

# File I

## Implementation

### 1 l3draw implementation

```

1 <*initex | package>
2 <@@=draw>
3 <*package>
4 \ProvidesExplPackage{l3draw}{2019-08-25}{}
5 {L3 Experimental core drawing support}
6 </package>
7 \RequirePackage { l3color }
8
  Everything else is in the sub-files!
8 </initex | package>

```

### 2 l3draw-boxes implementation

```

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

```

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

`\l__draw_tmp_box`

```

11 \box_new:N \l__draw_tmp_box

```

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

`\draw_box_use:N`  
`\__draw_box_use:Nnnnn`

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

```

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

```

```

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

```

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

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

```

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

```

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

70 `\</initex | package>`

### 3 l3draw-layers implementation

71 `\<*initex | package>`

72 <@@=draw>

### 3.1 User interface

`\draw_layer_new:n`

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

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

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

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

(End definition for `\l__draw_layer_tl`.)

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

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

(End definition for `\l__draw_layer_close_bool`.)

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

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

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

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

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

```

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

```

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

### 3.2 Internal cross-links

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

```

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

```

(End definition for `\__draw_layers_insert:.`)

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

```

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

```

```

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

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

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

```

## 4 l3draw-paths implementation

```

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

```

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

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

`\l__draw_path_tmp_tl` Scratch space.

```

\l__draw_path_tmpa_fp 173 \tl_new:N \l__draw_path_tmp_tl
\l__draw_path_tmpb_fp 174 \fp_new:N \l__draw_path_tmpa_fp
175 \fp_new:N \l__draw_path_tmpb_fp

```

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

## 4.1 Tracking paths

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

`\g__draw_path_lasty_dim` 176 \dim\_new:N \g\_\_draw\_path\_lastx\_dim  
177 \dim\_new:N \g\_\_draw\_path\_lasty\_dim

*(End definition for \g\_\_draw\_path\_lastx\_dim and \g\_\_draw\_path\_lasty\_dim.)*

`\g__draw_path_xmax_dim` The limiting size of a path.

`\g__draw_path_xmin_dim` 178 \dim\_new:N \g\_\_draw\_path\_xmax\_dim  
179 \dim\_new:N \g\_\_draw\_path\_xmin\_dim  
180 \dim\_new:N \g\_\_draw\_path\_ymax\_dim  
181 \dim\_new:N \g\_\_draw\_path\_ymin\_dim

*(End definition for \g\_\_draw\_path\_xmax\_dim and others.)*

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

`\__draw_path_reset_limits:` 182 \cs\_new\_protected:Npn \\_\_draw\_path\_update\_limits:nn #1#2  
183 {  
184 \dim\_gset:Nn \g\_\_draw\_path\_xmax\_dim  
185 { \dim\_max:nn \g\_\_draw\_path\_xmax\_dim {#1} }  
186 \dim\_gset:Nn \g\_\_draw\_path\_xmin\_dim  
187 { \dim\_min:nn \g\_\_draw\_path\_xmin\_dim {#1} }  
188 \dim\_gset:Nn \g\_\_draw\_path\_ymax\_dim  
189 { \dim\_max:nn \g\_\_draw\_path\_ymax\_dim {#2} }  
190 \dim\_gset:Nn \g\_\_draw\_path\_ymin\_dim  
191 { \dim\_min:nn \g\_\_draw\_path\_ymin\_dim {#2} }  
192 \bool\_if:NT \l\_draw\_bb\_update\_bool  
193 {  
194 \dim\_gset:Nn \g\_\_draw\_xmax\_dim  
195 { \dim\_max:nn \g\_\_draw\_xmax\_dim {#1} }  
196 \dim\_gset:Nn \g\_\_draw\_xmin\_dim  
197 { \dim\_min:nn \g\_\_draw\_xmin\_dim {#1} }  
198 \dim\_gset:Nn \g\_\_draw\_ymax\_dim  
199 { \dim\_max:nn \g\_\_draw\_ymax\_dim {#2} }  
200 \dim\_gset:Nn \g\_\_draw\_ymin\_dim  
201 { \dim\_min:nn \g\_\_draw\_ymin\_dim {#2} }  
202 }  
203 }  
204 \cs\_new\_protected:Npn \\_\_draw\_path\_reset\_limits:  
205 {  
206 \dim\_gset:Nn \g\_\_draw\_path\_xmax\_dim { -\c\_max\_dim }  
207 \dim\_gset:Nn \g\_\_draw\_path\_xmin\_dim { \c\_max\_dim }  
208 \dim\_gset:Nn \g\_\_draw\_path\_ymax\_dim { -\c\_max\_dim }  
209 \dim\_gset:Nn \g\_\_draw\_path\_ymin\_dim { \c\_max\_dim }  
210 }

