

1638 \tl_new:N \l__draw_softpath_move_tl

```

(End of definition for \l\_\_draw\_softpath\_lastx\_fp and others.)

<code>\c__draw_softpath_arc_fp</code>	The magic constant.
---------------------------------------	---------------------

```
1639 \fp_const:Nn \c__draw_softpath_arc_fp { 4/3 * (sqrt(2) - 1) }
```

(End of definition for \c\_\_draw\_softpath\_arc\_fp.)

<code>\_draw_softpath_round_corners:</code>	Rounding corners on a path means going through the entire path and adjusting it. As
<code>\_draw_softpath_round_loop:nnn</code>	such, we avoid this entirely if we know there are no corners to deal with. Assuming there
<code>\_draw_softpath_round_action:nn</code>	is work to do, we recover the existing path and start a loop.

```
\__draw_softpath_round_action:Nnn 1640 \cs_new_protected:Npn \__draw_softpath_round_corners:
```

```

--draw_softpath_round_action_curve: NnnNnn      1641      {
    \draw softpath round action close:           1642      \bool_if:NT \g__draw_softpath_corners_bool

```

```
\draw softpath round lookahead:NnnNnn 1643 {
```

```
\draw softpath round roundpoint:NnnNnnNnn 1644 \group_begin:
```

```

\draw softpath round calc:NnnNnn 1645 \tl_clear:N \l__draw_softpath_main_tl

```

```
\draw softpath round calc:nnnnnn 1646 \tl_clear:N \l__draw_softpath_part_tl
```

```

\draw softpath round calc:(\nnnnn
1647
\fp_zero:N \l__draw_softpath_lastx_fp

```

```

\fp_zero:N \l__draw_softpath_lasty_fp
\draw_softpath_round calc:nnnnu

```

```

\__draw_softpath_round_close:nnnnnnw 1649 \tl_clear:N \l__draw_softpath_first_t
\__draw_softpath_round_close:nn

```

```

\__draw_softpath_round_close.mml      1650      \tl_clear:N \l__draw_softpath_move_tl
\__draw_softpath_round_close.mml

```

```
\draw softpath round close:w      1651      \tl_build_gend:N \g__draw_softpath_main_tl
```

```

__draw_softpath_round_end:      1652      \exp_after:wN \__draw_softp

```

```
1653         \g__draw_softpath_main_tl
```

```
1654 \q__draw_recursion_tail
```

1655 \q\_{--dra}

```

1656                                     \group_end:

```

1657

```
1658         \bool_gset_false:N \g__draw_softpath_corners_bool
```

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.

```

1660 \cs_new_protected:Npn \__draw_softpath_round_loop:Nnn #1#2#3
1661 {
1662   \__draw_if_recursion_tail_stop_do:Nn #1 { \__draw_softpath_round_end: }
1663   \token_if_eq_meaning:NNTF #1 \__draw_softpath_roundpoint_op:nn
1664   { \__draw_softpath_round_action:nn {#2} {#3} }
1665   {
1666     \tl_if_empty:NT \l__draw_softpath_first_tl
1667     { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1668     \fp_set:Nn \l__draw_softpath_lastx_fp {#2}
1669     \fp_set:Nn \l__draw_softpath_lasty_fp {#3}
1670     \token_if_eq_meaning:NNTF #1 \__draw_softpath_moveto_op:nn
1671     {
1672       \tl_put_right:No \l__draw_softpath_main_tl
1673       \l__draw_softpath_move_tl
1674       \tl_put_right:No \l__draw_softpath_main_tl
1675       \l__draw_softpath_part_tl
1676       \tl_set:Nn \l__draw_softpath_move_tl { #1 {#2} {#3} }
1677       \tl_clear:N \l__draw_softpath_first_tl
1678       \tl_clear:N \l__draw_softpath_part_tl
1679     }
1680     { \tl_put_right:Nn \l__draw_softpath_part_tl { #1 {#2} {#3} } }
1681     \__draw_softpath_round_loop:Nnn
1682   }
1683 }
1684 \cs_new_protected:Npn \__draw_softpath_round_action:nn #1#2
1685 {
1686   \dim_set:Nn \l__draw_softpath_corneri_dim {#1}
1687   \dim_set:Nn \l__draw_softpath_cornerii_dim {#2}
1688   \bool_lazy_and:nnTF
1689   { \dim_compare_p:nNn \l__draw_softpath_corneri_dim = { Opt } }
1690   { \dim_compare_p:nNn \l__draw_softpath_cornerii_dim = { Opt } }
1691   { \__draw_softpath_round_loop:Nnn }
1692   { \__draw_softpath_round_action:Nnn }
1693 }

```

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.

```

1694 \cs_new_protected:Npn \__draw_softpath_round_action:Nnn #1#2#3
1695 {
1696   \tl_if_empty:NT \l__draw_softpath_first_tl
1697   { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1698   \token_if_eq_meaning:NNTF #1 \__draw_softpath_curveto_opi:nn
1699   { \__draw_softpath_round_action_curveto:NnnNnn }
1700   {
1701     \token_if_eq_meaning:NNTF #1 \__draw_softpath_close_op:nn
1702     { \__draw_softpath_round_action_close: }
1703     {
1704       \token_if_eq_meaning:NNTF #1 \__draw_softpath_lineto_op:nn

```

```

1705         { \__draw_softpath_round_lookahead:NnnNnn }
1706         { \__draw_softpath_round_loop:Nnn }
1707     }
1708 }
1709 #1 {#2} {#3}
1710 }

```

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.

```

1711 \cs_new_protected:Npn \__draw_softpath_round_action_curveto:NnnNnn
1712 #1#2#3#4#5#6
1713 {
1714   \tl_put_right:Nn \l__draw_softpath_part_tl
1715     { #1 {#2} {#3} #4 {#5} {#6} }
1716   \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1717   \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1718   \__draw_softpath_round_lookahead:NnnNnn
1719 }
1720 \cs_new_protected:Npn \__draw_softpath_round_action_close:
1721 {
1722   \bool_lazy_and:nnTF
1723     { ! \tl_if_empty_p:N \l__draw_softpath_first_tl }
1724     { ! \tl_if_empty_p:N \l__draw_softpath_move_tl }
1725   {
1726     \exp_after:wN \__draw_softpath_round_close:nn
1727       \l__draw_softpath_first_tl
1728   }
1729   { \__draw_softpath_round_loop:Nnn }
1730 }

```

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.

```

1731 \cs_new_protected:Npn \__draw_softpath_round_lookahead:NnnNnn #1#2#3#4#5#6
1732 {
1733   \bool_lazy_any:nTF
1734     {
1735       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_lineto_op:nn }
1736       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_curveto_opi:nn }
1737       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_close_op:nn }
1738     }
1739     {
1740       \__draw_softpath_round_calc:NnnNnn
1741       \__draw_softpath_round_loop:Nnn
1742       {#5} {#6}
1743     }
1744     {
1745       \token_if_eq_meaning:NNTF #4 \__draw_softpath_roundpoint_op:nn
1746       { \__draw_softpath_round_roundpoint:NnnNnnNnn }
1747       { \__draw_softpath_round_loop:Nnn }
1748     }
1749     #1 {#2} {#3}
1750     #4 {#5} {#6}
1751 }
1752 \cs_new_protected:Npn \__draw_softpath_round_roundpoint:NnnNnnNnn

```



```

1753 #1#2#3#4#5#6#7#8#9
1754 {
1755     \__draw_softpath_round_calc:NnnNnn
1756     \__draw_softpath_round_loop:Nnn
1757     {#8} {#9}
1758     #1 {#2} {#3}
1759     #4 {#5} {#6} #7 {#8} {#9}
1760 }

```

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

```

1761 \cs_new_protected:Npn \__draw_softpath_round_calc:NnnNnn #1#2#3#4#5#6
1762 {
1763     \tl_set:Nx \l__draw_softpath_curve_end_tl
1764     {
1765         \draw_point_interpolate_distance:nnn
1766         \l__draw_softpath_cornerii_dim
1767         { #5 , #6 } { #2 , #3 }
1768     }
1769     \tl_put_right:Nx \l__draw_softpath_part_tl
1770     {
1771         \exp_not:N #4
1772         \__draw_softpath_round_calc:eVnnnn
1773         {
1774             \draw_point_interpolate_distance:nnn
1775             \l__draw_softpath_corneri_dim
1776             { #5 , #6 }
1777             {
1778                 \l__draw_softpath_lastx_fp ,
1779                 \l__draw_softpath_lasty_fp
1780             }
1781         }
1782         \l__draw_softpath_curve_end_tl
1783         {#5} {#6} {#2} {#3}
1784     }
1785     \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1786     \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1787     #1
1788 }

```

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

```

1789 \cs_new:Npn \__draw_softpath_round_calc:nnnnnn #1#2#3#4#5#6
1790 {
1791     \__draw_softpath_round_calc:nnnnw {#3} {#4} {#5} {#6}
1792     #1 \s__draw_mark #2 \s__draw_stop
1793 }
1794 \cs_generate_variant:Nn \__draw_softpath_round_calc:nnnnnn { eV }

```

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

```

1795 \cs_new:Npn \__draw_softpath_round_calc:nnnnw
1796   #1#2#3#4 #5 , #6 \s__draw_mark #7 , #8 \s__draw_stop
1797   {
1798     {#5} {#6}
1799     \exp_not:N \__draw_softpath_curveto_opi:nn
1800     {
1801       \fp_to_dim:n
1802       { #5 + \c__draw_softpath_arc_fp * ( #1 - #5 ) }
1803     }
1804     {
1805       \fp_to_dim:n
1806       { #6 + \c__draw_softpath_arc_fp * ( #2 - #6 ) }
1807     }
1808     \exp_not:N \__draw_softpath_curveto_opii:nn
1809     {
1810       \fp_to_dim:n
1811       { #7 + \c__draw_softpath_arc_fp * ( #1 - #7 ) }
1812     }
1813     {
1814       \fp_to_dim:n
1815       { #8 + \c__draw_softpath_arc_fp * ( #2 - #8 ) }
1816     }
1817     \exp_not:N \__draw_softpath_curveto_opiii:nn
1818     {#7} {#8}
1819   }

```

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.

```

1820 \cs_new_protected:Npn \__draw_softpath_round_close:nn #1#2
1821   {
1822     \use:e
1823     {
1824       \__draw_softpath_round_calc:NnnNnn
1825       {
1826         \tl_set:Nc \exp_not:N \l__draw_softpath_move_tl
1827         {
1828           \__draw_softpath_moveto_op:nn
1829           \exp_not:N \exp_after:wN
1830           \exp_not:N \__draw_softpath_round_close:w
1831           \exp_not:N \l__draw_softpath_curve_end_tl
1832           \s__draw_stop
1833         }
1834       }
1835       \use:e
1836       {
1837         \exp_not:N \exp_not:N \exp_not:N \use_i:nnnn
1838         {
1839           \__draw_softpath_round_loop:Nnn
1840           \__draw_softpath_close_op:nn
1841           \exp_not:N \exp_after:wN
1842           \exp_not:N \__draw_softpath_round_close:w

```

```

1842 \exp_not:N \l__draw_softpath_curve_end_tl
1843 \s__draw_stop
1844 }
1845 }
1846 }
1847 {#1} {#2}
1848 \__draw_softpath_lineto_op:nn
1849 \exp_after:wN \use_none:n \l__draw_softpath_move_tl
1850 }
1851 }
1852 \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.

```

1853 \cs_new_protected:Npn \__draw_softpath_round_end:
1854 {
1855   \tl_put_right:No \l__draw_softpath_main_tl
1856   \l__draw_softpath_move_tl
1857   \tl_put_right:No \l__draw_softpath_main_tl
1858   \l__draw_softpath_part_tl
1859   \tl_build_gbegin:N \g__draw_softpath_main_tl
1860   \__draw_softpath_add:o \l__draw_softpath_main_tl
1861 }

```

*(End of definition for \\_\_draw\_softpath\_round\_corners: and others.)*

```

1862 </package>

```

## 8 l3draw-state implementation

```

1863 <*package>
1864 <@@=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.

```

1865 \dim_new:N \g__draw_linewidth_dim

```

*(End of definition for \g\_\_draw\_linewidth\_dim.)*

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

```

1866 \dim_new:N \l_draw_default_linewidth_dim
1867 \dim_set:Nn \l_draw_default_linewidth_dim { 0.4pt }

```

*(End of definition for \l\_draw\_default\_linewidth\_dim. This variable is documented on page ??.)*

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

```

1868 \cs_new_protected:Npn \draw_set_linewidth:n #1
1869 {
1870   \dim_gset:Nn \g__draw_linewidth_dim { \fp_to_dim:n {#1} }
1871   \__draw_backend_linewidth:n \g__draw_linewidth_dim
1872 }

```

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

```

\draw_set_dash_pattern:nn Evaluated all of the list and pass it to the driver layer.
  \l__draw_tmp_seq 1873 \cs_new_protected:Npn \draw_set_dash_pattern:nn #1#2
                    1874 {
                    1875   \group_begin:
                    1876     \seq_set_from_clist:Nn \l__draw_tmp_seq {#1}
                    1877     \seq_set_map:Nn \l__draw_tmp_seq \l__draw_tmp_seq
                    1878       { \fp_to_dim:n {##1} }
                    1879     \use:e
                    1880     {
                    1881       \__draw_backend_dash_pattern:nn
                    1882       { \seq_use:Nn \l__draw_tmp_seq { , } }
                    1883       { \fp_to_dim:n {#2} }
                    1884     }
                    1885   \group_end:
                    1886 }
                    1887 \seq_new:N \l__draw_tmp_seq

```

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

```

\draw_set_miterlimit:n Pass through to the driver layer.
                    1888 \cs_new_protected:Npn \draw_set_miterlimit:n #1
                    1889   { \exp_args:Ne \__draw_backend_miterlimit:n { \fp_eval:n {#1} } }

```

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

```

\draw_set_cap_but: All straight wrappers.
\draw_set_cap_rectangle: 1890 \cs_new_protected:Npn \draw_set_cap_but: { \__draw_backend_cap_but: }
\draw_set_cap_round: 1891 \cs_new_protected:Npn \draw_set_cap_rectangle: { \__draw_backend_cap_rectangle: }
\draw_set_evenodd_rule: 1892 \cs_new_protected:Npn \draw_set_cap_round: { \__draw_backend_cap_round: }
\draw_set_nonzero_rule: 1893 \cs_new_protected:Npn \draw_set_evenodd_rule: { \__draw_backend_evenodd_rule: }
\draw_set_join_bevel: 1894 \cs_new_protected:Npn \draw_set_nonzero_rule: { \__draw_backend_nonzero_rule: }
\draw_set_join_miter: 1895 \cs_new_protected:Npn \draw_set_join_bevel: { \__draw_backend_join_bevel: }
\draw_set_join_round: 1896 \cs_new_protected:Npn \draw_set_join_miter: { \__draw_backend_join_miter: }
                    1897 \cs_new_protected:Npn \draw_set_join_round: { \__draw_backend_join_round: }

```

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

```
1898 \</package>
```

## 9 l3draw-transforms implementation

```
1899 \*package>
```

```
1900 \@@=draw>
```

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

- `\pgfgettransform`, `\pgfgettransformentries`: Awaiting use cases.
- `\pgftransformlineattime`, `\pgftransformarcaxesattime`, `\pgftransformcurveattime`: Need to look at the use cases for these to fully understand them.
- `\pgftransformarrow`: Likely to be done when other arrow functions are added.

- `\pgftransformationadjustments`: Used mainly by CircuiTikZ although also for shapes, likely needs more use cases before addressing.
- `\pgfflowlevelsynccm`, `\pgfflowlevel`: Likely to be added when use cases are encountered in other parts of the code.
- `\pgfviewboxscope`: Seems very specialized, need to understand the requirements here.

<code>\l__draw_matrix_active_bool</code>	An internal flag to avoid redundant calculations. <pre> 1901 \bool_new:N \l__draw_matrix_active_bool </pre> <i>(End of definition for \l__draw_matrix_active_bool.)</i>
<code>\l__draw_matrix_a_fp</code> <code>\l__draw_matrix_b_fp</code> <code>\l__draw_matrix_c_fp</code> <code>\l__draw_xshift_dim</code> <code>\l__draw_yshift_dim</code>	The active matrix and shifts. <pre> 1902 \fp_new:N \l__draw_matrix_a_fp 1903 \fp_new:N \l__draw_matrix_b_fp 1904 \fp_new:N \l__draw_matrix_c_fp 1905 \fp_new:N \l__draw_matrix_d_fp 1906 \dim_new:N \l__draw_xshift_dim 1907 \dim_new:N \l__draw_yshift_dim </pre> <i>(End of definition for \l__draw_matrix_a_fp and others.)</i>
<code>\draw_transform_matrix_reset:</code> <code>\draw_transform_shift_reset:</code>	Fast resetting. <pre> 1908 \cs_new_protected:Npn \draw_transform_matrix_reset: 1909 { 1910   \fp_set:Nn \l__draw_matrix_a_fp { 1 } 1911   \fp_zero:N \l__draw_matrix_b_fp 1912   \fp_zero:N \l__draw_matrix_c_fp 1913   \fp_set:Nn \l__draw_matrix_d_fp { 1 } 1914   \bool_set_false:N \l__draw_matrix_active_bool 1915 } 1916 \cs_new_protected:Npn \draw_transform_shift_reset: 1917 { 1918   \dim_zero:N \l__draw_xshift_dim 1919   \dim_zero:N \l__draw_yshift_dim 1920 } 1921 \draw_transform_matrix_reset: 1922 \draw_transform_shift_reset: </pre> <i>(End of definition for \draw_transform_matrix_reset: and \draw_transform_shift_reset:. These functions are documented on page ??.)</i>
<code>\draw_transform_matrix_absolute:nmmn</code> <code>\draw_transform_shift_absolute:n</code> <code>\__draw_transform_shift_absolute:nn</code>	Setting the transform matrix is straight-forward, with just a bit of expansion to sort out. With the mechanism active, the identity matrix is set. <pre> 1923 \cs_new_protected:Npn \draw_transform_matrix_absolute:nmmn #1#2#3#4 1924 { 1925   \fp_set:Nn \l__draw_matrix_a_fp {#1} 1926   \fp_set:Nn \l__draw_matrix_b_fp {#2} 1927   \fp_set:Nn \l__draw_matrix_c_fp {#3} 1928   \fp_set:Nn \l__draw_matrix_d_fp {#4} 1929   \bool_lazy_all:nTF 1930   { 1931     { \fp_compare_p:nNn \l__draw_matrix_a_fp = \c_one_fp } </pre>

```

1932     { \fp_compare_p:nNn \l__draw_matrix_b_fp = \c_zero_fp }
1933     { \fp_compare_p:nNn \l__draw_matrix_c_fp = \c_zero_fp }
1934     { \fp_compare_p:nNn \l__draw_matrix_d_fp = \c_one_fp }
1935   }
1936   { \bool_set_false:N \l__draw_matrix_active_bool }
1937   { \bool_set_true:N \l__draw_matrix_active_bool }
1938 }
1939 \cs_new_protected:Npn \draw_transform_shift_absolute:n #1
1940 {
1941   \__draw_point_process:nn
1942   { \__draw_transform_shift_absolute:nn } { \draw_point:n {#1} }
1943 }
1944 \cs_new_protected:Npn \__draw_transform_shift_absolute:nn #1#2
1945 {
1946   \dim_set:Nn \l__draw_xshift_dim {#1}
1947   \dim_set:Nn \l__draw_yshift_dim {#2}
1948 }

```

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

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

```

1949 \cs_new_protected:Npn \draw_transform_matrix:nnnn #1#2#3#4
1950 {
1951   \use:e
1952   {
1953     \__draw_transform:nnnn
1954     { \fp_eval:n {#1} }
1955     { \fp_eval:n {#2} }
1956     { \fp_eval:n {#3} }
1957     { \fp_eval:n {#4} }
1958   }
1959 }
1960 \cs_new_protected:Npn \__draw_transform:nnnn #1#2#3#4
1961 {
1962   \use:e
1963   {
1964     \draw_transform_matrix_absolute:nnnn
1965     { #1 * \l__draw_matrix_a_fp + #2 * \l__draw_matrix_c_fp }
1966     { #1 * \l__draw_matrix_b_fp + #2 * \l__draw_matrix_d_fp }
1967     { #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_c_fp }
1968     { #3 * \l__draw_matrix_b_fp + #4 * \l__draw_matrix_d_fp }
1969   }
1970 }
1971 \cs_new_protected:Npn \draw_transform_shift:n #1
1972 {
1973   \__draw_point_process:nn
1974   { \__draw_transform_shift:nn } { \draw_point:n {#1} }
1975 }
1976 \cs_new_protected:Npn \__draw_transform_shift:nn #1#2
1977 {
1978   \__draw_transform_shift:nnnn
1979   \l__draw_xshift_dim

```