*(End definition for \\_\_draw\_path\_update\_limits:nn and \\_\_draw\_path\_reset\_limits:.)*

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

211 \cs\_new\_protected:Npn \\_\_draw\_path\_update\_last:nn #1#2  
212 {  
213 \dim\_gset:Nn \g\_\_draw\_path\_lastx\_dim {#1}  
214 \dim\_gset:Nn \g\_\_draw\_path\_lasty\_dim {#2}  
215 }

(End definition for \\_draw\_path\_update\_last:nn.)

## 4.2 Corner arcs

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

\l\_\_draw\_corner\_xarc\_dim The two arcs in use.

```

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

```

(End definition for \l\_\_draw\_corner\_xarc\_dim and \l\_\_draw\_corner\_yarc\_dim.)

\l\_\_draw\_corner\_arc\_bool A flag to speed up the repeated checks.

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

(End definition for \l\_\_draw\_corner\_arc\_bool.)

\draw\_path\_corner\_arc:nn Calculate the arcs, check they are non-zero.

```

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

```

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

\\_\_draw\_path\_mark\_corner: Mark up corners for arc post-processing.

```

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

```

(End definition for \\_\_draw\_path\_mark\_corner:.)

### 4.3 Basic path constructions

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

```

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

```

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

\draw\_path\_close: A simple wrapper.

```

282 \cs_new_protected:Npn \draw_path_close:
283 {

```



```

284     \__draw_path_mark_corner:
285     \__draw_softpath_closepath:
286 }

```

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

## 4.4 Canvas path constructions

```

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

```

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

## 4.5 Computed curves

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

```

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

```

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

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

and

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

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

```

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

```

```

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

```

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

`\draw_path_arc:nnn` Drawing an arc means dividing the total curve required into sections: using Bézier curves we can cover at most  $90^\circ$  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  $115^\circ$ .

`\draw_path_arc:nnnn`

`\__draw_path_arc:nnnn`

`\__draw_path_arc:nnNnn`

`\__draw_path_arc_auxi:nnnnNnn`

`\__draw_path_arc_auxi:fnnnNnn`

`\__draw_path_arc_auxi:fnfnNnn`

`\__draw_path_arc_auxii:nnnNnnnn`

`\__draw_path_arc_auxiii:nn`

`\__draw_path_arc_auxiv:nnnn`

`\__draw_path_arc_auxv:nn`

`\__draw_path_arc_auxvi:nn`

`\__draw_path_arc_add:nnnn`

`\l__draw_path_arc_delta_fp`

`\l__draw_path_arc_start_fp`

`\c__draw_path_arc_90_fp`

`\c__draw_path_arc_60_fp`

```

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

```

```

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

```

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

```

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

```

```

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

```

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

```

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

```

The first control point.

```

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

```

The end point: simple arithmetic.

```

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

```

```

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

```

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

```

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

```

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

`\draw_path_arc_axes:nnnn` A simple wrapper.

```

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

```

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

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

```

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

```

```

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

```

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

`\draw_path_circle:nn` A shortcut.

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

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

## 4.6 Rectangles

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

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

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

`\draw_path_rectangle_corners:nn` Another shortcut wrapper.

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

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

## 4.7 Grids

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

```

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

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

```



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

## 4.8 Using paths

Actions to pass to the driver.

```

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

```

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

Actions handled at the macro layer.

```

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

```

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

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

```

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

```

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

```

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

```

```

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

```

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

```

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

```

```

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

```

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

## 4.9 Scoping paths

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

```

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

```

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

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

```

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

```

```

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

```

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

```

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

```

## 5 l3draw-points implementation

```

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

```

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

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

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

- `\pgfpointcylindrical`, `\pgfpointspherical`: The usefulness of these commands is not currently clear.
- `\pgfpointborderrectangle`, `\pgfpointborderellipse`: To be revisited once the semantics and use cases are clear.
- `\pgfqpoint`, `\pgfqpointscale`, `\pgfqpointpolar`, `\pgfqpointxy`, `\pgfqpointxyz`: The expandable approach taken in the code here, along with the absolute requirement for  $\varepsilon$ -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  $y$  co-ordinates. The first argument here should itself absorb two arguments. There is also a version to deal with two co-ordinates: common enough to justify a separate function.

```

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

```

```

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

```

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

```

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

```

## 5.2 Polar co-ordinates

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

```

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

```

## 5.3 Point expression arithmetic

These functions all take point expressions as arguments.