```

1980     \l__draw_yshift_dim
1981     {#1} {#2}
1982 }

```

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

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

```

1983 \cs_new_protected:Npn \__draw_transform_shift:nnnn #1#2#3#4
1984 {
1985     \dim_set:Nn \l__draw_xshift_dim
1986     {
1987         \fp_to_dim:n
1988         {
1989             #1 +
1990             ( #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_c_fp )
1991         }
1992     }
1993     \dim_set:Nn \l__draw_yshift_dim
1994     {
1995         \fp_to_dim:n
1996         {
1997             #2 +
1998             ( #3 * \l__draw_matrix_b_fp + #4 * \l__draw_matrix_d_fp )
1999         }
2000     }
2001 }

```

(End of definition for `\__draw_transform_shift:nnnn`.)

`\draw_transform_matrix_invert:` Standard mathematics: calculate the inverse affine transformation. For the matrix, the inverse is calculated and applied. For the shift, the inverse vector is evaluated as  $-M^{-1} \cdot S$ .  
`\__draw_transform_invert:n` We use e-type expansion to snapshot the current floating-point values of the matrix and  
`\__draw_transform_invert:e` shift prior to assignment.

```

\draw_transform_shift_invert:
\__draw_transform_shift_invert:n
\__draw_transform_shift_invert:e
2002 \cs_new_protected:Npn \draw_transform_matrix_invert:
2003 {
2004     \bool_if:NT \l__draw_matrix_active_bool
2005     {
2006         \__draw_transform_invert:e
2007         {
2008             \fp_eval:n
2009             {
2010                 1 /
2011                 (
2012                     \l__draw_matrix_a_fp * \l__draw_matrix_d_fp
2013                     - \l__draw_matrix_b_fp * \l__draw_matrix_c_fp
2014                 )
2015             }
2016         }
2017     }
2018 }
2019 \cs_new_protected:Npn \__draw_transform_invert:n #1
2020 {

```

```

2021     \use:e
2022     {
2023         \draw_transform_matrix_absolute:nnnn
2024         { \l__draw_matrix_d_fp * #1 }
2025         { -\l__draw_matrix_b_fp * #1 }
2026         { -\l__draw_matrix_c_fp * #1 }
2027         { \l__draw_matrix_a_fp * #1 }
2028     }
2029 }
2030 \cs_generate_variant:Nn \__draw_transform_invert:n { e }
2031 \cs_new_protected:Npn \draw_transform_shift_invert:
2032 {
2033     \bool_if:NTF \l__draw_matrix_active_bool
2034     {
2035         \__draw_transform_shift_invert:e
2036         {
2037             \fp_eval:n
2038             {
2039                 1 /
2040                 (
2041                     \l__draw_matrix_a_fp * \l__draw_matrix_d_fp
2042                     - \l__draw_matrix_b_fp * \l__draw_matrix_c_fp
2043                 )
2044             }
2045         }
2046     }
2047     {
2048         \dim_set:Nn \l__draw_xshift_dim { -\l__draw_xshift_dim }
2049         \dim_set:Nn \l__draw_yshift_dim { -\l__draw_yshift_dim }
2050     }
2051 }
2052 \cs_new_protected:Npn \__draw_transform_shift_invert:n #1
2053 {
2054     \use:e
2055     {
2056         \__draw_transform_shift:nn
2057         {
2058             \fp_to_dim:n
2059             {
2060                 (
2061                     \l__draw_matrix_c_fp * \l__draw_yshift_dim
2062                     - \l__draw_matrix_d_fp * \l__draw_xshift_dim
2063                 ) * #1
2064             }
2065         }
2066         {
2067             \fp_to_dim:n
2068             {
2069                 (
2070                     \l__draw_matrix_b_fp * \l__draw_xshift_dim
2071                     - \l__draw_matrix_a_fp * \l__draw_yshift_dim
2072                 ) * #1
2073             }
2074         }

```



```

2075     }
2076   }
2077   \cs_generate_variant:Nn \__draw_transform_shift_invert:n { e }

```

(End of definition for `\draw_transform_matrix_invert:` and others. 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.

```

2078   \cs_new_protected:Npn \draw_transform_triangle:nnn #1#2#3
2079   {
2080     \__draw_point_process:nnnn
2081     { \__draw_transform_triangle:nnnnnn }
2082     { \draw_point:n {#1} }
2083     { \draw_point:n {#2} }
2084     { \draw_point:n {#3} }
2085   }
2086   \cs_new_protected:Npn \__draw_transform_triangle:nnnnnn #1#2#3#4#5#6
2087   {
2088     \use:e
2089     {
2090       \draw_transform_matrix:nnnn
2091       { #3 - #1 }
2092       { #4 - #2 }
2093       { #5 - #1 }
2094       { #6 - #2 }
2095       \draw_transform_shift:n { #1 , #2 }
2096     }
2097   }

```

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

`\draw_transform_scale:n` Lots of shortcuts.

```

\draw_transform_xscale:n 2098 \cs_new_protected:Npn \draw_transform_scale:n #1
\draw_transform_yscale:n 2099 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { #1 } }
\draw_transform_xshift:n 2100 \cs_new_protected:Npn \draw_transform_xscale:n #1
\draw_transform_yshift:n 2101 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { 1 } }
\draw_transform_xslant:n 2102 \cs_new_protected:Npn \draw_transform_yscale:n #1
\draw_transform_yslant:n 2103 { \draw_transform_matrix:nnnn { 1 } { 0 } { 0 } { #1 } }
2104 \cs_new_protected:Npn \draw_transform_xshift:n #1
2105 { \draw_transform_shift:n { #1 , 0pt } }
2106 \cs_new_protected:Npn \draw_transform_yshift:n #1
2107 { \draw_transform_shift:n { 0pt , #1 } }
2108 \cs_new_protected:Npn \draw_transform_xslant:n #1
2109 { \draw_transform_matrix:nnnn { 1 } { 0 } { #1 } { 1 } }
2110 \cs_new_protected:Npn \draw_transform_yslant:n #1
2111 { \draw_transform_matrix:nnnn { 1 } { #1 } { 0 } { 1 } }

```

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

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

```

\__draw_transform_rotate:n 2112 \cs_new_protected:Npn \draw_transform_rotate:n #1
\__draw_transform_rotate:e 2113 { \__draw_transform_rotate:e { \fp_eval:n {#1} } }
\__draw_transform_rotate:nn 2114 \cs_new_protected:Npn \__draw_transform_rotate:n #1
\__draw_transform_rotate:ee 2115 {

```

```

2116     \_draw_transform_rotate:ee
2117     { \fp_eval:n { cosd(#1) } }
2118     { \fp_eval:n { sind(#1) } }
2119 }
2120 \cs_generate_variant:Nn \_draw_transform_rotate:n { e }
2121 \cs_new_protected:Npn \_draw_transform_rotate:nn #1#2
2122 { \draw_transform_matrix:nnnn {#1} {#2} { -#2 } { #1 } }
2123 \cs_generate_variant:Nn \_draw_transform_rotate:nn { ee }

```

*(End of definition for \draw\_transform\_rotate:n, \\_draw\_transform\_rotate:n, and \\_draw\_transform\_rotate:nn. This function is documented on page ??.)*

```

2124 \</package>

```

# Index

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

## B

`\begin` . . . 200, 837, 1157, 1161, 1185, 1188

bool commands:

`\bool_gset_eq:NN` . . . . . 1552

`\bool_gset_false:N` . . . . . 1535, 1658

`\bool_gset_true:N` . . . . . 1559, 1605

`\bool_if:NTF` . . . . . 33, 145,

223, 262, 738, 754, 755, 759, 1289,

1322, 1456, 1587, 1642, 2004, 2033

`\bool_lazy_all:nTF` . . . . . 1929

`\bool_lazy_and:nnTF` . . . . .

. . . . . 254, 733, 1688, 1722

`\bool_lazy_any:nTF` . . . . . 119, 1733

`\bool_lazy_or:nnTF` . . . . .

. . . . . 107, 591, 692, 742, 747

`\bool_new:N` . 99, 249, 681, 682, 683,

684, 804, 1346, 1447, 1521, 1558, 1901

`\bool_set_eq:NN` . . . . . 1543

`\bool_set_false:N` . . . . . 132,

257, 720, 721, 722, 741, 1914, 1936

`\bool_set_true:N` . . . . . 134, 258,

726, 767, 771, 772, 1365, 1451, 1937

box commands:

`\box_dp:N` . . . . . 17, 22, 89

`\box_gclear:N` . . . . . 177

`\box_gset_eq:NN` . . . . . 187

`\box_gset_wd:Nn` . . . . . 136, 163

`\box_ht:N` . . . . . 17, 22, 91

`\box_if_exist_p:N` . . . . . 109, 123

`\box_move_down:nn` . . . . . 1421, 1434

`\box_move_up:nn` . . . . . 63

`\box_new:N` . . . 13, 111, 112, 1347, 1348

`\box_set_dp:Nn` . . . . . 67, 1425, 1438

`\box_set_eq:NN` . . . . . 175

`\box_set_ht:Nn` . . . . . 66, 1426, 1444

`\box_set_wd:Nn` . . . . . 68, 158, 1395

`\box_use_drop:N` . . . . . 64,

69, 138, 159, 164, 1398, 1423, 1436

`\box_wd:N` . . . . . 17, 22, 88, 90

## C

clist commands:

`\clist_if_in:NnTF` . . . . . 105

`\clist_map_inline:Nn` . . 154, 171, 183

`\clist_map_inline:nn` . . . . . 723

`\clist_new:N` . . . . . 100, 102

`\clist_set:Nn` . . . . . 101, 1383

coffin commands:

`\coffin_typeset:Nnnnn` . . . . . 86

`\coffin_wd:N` . . . . . 88

color commands:

`\color_ensure_current:` . . . . . 1370

cs commands:

`\cs_generate_variant:Nn` . . . . 453,

647, 680, 845, 852, 859, 870, 878,

886, 904, 932, 952, 959, 967, 974,

983, 989, 1011, 1019, 1026, 1032,

1045, 1048, 1066, 1086, 1094, 1100,

1123, 1132, 1156, 1170, 1184, 1205,

1238, 1257, 1269, 1281, 1415, 1524,

1607, 1794, 2030, 2077, 2120, 2123

`\cs_if_exist:NTF` . . . . . 725, 1406

`\cs_if_exist_use:NTF` . . 436, 445, 728

`\cs_new:Npn` . . . . .

. . . . 318, 319, 546, 556, 566, 576,

841, 843, 846, 848, 850, 853, 855,

857, 860, 863, 865, 871, 874, 876,

879, 880, 882, 884, 887, 889, 895,

905, 914, 924, 933, 940, 945, 953,

960, 968, 975, 984, 990, 998, 1003,

1012, 1020, 1027, 1033, 1040, 1046,

1049, 1055, 1064, 1067, 1075, 1080,

1087, 1095, 1101, 1109, 1115, 1124,

1133, 1142, 1148, 1162, 1171, 1177,

1189, 1191, 1200, 1228, 1230, 1239,

1244, 1258, 1260, 1262, 1270, 1282,

1287, 1314, 1320, 1789, 1795, 1852

`\cs_new_protected:Npe` . . . . . 1401

`\cs_new_protected:Npn` . . . . .

. . . 14, 19, 24, 31, 72, 77, 82, 103,

128, 143, 152, 169, 181, 213, 235,

242, 250, 260, 269, 275, 281, 287,

294, 305, 313, 320, 322, 324, 335,

342, 378, 380, 391, 397, 427, 454,

483, 489, 495, 500, 508, 517, 524,

532, 587, 589, 605, 612, 621, 628,

630, 641, 648, 654, 661, 685, 690,

701, 710, 718, 765, 769, 774, 781,

805, 819, 1212, 1214, 1216, 1218,

1220, 1350, 1357, 1379, 1416, 1429,

1449, 1454, 1467, 1474, 1485, 1494,

1502, 1510, 1522, 1525, 1532, 1537,

1548, 1560, 1569, 1578, 1583, 1593,

1601, 1608, 1610, 1612, 1614, 1616,

1618, 1620, 1622, 1624, 1626, 1628,

1640, 1660, 1684, 1694, 1711, 1720,  
 1731, 1752, 1761, 1820, 1853, 1868,  
 1873, 1888, 1890, 1891, 1892, 1893,  
 1894, 1895, 1896, 1897, 1908, 1916,  
 1923, 1939, 1944, 1949, 1960, 1971,  
 1976, 1983, 2002, 2019, 2031, 2052,  
 2078, 2086, 2098, 2100, 2102, 2104,  
 2106, 2108, 2110, 2112, 2114, 2121

## D

dim commands:

\dim\_abs:n ..... 635, 636  
 \dim\_compare:nNnTF 643, 650, 783, 1387  
 \dim\_compare\_p:nNn 255, 256, 1689, 1690  
 \dim\_gset:Nn ..... 215, 217, 219,  
 221, 225, 227, 229, 231, 237, 238,  
 239, 240, 244, 245, 712, 785, 1352,  
 1353, 1354, 1355, 1589, 1590, 1870  
 \dim\_gset\_eq:NN .....  
 .... 822, 823, 824, 825, 826, 827,  
 828, 829, 1477, 1496, 1497, 1498, 1499  
 \dim\_gzero:N .. 1389, 1390, 1391, 1392  
 \dim\_horizontal:N ..... 62  
 \dim\_max:nn .. 216, 220, 226, 230, 1440  
 \dim\_min:nn ..... 218, 222, 228, 232  
 \dim\_new:N ..... 207, 208,  
 209, 210, 211, 212, 247, 248, 796,  
 797, 798, 799, 800, 801, 802, 803,  
 1206, 1207, 1208, 1209, 1210, 1211,  
 1342, 1343, 1344, 1345, 1448, 1466,  
 1481, 1482, 1483, 1484, 1556, 1557,  
 1635, 1636, 1865, 1866, 1906, 1907  
 \dim\_set:Nn ..... 252, 253,  
 1222, 1223, 1452, 1686, 1687, 1867,  
 1946, 1947, 1985, 1993, 2048, 2049  
 \dim\_set\_eq:NN .....  
 .... 808, 809, 810, 811, 812, 813,  
 814, 815, 1471, 1488, 1489, 1490, 1491  
 \dim\_step\_inline:nnnn ..... 663, 671  
 \dim\_use:N ..... 318, 319,  
 714, 783, 788, 790, 1461, 1565, 1566  
 \dim\_zero:N ..... 1918, 1919  
 \c\_max\_dim ..... 237, 238, 239,  
 240, 783, 1352, 1353, 1354, 1355, 1387

draw commands:

\l\_draw\_bb\_update\_bool .....  
 .... 33, 223, 734, 741, 1346, 1365  
 \g\_draw\_bb\_xmax\_dim ..... 225,  
 226, 1342, 1352, 1389, 1396, 1488, 1496  
 \g\_draw\_bb\_xmin\_dim .....  
 .... 227, 228, 1342, 1353, 1387,  
 1390, 1396, 1420, 1433, 1489, 1497  
 \g\_draw\_bb\_ymax\_dim 229, 230, 1342,  
 1354, 1391, 1427, 1445, 1490, 1498

\g\_draw\_bb\_ymin\_dim .....  
 .... 231, 232, 1342, 1355,  
 1392, 1422, 1427, 1441, 1491, 1499  
 \draw\_begin: ..... 1357, 1357  
 \draw\_box\_use:N ..... 14, 14  
 \draw\_box\_use:Nn ..... 14, 19  
 \draw\_coffin\_use:Nnn ..... 37, 72, 72  
 \draw\_coffin\_use:Nnnn ..... 72, 77  
 \l\_draw\_default\_linewidth\_dim ...  
 .... 141, 1369, 1866  
 \draw\_end: ..... 1357, 1379  
 \g\_draw\_id\_int ..... 1349, 1360, 1409  
 \draw\_layer\_begin:n ..... 103, 103  
 \draw\_layer\_end: ..... 103, 143  
 \l\_draw\_layers\_clist .....  
 .... 100, 105, 154, 171, 183, 1383  
 \draw\_path\_arc:nnn .... 378, 378, 521  
 \draw\_path\_arc:nnnn ... 378, 379, 380  
 \draw\_path\_arc\_axes:nnnn ... 517, 517  
 \draw\_path\_canvas\_curveto:nnn ...  
 .... 320, 324  
 \draw\_path\_canvas\_lineto:n . 320, 322  
 \draw\_path\_canvas\_moveto:n . 320, 320  
 \draw\_path\_circle:nn ..... 587, 587  
 \draw\_path\_close: ..... 313, 313, 618  
 \draw\_path\_corner\_arc:nn ... 250, 250  
 \draw\_path\_curveto:nn ..... 335, 335  
 \draw\_path\_curveto:nnn ..... 269, 294  
 \draw\_path\_ellipse:nnn . 524, 524, 588  
 \draw\_path\_grid:nnnn ..... 630, 630  
 \draw\_path\_lastx: ..... 318, 318  
 \draw\_path\_lasty: ..... 318, 319  
 \draw\_path\_lineto:n .....  
 .... 269, 281, 615, 616, 617, 669, 677  
 \draw\_path\_moveto:n .....  
 .... 269, 269, 614, 619, 668, 676  
 \draw\_path\_rectangle:nn 589, 589, 629  
 \draw\_path\_rectangle\_corners:nn .  
 .... 621, 621  
 \draw\_path\_replace\_bb: ..... 685, 701  
 \draw\_path\_scope\_begin: .....  
 .... 805, 805, 1472, 1505  
 \draw\_path\_scope\_end: .....  
 .... 805, 819, 1476, 1513  
 \draw\_path\_use:n ..... 685, 685  
 \draw\_path\_use\_clear:n ..... 685, 690  
 \draw\_point:n 321, 323, 331, 332, 333,  
 596, 597, 601, 602, 625, 626, 638,  
 639, 874, 874, 885, 888, 900, 909,  
 910, 911, 912, 926, 937, 938, 986,  
 994, 995, 996, 1029, 1037, 1038,  
 1047, 1065, 1071, 1072, 1073, 1097,  
 1105, 1106, 1107, 1126, 1137, 1138,  
 1139, 1140, 1202, 1219, 1232, 1246,

1272, 1285, 1291, 1307, 1318, 1324,  
 1337, 1942, 1974, 2082, 2083, 2084  
 \draw\_point\_interpolate\_arc\_-  
     axes:nnnnnn ..... 1067, 1067  
 \draw\_point\_interpolate\_curve:nnnn  
     ..... 1101, 1101  
 \draw\_point\_interpolate\_curve:nnnnn  
     ..... 1133, 1133  