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

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

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

```

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

```

## 5.4 Intersection calculations

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

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

and

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

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

```

868 \cs_new:Npn \draw_point_intersect_lines:nnnn #1#2#3#4
869 {
870   \__draw_point_process:nnnnn
871   { \__draw_point_intersect_lines:nnnnnnnn }
872   {#1} {#2} {#3} {#4}
873 }

```

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

```

#1 x1
#2 y1
#3 x2
#4 y2
#5 x3
#6 y3
#7 x4
#8 y4

```

so now just have to do all of the calculation.

```

874 \cs_new:Npn \__draw_point_intersect_lines:nnnnnnnn #1#2#3#4#5#6#7#8
875 {
876   \__draw_point_intersect_lines_aux:ffffff
877   { \fp_eval:n { #1 * #4 - #2 * #3 } }
878   { \fp_eval:n { #5 * #8 - #6 * #7 } }
879   { \fp_eval:n { #1 - #3 } }
880   { \fp_eval:n { #5 - #7 } }
881   { \fp_eval:n { #2 - #4 } }
882   { \fp_eval:n { #6 - #8 } }
883 }
884 \cs_new:Npn \__draw_point_intersect_lines_aux:nnnnnn #1#2#3#4#5#6
885 {
886   \__draw_point_to_dim:n
887   {
888     ( #2 * #3 - #1 * #4 , #2 * #5 - #1 * #6 )
889     / ( #4 * #5 - #6 * #3 )

```

```

890     }
891   }
892   \cs_generate_variant:Nn \__draw_point_intersect_lines_aux:nnnnnn { ffffff }

```

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

```

893   \cs_new:Npn \draw_point_intersect_circles:nnnnn #1#2#3#4#5
894   {
895     \__draw_point_process:nnn
896     { \__draw_point_intersect_circles_auxi:nnnnnn {#2} {#4} {#5} }
897     {#1} {#3}
898   }
899   \cs_new:Npn \__draw_point_intersect_circles_auxi:nnnnnn #1#2#3#4#5#6#7
900   {
901     \__draw_point_intersect_circles_auxii:ffnnnnnn
902     { \fp_eval:n {#1} } { \fp_eval:n {#2} } {#4} {#5} {#6} {#7} {#3}
903   }

```

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

```

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

```



#7  $n$

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

```

904 \cs_new:Npn \__draw_point_intersect_circles_auxii:nnnnnnn #1#2#3#4#5#6#7
905 {
906   \__draw_point_intersect_circles_auxiii:ffnnnnnn
907   { \fp_eval:n { #5 - #3 } }
908   { \fp_eval:n { #6 - #4 } }
909   {#1} {#2} {#3} {#4} {#7}
910 }
911 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxii:nnnnnnn { ff }
912 \cs_new:Npn \__draw_point_intersect_circles_auxiii:nnnnnnn #1#2#3#4#5#6#7
913 {
914   \__draw_point_intersect_circles_auxiv:fnnnnnnn
915   { \fp_eval:n { sqrt( #1 * #1 + #2 * #2 ) } }
916   {#1} {#2} {#3} {#4} {#5} {#6} {#7}
917 }
918 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxiii:nnnnnnn { ff }

```

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

```

919 \cs_new:Npn \__draw_point_intersect_circles_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
920 {
921   \__draw_point_intersect_circles_auxv:ffnnnnnnnn
922   { \fp_eval:n { 1 / #1 } }
923   { \fp_eval:n { #4 * #4 } }
924   {#1} {#2} {#3} {#5} {#6} {#7} {#8}
925 }
926 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxiv:nnnnnnnn { f }
927 \cs_new:Npn \__draw_point_intersect_circles_auxv:nnnnnnnnn #1#2#3#4#5#6#7#8#9
928 {
929   \__draw_point_intersect_circles_auxvi:fnnnnnnnn
930   { \fp_eval:n { 0.5 * #1 * ( #2 + #3 * #3 - #6 * #6 ) } }
931   {#1} {#2} {#4} {#5} {#7} {#8} {#9}
932 }
933 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxv:nnnnnnnnn { ff }