 \draw\_point\_interpolate\_distance:nnn  
     ..... 1049, 1049, 1765, 1774  
 \draw\_point\_interpolate\_line:nnn  
     ..... 1033, 1033  
 \draw\_point\_intersect\_circles:nnnnn  
     ..... 933, 933  
 \draw\_point\_intersect\_line\_-  
     circle:nnnnn ..... 990, 990  
 \draw\_point\_intersect\_lines:nnnn  
     ..... 905, 905  
 \draw\_point\_polar:nn ..... 880, 880  
 \draw\_point\_polar:nnn .....  
     .... 461, 467, 471, 477, 880, 881, 882  
 \draw\_point\_transform:n ..... 37,  
     40, 43, 46, 273, 285, 301, 302, 303,  
     339, 340, 466, 470, 528, 1282, 1282  
 \draw\_point\_unit\_vector:n .....  
     ..... 887, 887, 1062  
 \draw\_point\_vec:nn ..... 1228, 1228  
 \draw\_point\_vec:nnn ..... 1228, 1239  
 \draw\_point\_vec\_polar:nn . 1258, 1258  
 \draw\_point\_vec\_polar:nnn .....  
     ..... 1258, 1259, 1260  
 \draw\_scope\_begin: ... 26, 1467, 1467  
 \draw\_scope\_end: ..... 29, 1474  
 \draw\_set\_baseline:n ..... 1449, 1449  
 \draw\_set\_cap\_but: . 1372, 1890, 1890  
 \draw\_set\_cap\_rectangle: . 1890, 1891  
 \draw\_set\_cap\_round: .... 1890, 1892  
 \draw\_set\_dash\_pattern:nn .....  
     ..... 1375, 1873, 1873  
 \draw\_set\_evenodd\_rule: .. 1890, 1893  
 \draw\_set\_join\_bevel: ... 1890, 1895  
 \draw\_set\_join\_miter: 1373, 1890, 1896  
 \draw\_set\_join\_round: ... 1890, 1897  
 \draw\_set\_linewidth:n .....  
     ..... 141, 1369, 1868, 1868  
 \draw\_set\_miterlimit:n .....  
     ..... 1374, 1888, 1888  
 \draw\_set\_nonzero\_rule: .....  
     ..... 1371, 1890, 1894  
 \draw\_suspend\_begin: .... 1502, 1502  
 \draw\_suspend\_end: ..... 1502, 1510  
 \draw\_transform\_matrix:nnnn .....  
     ..... 1949, 1949, 2090,  
     2099, 2101, 2103, 2109, 2111, 2122  
 \draw\_transform\_matrix\_absolute:nnnn  
     ..... 1923, 1923, 1964, 2023  
 \draw\_transform\_matrix\_invert: ..  
     ..... 2002, 2002  
 \draw\_transform\_matrix\_reset: ...  
     ..... 1366, 1506, 1908, 1908, 1921  
 \draw\_transform\_rotate:n . 2112, 2112  
 \draw\_transform\_scale:n . 2098, 2098  
 \draw\_transform\_shift:n .....  
     .... 27, 1949, 1971, 2095, 2105, 2107  
 \draw\_transform\_shift\_absolute:n  
     ..... 1923, 1939  
 \draw\_transform\_shift\_invert: ...  
     ..... 2002, 2031  
 \draw\_transform\_shift\_reset: ...  
     ..... 1367, 1507, 1908, 1916, 1922  
 \draw\_transform\_triangle:nnn ...  
     ..... 520, 2078, 2078  
 \draw\_transform\_xscale:n . 2098, 2100  
 \draw\_transform\_xshift:n . 2098, 2104  
 \draw\_transform\_xslant:n . 2098, 2108  
 \draw\_transform\_yscale:n . 2098, 2102  
 \draw\_transform\_yshift:n . 2098, 2106  
 \draw\_transform\_yslant:n . 2098, 2110  
 \draw\_xvec:n ..... 1212, 1212, 1225  
 \draw\_yvec:n ..... 1212, 1214, 1226  
 \draw\_zvec:n ..... 1212, 1216, 1227  
 draw internal commands:  
     \\_draw\_backend\_begin: ..... 1362  
     \\_draw\_backend\_box\_use:Nnnnn ... 53  
     \\_draw\_backend\_cap\_but: .... 1890  
     \\_draw\_backend\_cap\_rectangle: 1891  
     \\_draw\_backend\_cap\_round: ... 1892  
     \\_draw\_backend\_clip: ..... 740  
     \\_draw\_backend\_closepath: ... 1609  
     \\_draw\_backend\_curveto:nnnnnn 1613  
     \\_draw\_backend\_dash\_pattern:nn 1881  
     \\_draw\_backend\_discardpath: .. 745  
     \\_draw\_backend\_end: ..... 1385  
     \\_draw\_backend\_evenodd\_rule: . 1893  
     \\_draw\_backend\_join\_bevel: .. 1895  
     \\_draw\_backend\_join\_miter: .. 1896  
     \\_draw\_backend\_join\_round: .. 1897  
     \\_draw\_backend\_lineto:nn .... 1619  
     \\_draw\_backend\_linewidth:n .. 1871  
     \\_draw\_backend\_miterlimit:n . 1889  
     \\_draw\_backend\_moveto:nn .... 1621  
     \\_draw\_backend\_nonzero\_rule: . 1894  
     \\_draw\_backend\_rectangle:nnnn 1627  
     \\_draw\_backend\_scope\_begin: ...  
         ..... 162, 1469  
     \\_draw\_backend\_scope\_end: 165, 1479  
     \\_l\\_draw\_baseline\_bool .....  
         ..... 1447, 1451, 1456

\l__draw_baseline_dim	<a href="#">1447</a> , <a href="#">1452</a> , <a href="#">1461</a>	\__draw_path_arc:nnnn	<a href="#">378</a> , <a href="#">384</a> , <a href="#">391</a>
\__draw_baseline_finalize:w	....	\__draw_path_arc:nnNnn	.....
.....	<a href="#">1381</a> , <a href="#">1454</a> , <a href="#">1454</a>	.....	<a href="#">378</a> , <a href="#">394</a> , <a href="#">395</a> , <a href="#">397</a>
\__draw_box_use:Nnnnn	..... <a href="#">14</a>	\c__draw_path_arc_60_fp	..... <a href="#">378</a>
\__draw_box_use:nNnnnn	.... <a href="#">21</a> , <a href="#">24</a> , <a href="#">79</a>	\c__draw_path_arc_90_fp	..... <a href="#">378</a>
\__draw_box_use:Nnnnnnn	<a href="#">16</a> , <a href="#">28</a> , <a href="#">31</a> , <a href="#">74</a>	\__draw_path_arc_add:nnnn	..... <a href="#">378</a>
\__draw_box_use:nNnnnnnn	..... <a href="#">14</a>	\__draw_path_arc_aux_add:nn	....
\__draw_coffin_use:nNnn	<a href="#">72</a> , <a href="#">74</a> , <a href="#">79</a> , <a href="#">82</a>	.....	<a href="#">485</a> , <a href="#">491</a> , <a href="#">503</a> , <a href="#">508</a>
\l__draw_corner_arc_bool	.....	\__draw_path_arc_auxi:nnnnNnn	...
.....	<a href="#">249</a> , <a href="#">257</a> , <a href="#">258</a> , <a href="#">262</a> , <a href="#">592</a>	.....	<a href="#">378</a> , <a href="#">405</a> , <a href="#">412</a> , <a href="#">420</a> , <a href="#">427</a> , <a href="#">453</a>
\l__draw_corner_xarc_dim	.....	\__draw_path_arc_auxii:nnnNnnnn	..
.....	<a href="#">247</a> , <a href="#">252</a> , <a href="#">255</a> , <a href="#">265</a>	.....	<a href="#">378</a> , <a href="#">431</a> , <a href="#">454</a>
\l__draw_corner_yarc_dim	.....	\__draw_path_arc_auxiii:nn	.....
.....	<a href="#">247</a> , <a href="#">253</a> , <a href="#">256</a> , <a href="#">266</a>	.....	<a href="#">378</a> , <a href="#">458</a> , <a href="#">483</a>
\__draw_draw_polar:nnn	.....	\__draw_path_arc_auxiv:nnnn	.....
.....	<a href="#">880</a> , <a href="#">883</a> , <a href="#">884</a> , <a href="#">886</a>	.....	<a href="#">378</a> , <a href="#">464</a> , <a href="#">489</a>
\__draw_finalize:	.....	\__draw_path_arc_auxv:nn	<a href="#">378</a> , <a href="#">474</a> , <a href="#">495</a>
.....	<a href="#">1394</a> , <a href="#">1416</a> , <a href="#">1416</a> , <a href="#">1454</a> , <a href="#">1464</a>	\__draw_path_arc_auxvi:nn	.....
\__draw_finalize_baseline:n	....	.....	<a href="#">378</a> , <a href="#">497</a> , <a href="#">500</a>
.....	<a href="#">1416</a> , <a href="#">1429</a> , <a href="#">1461</a>	\l__draw_path_arc_delta_fp	.... <a href="#">378</a>
\__draw_if_recursion_tail_stop-		\l__draw_path_arc_start_fp	.... <a href="#">378</a>
do:Nn	..... <a href="#">9</a> , <a href="#">9</a> , <a href="#">1662</a>	\__draw_path_curveto:nnnn	.....
\__draw_layer_begin:n	.. <a href="#">103</a> , <a href="#">116</a> , <a href="#">128</a>	.....	<a href="#">335</a> , <a href="#">338</a> , <a href="#">342</a>
\l__draw_layer_close_bool	.....	\__draw_path_curveto:nnnnnn	....
.....	<a href="#">99</a> , <a href="#">132</a> , <a href="#">134</a> , <a href="#">145</a>	.....	<a href="#">269</a> , <a href="#">299</a> ,
\l__draw_layer_main_box	.....		<a href="#">305</a> , <a href="#">329</a> , <a href="#">349</a> , <a href="#">479</a> , <a href="#">548</a> , <a href="#">558</a> , <a href="#">568</a> , <a href="#">578</a>
.....	<a href="#">158</a> , <a href="#">159</a> , <a href="#">1347</a> , <a href="#">1376</a>	\c__draw_path_curveto_a_fp	.... <a href="#">335</a>
\l__draw_layer_tl	..... <a href="#">97</a> , <a href="#">131</a> , <a href="#">135</a>	\c__draw_path_curveto_b_fp	.... <a href="#">335</a>
\g__draw_layers_clist	..... <a href="#">100</a>	\__draw_path_ellipse:nnnnnn	....
\__draw_layers_insert:	<a href="#">152</a> , <a href="#">152</a> , <a href="#">1384</a>	.....	<a href="#">524</a> , <a href="#">527</a> , <a href="#">532</a>
\__draw_layers_restore:	<a href="#">169</a> , <a href="#">181</a> , <a href="#">1512</a>	\__draw_path_ellipse_arci:nnnnnn	.....
\__draw_layers_save:	.. <a href="#">169</a> , <a href="#">169</a> , <a href="#">1508</a>	.....	<a href="#">524</a> , <a href="#">538</a> , <a href="#">546</a>
\g__draw_linewidth_dim	.....	\__draw_path_ellipse_arcii:nnnnnn	.....
	<a href="#">715</a> , <a href="#">791</a> , <a href="#">1471</a> , <a href="#">1477</a> , <a href="#">1865</a> , <a href="#">1870</a> , <a href="#">1871</a>	.....	<a href="#">524</a> , <a href="#">539</a> , <a href="#">556</a>
\l__draw_linewidth_dim	.....	\__draw_path_ellipse_arciiii:nnnnnn	.....
.....	<a href="#">1466</a> , <a href="#">1471</a> , <a href="#">1477</a>	.....	<a href="#">524</a> , <a href="#">540</a> , <a href="#">566</a>
\l__draw_main_box	.....	\__draw_path_ellipse_arciv:nnnnnn	.....
	<a href="#">1347</a> , <a href="#">1361</a> , <a href="#">1395</a> , <a href="#">1398</a> , <a href="#">1418</a> , <a href="#">1423</a> ,	.....	<a href="#">524</a> , <a href="#">541</a> , <a href="#">576</a>
	<a href="#">1425</a> , <a href="#">1426</a> , <a href="#">1431</a> , <a href="#">1436</a> , <a href="#">1438</a> , <a href="#">1444</a>	\c__draw_path_ellipse_fp	..... <a href="#">524</a>
\l__draw_matrix_a_fp	... <a href="#">54</a> , <a href="#">1294</a> ,	\__draw_path_grid_auxi:nnnnnn	...
	<a href="#">1327</a> , <a href="#">1902</a> , <a href="#">1910</a> , <a href="#">1925</a> , <a href="#">1931</a> , <a href="#">1965</a> ,	.....	<a href="#">630</a> , <a href="#">634</a> , <a href="#">641</a> , <a href="#">647</a>
	<a href="#">1967</a> , <a href="#">1990</a> , <a href="#">2012</a> , <a href="#">2027</a> , <a href="#">2041</a> , <a href="#">2071</a>	\__draw_path_grid_auxii:nnnnnn	..
\l__draw_matrix_active_bool	....	.....	<a href="#">630</a> , <a href="#">644</a> , <a href="#">645</a> , <a href="#">648</a>
.....	<a href="#">593</a> , <a href="#">1289</a> , <a href="#">1322</a> ,	\__draw_path_grid_auxiii:nnnnnn	..
	<a href="#">1901</a> , <a href="#">1914</a> , <a href="#">1936</a> , <a href="#">1937</a> , <a href="#">2004</a> , <a href="#">2033</a>	.....	<a href="#">630</a> , <a href="#">651</a> , <a href="#">652</a> , <a href="#">654</a>
\l__draw_matrix_b_fp	... <a href="#">55</a> , <a href="#">1300</a> ,	\__draw_path_grid_auxiiii:nnnnnn	<a href="#">630</a>
	<a href="#">1332</a> , <a href="#">1902</a> , <a href="#">1911</a> , <a href="#">1926</a> , <a href="#">1932</a> , <a href="#">1966</a> ,	\__draw_path_grid_auxiv:nnnnnnnn	.....
	<a href="#">1968</a> , <a href="#">1998</a> , <a href="#">2013</a> , <a href="#">2025</a> , <a href="#">2042</a> , <a href="#">2070</a>	.....	<a href="#">630</a> , <a href="#">656</a> , <a href="#">661</a> , <a href="#">680</a>
\l__draw_matrix_c_fp	... <a href="#">56</a> , <a href="#">1295</a> ,	\g__draw_path_lastx_dim	.....
	<a href="#">1328</a> , <a href="#">1902</a> , <a href="#">1912</a> , <a href="#">1927</a> , <a href="#">1933</a> , <a href="#">1965</a> ,		<a href="#">207</a> , <a href="#">244</a> , <a href="#">318</a> , <a href="#">353</a> , <a href="#">486</a> , <a href="#">492</a> , <a href="#">808</a> , <a href="#">828</a>
	<a href="#">1967</a> , <a href="#">1990</a> , <a href="#">2013</a> , <a href="#">2026</a> , <a href="#">2042</a> , <a href="#">2061</a>	\l__draw_path_lastx_dim	<a href="#">796</a> , <a href="#">808</a> , <a href="#">828</a>
\l__draw_matrix_d_fp	... <a href="#">57</a> , <a href="#">1301</a> ,	\g__draw_path_lasty_dim	.....
	<a href="#">1333</a> , <a href="#">1905</a> , <a href="#">1913</a> , <a href="#">1928</a> , <a href="#">1934</a> , <a href="#">1966</a> ,		<a href="#">207</a> , <a href="#">245</a> , <a href="#">319</a> , <a href="#">360</a> , <a href="#">487</a> , <a href="#">493</a> , <a href="#">809</a> , <a href="#">829</a>
	<a href="#">1968</a> , <a href="#">1998</a> , <a href="#">2012</a> , <a href="#">2024</a> , <a href="#">2041</a> , <a href="#">2062</a>	\l__draw_path_lasty_dim	<a href="#">796</a> , <a href="#">809</a> , <a href="#">829</a>

\__draw_path_lineto:nn .....	\g__draw_path_ymin_dim .....
..... 269, 284, 287, 323	..... 209, 221, 222, 240, 813, 827
\__draw_path_mark_corner: .....	\l__draw_path_ymin_dim . 796, 813, 827
..... 260, 260, 289, 298, 315, 328, 348, 419	\__draw_point_interpolate_-
\__draw_path_moveto:nn .....	arcaxes_auxi:nnnnnnnnnn .....
..... 269, 272, 275, 321, 536, 544	..... 1067, 1070, 1075
\__draw_path_rectangle:nnnn .....	\__draw_point_interpolate_-
..... 589, 600, 605	arcaxes_auxii:nnnnnnnnnn .....
\__draw_path_rectangle_corners:nnnn	..... 1067, 1077, 1080, 1086
..... 621	\__draw_point_interpolate_-
\__draw_path_rectangle_corners:nnnnn	arcaxes_auxiii:nnnnnnnn .....
..... 624, 628	..... 1067, 1082, 1087, 1094
\__draw_path_rectangle_rounded:nnnn	\__draw_point_interpolate_-
..... 589, 595, 612	arcaxes_auxiv:nnnnnnnn .....
\__draw_path_replace_bb:NnN .....	..... 1067, 1089, 1095, 1100
..... 685, 703, 704, 705, 706, 710	\__draw_point_interpolate_curve_-
\__draw_path_reset_limits: .....	auxi:nnnnnnnnnn .. 1133, 1136, 1142
... 213, 235, 697, 708, 762, 816, 1364	\__draw_point_interpolate_curve_-
\l__draw_path_tmp_tl .....	auxii:nnnnnnnnnn .....
..... 204, 456, 479, 498, 502, 506, 510	..... 1133, 1144, 1148, 1156
\l__draw_path_tmpa_fp .....	\__draw_point_interpolate_curve_-
..... 204, 344, 354, 366	auxiii:nnnnnnnn 1133, 1151, 1162, 1170
\l__draw_path_tmpb_fp .....	\__draw_point_interpolate_curve_-
..... 204, 345, 361, 370	auxiv:nnnnnn .....
\__draw_path_update_last:nn .....	..... 1133, 1164, 1165, 1166, 1171
..... 242, 242, 279, 292, 311, 610	\__draw_point_interpolate_curve_-
\__draw_path_update_limits:nn .....	auxv:nnw ... 1133, 1173, 1177, 1184
..... 36, 39, 42, 45, 213,	\__draw_point_interpolate_curve_-
213, 277, 290, 307, 308, 309, 607, 608	auxvi:n .....
\__draw_path_use:n . 685, 688, 699, 718	..... 1133, 1168, 1189
\__draw_path_use_action_draw: .....	\__draw_point_interpolate_curve_-
..... 685, 765	auxvii:nnnnnnnnnn . 1133, 1190, 1191
\__draw_path_use_action_fillstroke:	\__draw_point_interpolate_curve_-
..... 685, 769	auxviii:nnnnnn .....
\__draw_path_use_bb:NnN .....	..... 1133, 1193, 1200, 1205
..... 685, 776, 777, 778, 779, 781	\__draw_point_interpolate_curve_-
\l__draw_path_use_clear_bool 684, 759	quad_auxi:nnnnnnnn 1101, 1104, 1109
\l__draw_path_use_clip_bool .....	\__draw_point_interpolate_curve_-
..... 681, 720, 738	quad_auxii:nnnnnnnn .....
\l__draw_path_use_fill_bool .....	..... 1101, 1111, 1115, 1123
..... 681, 721, 743, 748, 754, 771	\__draw_point_interpolate_curve_-
\__draw_path_use_stroke_bb: .....	quad_auxiii:nnnnnnnnnn .....
..... 685, 736, 774	..... 1101, 1117, 1124, 1132
\l__draw_path_use_stroke_bool ...	\__draw_point_interpolate_-
681, 722, 735, 744, 749, 755, 767, 772	distance:nnnn .....
\g__draw_path_xmax_dim .....	..... 1052, 1055
..... 209, 215, 216, 237, 810, 824	\__draw_point_interpolate_-
\l__draw_path_xmax_dim . 796, 810, 824	distance:nnnnnn .....
\g__draw_path_xmin_dim .....	..... 1049, 1059, 1064, 1066
..... 209, 217, 218, 238, 811, 825	\__draw_point_interpolate_-
\l__draw_path_xmin_dim . 796, 811, 825	distance:nnnnnn .....
\g__draw_path_ymax_dim .....	..... 1049
..... 209, 219, 220, 239, 812, 826	\__draw_point_interpolate_line_-
\l__draw_path_ymax_dim . 796, 812, 826	aux:nnnnnn .. 1033, 1036, 1040, 1045
	\__draw_point_interpolate_line_-
	aux:nnnnnnnn . 1033, 1042, 1046, 1048

\_draw_point_intersect_circles_- auxi:nnnnnnn . . . . .	933, 936, 940	\_draw_point_process_auxv:nnnn . . . . . .	841, 856, 857, 859
\_draw_point_intersect_circles_- auxii:nnnnnnn . . . . .	933, 942, 945, 952	\_draw_point_process_auxvi:nw . . . . . . .	841, 858, 860
\_draw_point_intersect_circles_- auxiii:nnnnnnn . . . . .	933, 947, 953, 959	\_draw_point_process_auxvii:nnnnn . . . . .	841, 864, 865, 870
\_draw_point_intersect_circles_- auxiv:nnnnnnnn . . . . .	933, 955, 960, 967	\_draw_point_process_auxviii:nw . . . . .	841, 867, 871
\_draw_point_intersect_circles_- auxv:nnnnnnnnn . . . . .	933, 962, 968, 974	\_draw_point_to_dim:n . . . . .	874, 875, 876, 878
\_draw_point_intersect_circles_- auxvi:nnnnnnnnn . . . . .	933, 970, 975, 983	\_draw_point_to_dim:w . . . . .	874, 877, 879
\_draw_point_intersect_circles_- auxvii:nnnnnnn . . . . .	933, 977, 984, 989	\_draw_point_transform:nn . . . . .	1282, 1285, 1287
\_draw_point_intersect_line_- circle_auxi:nnnnnnnn 990, 993, 998		\_draw_point_transform_noshift:n . . . . .	460, 476, 529, 530, 1314, 1314
\_draw_point_intersect_line_- circle_auxii:nnnnnnnn . . . . .	990, 1000, 1003, 1011	\_draw_point_transform_noshift:nn . . . . .	1314, 1317, 1320
\_draw_point_intersect_line_- circle_auxiii:nnnnnnnn . . . . .	990, 1005, 1012, 1019	\_draw_point_unit_vector:nn . . . . . . . .	887, 888, 889
\_draw_point_intersect_line_- circle_auxiv:nnnnnnnn . . . . .	990, 1014, 1020, 1026	\_draw_point_unit_vector:nnn . . . . . . . .	887, 891, 895, 904
\_draw_point_intersect_line_- circle_auxv:nnnnn . . . . .	990, 1022, 1027, 1032	\_draw_point_vec:nn . . . . .	1228, 1229, 1230, 1238
\_draw_point_intersect_lines:nnnnnn . . . . .	905	\_draw_point_vec:nnn . . . . .	1228, 1241, 1244, 1257
\_draw_point_intersect_lines:nnnnnnnn . . . . .	905, 908, 914	\_draw_point_vec_polar:nnn . . . . .	1258, 1261, 1262, 1269
\_draw_point_intersect_lines_- aux:nnnnnn . . . . .	905, 916, 924, 932	\_draw_point_vec_polar:nnnn . . . . . . . .	1258, 1264, 1270, 1281
\_draw_point_process:nn . . . . .	35, 38, 41, 44, 271, 283, 321, 323, 457, 473, 841, 841, 888, 1051, 1057, 1219, 1284, 1316, 1941, 1973	\_draw_record_origin: . . . . . . . . . .	1377, 1401, 1401
\_draw_point_process:nnn 337, 463, 595, 600, 623, 632, 841, 848, 935, 1035		\_draw_reset_bb: . . . . . . . . . .	1350, 1350, 1363, 1492
\_draw_point_process:nnnn . . . . .	31, 296, 326, 526, 841, 855, 992, 1069, 1103, 2080	\_draw_scope_bb_begin: . . . . . . . . . .	1485, 1485, 1504
\_draw_point_process:nnnnn . . . . .	841, 863, 907, 1135	\_draw_scope_bb_end: 1485, 1494, 1514	
\_draw_point_process_auxi:nn . . . . . . . .	841, 842, 843, 845	\_draw_softpath_add:n . . . . . . . . . .	1522, 1522, 1524, 1551, 1562, 1571, 1580, 1585, 1595, 1603, 1860
\_draw_point_process_auxii:nw . . . . . . .	841, 844, 846	\c__draw_softpath_arc_fp . . . . . . . . . .	1639, 1802, 1806, 1811, 1815
\_draw_point_process_auxiii:nnn . . . . .	841, 849, 850, 852	\_draw_softpath_clear: . . . . . 696, 707, 761, 1368, 1525, 1532, 1546, 1550	
\_draw_point_process_auxiv:nw . . . . . . .	841, 851, 853	\_draw_softpath_close_op:nn . . . . . . . .	1564, 1608, 1608, 1701, 1737, 1839
		\_draw_softpath_closepath: . . . . . . . . .	316, 543, 1560, 1560
		\l__draw_softpath_corneri_dim . . . . . . . .	1633, 1686, 1689, 1775
		\l__draw_softpath_cornerii_dim . . . . . . .	1633, 1687, 1690, 1766
		\g__draw_softpath_corners_bool . . . . . . .	1521, 1535, 1545, 1553, 1605, 1642, 1658



\l__draw_softpath_corners_bool ..	\__draw_softpath_rectangle_-
..... 796, 1544, 1554	opi:nn ..... 1597, 1608, 1624
\l__draw_softpath_curve_end_tl ..	\__draw_softpath_rectangle_-
..... 1632, 1763, 1782, 1831, 1842	opi:nnNnn ..... 1608, 1625, 1626
\__draw_softpath_curveto:nnnnnn ..	\__draw_softpath_rectangle_-
..... 310, 1560, 1569	opii:nn ..... 1598, 1608, 1628, 1629
\__draw_softpath_curveto_opi:nn ..	\__draw_softpath_restore: .....
.. 1573, 1608, 1610, 1698, 1736, 1799	..... 821, 1537, 1548
\__draw_softpath_curveto_-	\__draw_softpath_round_action:nn
opi:nnNnnNnn ..... 1608, 1611, 1612	..... 1640, 1664, 1684
\__draw_softpath_curveto_opii:nn	\__draw_softpath_round_action:Nnn
..... 1574, 1608, 1614, 1615, 1808	..... 1640, 1692, 1694
\__draw_softpath_curveto_-	\__draw_softpath_round_action_-
opiii:nn 1575, 1608, 1616, 1617, 1817	close: ..... 1640, 1702, 1720
\l__draw_softpath_first_tl .....	\__draw_softpath_round_action_-
..... 1633, 1649, 1666,	curveto:NnnNnn .. 1640, 1699, 1711
1667, 1677, 1696, 1697, 1723, 1727	\__draw_softpath_round_calc:NnnNnn
\g__draw_softpath_lastx_dim .....	..... 1640, 1740, 1755, 1761, 1824
..... 814, 822, 1556, 1565, 1589	\__draw_softpath_round_calc:nnnnnn
\l__draw_softpath_lastx_dim .....	..... 1640, 1772, 1789, 1794
..... 802, 814, 822	\__draw_softpath_round_calc:nnnnw
\l__draw_softpath_lastx_fp .....	..... 1640, 1791, 1795
.. 1633, 1647, 1668, 1716, 1778, 1785	\__draw_softpath_round_close:nn ..
\g__draw_softpath_lasty_dim .....	..... 1640, 1726, 1820
..... 815, 823, 1556, 1566, 1590	\__draw_softpath_round_close:w ..
\l__draw_softpath_lasty_dim .....	..... 1640, 1830, 1841, 1852
..... 803, 815, 823	\__draw_softpath_round_corners: ..
\l__draw_softpath_lasty_fp .....	..... 732, 1640, 1640
.. 1633, 1648, 1669, 1717, 1779, 1786	\__draw_softpath_round_end: .....
\__draw_softpath_lineto:nn .....	..... 1640, 1662, 1853
..... 291, 1560, 1578	\__draw_softpath_round_lookahead:NnnNnn
\__draw_softpath_lineto_op:nn ..	..... 1640, 1705, 1718, 1731
.. 1581, 1608, 1618, 1704, 1735, 1848	\__draw_softpath_round_loop:Nnn ..
\g__draw_softpath_main_tl .....	... 1640, 1652, 1660, 1681, 1691,
..... 1519, 1523, 1528,	1706, 1729, 1741, 1747, 1756, 1838
1534, 1539, 1542, 1651, 1653, 1859	\__draw_softpath_round_roundpoint:NnnNnnNnn
\l__draw_softpath_main_tl .....	..... 1640, 1746, 1752
..... 20, 1541, 1551, 1630,	\__draw_softpath_roundpoint:nn ..
1645, 1672, 1674, 1855, 1857, 1860	..... 264, 1560, 1601, 1607
\g__draw_softpath_move_bool .....	\__draw_softpath_roundpoint_-
..... 1558, 1587	op:nn .....
\l__draw_softpath_move_tl .....	.. 1604, 1608, 1622, 1623, 1663, 1745
..... 1633, 1650,	\__draw_softpath_save: 817, 1537, 1537
1673, 1676, 1724, 1826, 1849, 1856	\l__draw_softpath_tmp_tl .....
\__draw_softpath_moveto:nn .....	..... 1520, 1529, 1530
..... 278, 1560, 1583	\__draw_softpath_use: 737, 1525, 1525
\__draw_softpath_moveto_op:nn ..	\l__draw_tmp_box .....
..... 1586, 1608, 1620, 1670, 1828	..... 13, 49, 60,
\l__draw_softpath_part_tl .....	64, 66, 67, 68, 69, 85, 87, 88, 89, 90, 91
..... 1631, 1646,	\l__draw_tmp_seq .....
1675, 1678, 1680, 1714, 1769, 1858	..... 1873
\__draw_softpath_rectangle:nnnn ..	\__draw_transform:nnnn .....
..... 609, 1560, 1593	..... 1949, 1953, 1960
	\__draw_transform_invert:n .....
	..... 2002, 2006, 2019, 2030

<code>\__draw_transform_rotate:n</code> . . . . .	<code>\fp_compare_p:nNn</code> . . . . .
. . . . . 2112, 2113, 2114, 2120	. . . . . 1931, 1932, 1933, 1934
<code>\__draw_transform_rotate:nn</code> . . . . .	<code>\fp_const:Nn</code> . . . . .
. . . . . 2112, 2116, 2121, 2123	. . . . . 376, 377, 515, 516, 586, 1639
<code>\__draw_transform_shift:nn</code> . . . . .	<code>\fp_eval:n</code> . . . . .
. . . . . 1949, 1974, 1976, 2056	. 385, 386, 407, 414, 423, 875, 883,
<code>\__draw_transform_shift:nnnn</code> . . . . .	892, 917, 918, 919, 920, 921, 922,
. . . . . 1978, 1983, 1983	943, 948, 949, 956, 963, 964, 971,
<code>\__draw_transform_shift_absolute:nn</code> . . . . .	978, 980, 1001, 1006, 1007, 1008,
. . . . . 1923, 1942, 1944	1015, 1023, 1036, 1042, 1060, 1078,
<code>\__draw_transform_shift_invert:n</code> . . . . .	1083, 1090, 1091, 1112, 1118, 1119,
. . . . . 2002, 2035, 2052, 2077	1120, 1145, 1152, 1174, 1175, 1194,
<code>\__draw_transform_triangle:nnnnnn</code> . . . . .	1195, 1196, 1197, 1229, 1242, 1261,
. . . . . 2081, 2086	1265, 1266, 1889, 1954, 1955, 1956,
<code>\__draw_vec:nn</code> . . . . .	1957, 2008, 2037, 2113, 2117, 2118
. . . . . 1212, 1213, 1215, 1217, 1218	<code>\fp_new:N</code> . . . . . 205, 206, 513,
<code>\__draw_vec:nnn</code> . . . . . 1212, 1219, 1220	514, 1633, 1634, 1902, 1903, 1904, 1905
<code>\l__draw_xmax_dim</code> . . . . . 1481, 1488, 1496	<code>\fp_set:Nn</code> . . . . .
<code>\l__draw_xmin_dim</code> . . . . . 1481, 1489, 1497	. . . . . 344, 345, 399, 400, 480, 481,
<code>\l__draw_xshift_dim</code> . . . . .	1668, 1669, 1716, 1717, 1785, 1786,
. . . . . 62, 601, 1296, 1310, 1902, 1918,	1910, 1913, 1925, 1926, 1927, 1928
1946, 1979, 1985, 2048, 2062, 2070	<code>\fp_to_decimal:N</code> . . . . . 406, 413, 421
<code>\l__draw_xvec_x_dim</code> . . . . .	<code>\fp_to_dim:n</code> . . . . . 252, 253, 351,
. . . . . 1206, 1234, 1248, 1274	358, 365, 369, 387, 388, 434, 443,
<code>\l__draw_xvec_y_dim</code> . . . . .	511, 537, 549, 550, 551, 552, 553,
. . . . . 1206, 1235, 1252, 1277	554, 559, 560, 561, 562, 563, 564,
<code>\l__draw_ymax_dim</code> . . . . . 1481, 1490, 1498	569, 570, 571, 572, 573, 574, 579,
<code>\l__draw_ymin_dim</code> . . . . . 1481, 1491, 1499	580, 581, 582, 583, 584, 657, 658,
<code>\l__draw_yshift_dim</code> . . . . .	1452, 1801, 1805, 1810, 1814, 1870,
. . . . . 63, 601, 1302, 1310, 1902, 1919,	1878, 1883, 1987, 1995, 2058, 2067
1947, 1980, 1993, 2049, 2061, 2071	<code>\fp_use:N</code> . . . . . 54, 55, 56, 57, 586
<code>\l__draw_yvec_x_dim</code> . . . . .	<code>\fp_while_do:nNnn</code> . . . . . 401
. . . . . 1206, 1234, 1249, 1275	<code>\fp_zero:N</code> . . . . . 1647, 1648, 1911, 1912
<code>\l__draw_yvec_y_dim</code> . . . . .	<code>\c_one_fp</code> . . . . . 1931, 1934
. . . . . 1206, 1235, 1253, 1278	<code>\c_zero_fp</code> . . . . . 897, 1932, 1933
<code>\l__draw_zvec_x_dim</code> . . . . . 1206, 1250	
<code>\l__draw_zvec_y_dim</code> . . . . . 1206, 1254	
<b>E</b>	
<code>\end</code> . . . . . 198, 835	
exp commands:	
<code>\exp_after:wN</code> . . . . .	
479, 497, 1652, 1726, 1829, 1840, 1849	
<code>\exp_args:Ne</code> . . . . . 1889	
<code>\exp_args:NNNV</code> . . . . . 1382	
<code>\exp_not:N</code> . . . . . 1408, 1409,	
1771, 1799, 1808, 1817, 1826, 1829,	
1830, 1831, 1836, 1840, 1841, 1842	
<code>\exp_not:n</code> . . . . . 1460	
<b>F</b>	
fp commands:	
<code>\fp_compare:nNnTF</code> . . . . . 393, 403, 897	
<b>G</b>	
group commands:	
<code>\group_begin:</code> . . . . . 48, 84, 130, 139,	
519, 807, 1359, 1470, 1487, 1644, 1875	
<code>\group_end:</code> . . . . . 70, 92, 147, 150,	
522, 830, 1399, 1478, 1500, 1656, 1885	
<b>H</b>	
hbox commands:	
<code>\hbox_gset:Nw</code> . . . . . 137	
<code>\hbox_gset_end:</code> . . . . . 148	
<code>\hbox_set:Nn</code> . . . . . 49, 60, 85, 1418, 1431	
<code>\hbox_set:Nw</code> . . . . . 1361, 1376	
<code>\hbox_set_end:</code> . . . . . 1382, 1386	
<code>\hbox_to_wd:nn</code> . . . . . 1403	
<b>I</b>	
int commands:	
<code>\int_gincr:N</code> . . . . . 1360	

<code>\int_if_odd:nTF</code> . . . . .	979, 1016	property commands:	
<code>\int_new:N</code> . . . . .	1349	<code>\property_record:nn</code> . . . . .	1408, 1415
<code>\int_use:N</code> . . . . .	1409	<code>\ProvidesExplPackage</code> . . . . .	3
<b>K</b>		<b>Q</b>	
kernel internal commands:		quark commands:	
<code>\__kernel_quark_new_test:N</code> . . . . .	9	<code>\quark_new:N</code> . . . . .	7, 8
<b>M</b>		quark internal commands:	
mode commands:		<code>\q__draw_recursion_stop</code> . . . . .	7, 1655
<code>\mode_leave_vertical:</code> . . . . .	1397	<code>\q__draw_recursion_tail</code> . . . . .	7, 1654
msg commands:		<b>S</b>	
<code>\msg_error:nnn</code> . . . . .	115, 125, 729	scan commands:	
<code>\msg_new:nnnn</code> . . . . .	192, 195, 832	<code>\scan_new:N</code> . . . . .	5, 6
<b>P</b>		scan internal commands:	
<code>\pgfextractx</code> . . . . .	22	<code>\s__draw_mark</code> . . . . .	5,
<code>\pgfextracty</code> . . . . .	22	851, 853, 858, 861, 868, 872, 1792, 1796	
<code>\pgfgetlastxy</code> . . . . .	22	<code>\s__draw_stop</code> . . . . .	
<code>\pgfgettransform</code> . . . . .	52	.. 5, 844, 846, 851, 853, 858, 861,	
<code>\pgfgettransformentries</code> . . . . .	52	868, 872, 1792, 1796, 1832, 1843, 1852	
<code>\pgfinnerlinewidth</code> . . . . .	51	seq commands:	
<code>\pgflowlevel</code> . . . . .	53	<code>\seq_new:N</code> . . . . .	1887
<code>\pgflowlevelsynccm</code> . . . . .	53	<code>\seq_set_from_clist:Nn</code> . . . . .	1876
<code>\pgfpatharcto</code> . . . . .	6	<code>\seq_set_map:NNn</code> . . . . .	1877
<code>\pgfpatharctoprecomputed</code> . . . . .	6	<code>\seq_use:Nn</code> . . . . .	1882
<code>\pgfpathcosine</code> . . . . .	6	skip commands:	
<code>\pgfpathcurvebetweentime</code> . . . . .	6	<code>\skip_horizontal:n</code> . . . . .	1420, 1433
<code>\pgfpathcurvebetweentimecontinue</code> . . . . .	6	str commands:	
<code>\pgfpathparabola</code> . . . . .	6	<code>\str_if_eq:nnTF</code> 114, 131, 156, 173, 185	
<code>\pgfpathsine</code> . . . . .	6	<code>\str_if_eq_p:nn</code> . . . 108, 121, 122, 694	
<code>\pgfpointadd</code> . . . . .	22	<b>T</b>	
<code>\pgfpointborderellipse</code> . . . . .	22	TeX and L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> commands:	
<code>\pgfpointborderrectangle</code> . . . . .	22	<code>\@expl@finalise@setup@@</code> . . . . .	1406
<code>\pgfpointcylindrical</code> . . . . .	22	<code>\savepos</code> . . . . .	39
<code>\pgfpointdiff</code> . . . . .	22	tex commands:	
<code>\pgfpointorigin</code> . . . . .	22	<code>\tex_savepos:D</code> . . . . .	1405, 1412
<code>\pgfpointscale</code> . . . . .	22	tl commands:	
<code>\pgfpointspherical</code> . . . . .	22	<code>\tl_build_gbegin:N</code> . . . . .	1534, 1859
<code>\pgfqpoint</code> . . . . .	22	<code>\tl_build_gend:N</code> . . . . .	1539, 1651
<code>\pgfqpointpolar</code> . . . . .	22	<code>\tl_build_get_intermediate:NN</code> . . . . .	1527
<code>\pgfqpointscale</code> . . . . .	22	<code>\tl_build_gput_right:Nn</code> . . . . .	1523
<code>\pgfqpointxy</code> . . . . .	22	<code>\tl_clear:N</code> . . . . .	
<code>\pgfqpointxyz</code> . . . . .	22	456, 1645, 1646, 1649, 1650, 1677, 1678	
<code>\pgfsetinnerlinewidth</code> . . . . .	51	<code>\tl_if_blank:nTF</code> . . . . .	687
<code>\pgfsetinnerstrokecolor</code> . . . . .	51	<code>\tl_if_blank_p:n</code> . . . . .	693
<code>\pgftext</code> . . . . .	37	<code>\tl_if_empty:NTF</code> . . . . .	1666, 1696
<code>\pgftransformarcaxesatime</code> . . . . .	52	<code>\tl_if_empty_p:N</code> . . . . .	1723, 1724
<code>\pgftransformarrow</code> . . . . .	52	<code>\tl_new:N</code> . . . . .	97, 204, 1519,
<code>\pgftransformationadjustments</code> . . . . .	53	1520, 1630, 1631, 1632, 1637, 1638	
<code>\pgftransformcurveatime</code> . . . . .	52	<code>\tl_put_right:Nn</code> . . . . .	506, 510, 1672,
<code>\pgftransformlineatime</code> . . . . .	52	1674, 1680, 1714, 1769, 1855, 1857	
<code>\pgfviewboxscope</code> . . . . .	53	<code>\tl_set:Nn</code> . . . . .	98,
prg commands:		135, 502, 1667, 1676, 1697, 1763, 1826	
<code>\prg_do_nothing:</code> . . . . .	1167, 1178, 1181	<code>\tl_set_eq:NN</code> . . . . .	1540

token commands:			
\token_if_eq_meaning:NNTF .....		\use:n .....	51, 346,
.. 1663, 1670, 1698, 1701, 1704, 1745			382, 429, 534, 1458, 1822, 1834,
\token_if_eq_meaning_p:NN .....			1879, 1951, 1962, 2021, 2054, 2088
..... 1735, 1736, 1737		\use_i:nn .....	22
		\use_i:nnnn .....	1836
		\use_ii:nn .....	22
		\use_none:n .....	1849
U			
use commands:			
\use:N .....	751, 787		