```

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

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

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

```

934 \cs_new:Npn \__draw_point_intersect_circles_auxvi:nnnnnnnn #1#2#3#4#5#6#7#8
935 {
936   \__draw_point_intersect_circles_auxvii:fffnnnn
937   { \fp_eval:n { #1 * #2 } }
938   { \int_if_odd:nTF {#8} { 1 } { -1 } }
939   { \fp_eval:n { sqrt ( #3 - #1 * #1 ) * #2 } }
940   {#4} {#5} {#6} {#7}
941 }
942 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxvi:nnnnnnnn { f }
943 \cs_new:Npn \__draw_point_intersect_circles_auxvii:nnnnnnnn #1#2#3#4#5#6#7
944 {
945   \__draw_point_to_dim:n
946   { #6 + #4 * #1 + #2 * #3 * #5 , #7 + #5 * #1 + -1 * #2 * #3 * #4 }
947 }
948 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxvii:nnnnnnnn { fff }

```

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

Simple maths after expansion.

```

\draw_point_interpolate_line:nnn
\__draw_point_interpolate_line_aux:nnnnn
\__draw_point_interpolate_line_aux:fnnnn
\__draw_point_interpolate_line_aux:nnnnnn
\__draw_point_interpolate_line_aux:fnnnnnn
949 \cs_new:Npn \draw_point_interpolate_line:nnn #1#2#3
950 {
951   \__draw_point_process:nnn
952   { \__draw_point_interpolate_line_aux:fnnnn { \fp_eval:n {#1} } }
953   {#2} {#3}
954 }
955 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnn #1#2#3#4#5
956 {
957   \__draw_point_interpolate_line_aux:fnnnnn { \fp_eval:n { 1 - #1 } }
958   {#1} {#2} {#3} {#4} {#5}
959 }
960 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnn { f }
961 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnnn #1#2#3#4#5#6
962 { \__draw_point_to_dim:n { #2 * #3 + #1 * #5 , #2 * #4 + #1 * #6 } }
963 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnnn { f }

```

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

```

\draw_point_interpolate_distance:nnn
\__draw_point_interpolate_distance:nnnnn
\__draw_point_interpolate_distance:nnnnnn
\__draw_point_interpolate_distance:fnnnnnn
964 \cs_new:Npn \draw_point_interpolate_distance:nnn #1#2#3
965 {
966   \__draw_point_process:nn
967   { \__draw_point_interpolate_distance:nnnn {#1} {#3} }
968   {#2}
969 }
970 \cs_new:Npn \__draw_point_interpolate_distance:nnnn #1#2#3#4
971 {
972   \__draw_point_process:nn
973   {
974     \__draw_point_interpolate_distance:fnnnn
975     { \fp_eval:n {#1} } {#3} {#4}
976   }
977   { \draw_point_unit_vector:n { ( #2 ) - ( #3 , #4 ) } }

```

```

978 }
979 \cs_new:Npn \__draw_point_interpolate_distance:nnnnn #1#2#3#4#5
980 { \__draw_point_to_dim:n { #2 + #1 * #4 , #3 + #1 * #5 } }
981 \cs_generate_variant:Nn \__draw_point_interpolate_distance:nnnnn { f }

```

(End definition for \\_\_draw\_point\_to\_dim:n and others. These functions are documented on page ??.)

```

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

```

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

```

982 \cs_new:Npn \draw_point_interpolate_arcaxes:nnnnnn #1#2#3#4#5#6
983 {
984   \__draw_point_process:nnnn
985   { \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn {#1} {#5} {#6} }
986   {#2} {#3} {#4}
987 }
988 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
989 {
990   \__draw_point_interpolate_arcaxes_auxii:ffnnnnnnnn
991   { \fp_eval:n {#1} } {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
992 }

```

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

```

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

```

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

```

993 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
994 {
995   \__draw_point_interpolate_arcaxes_auxiii:ffnnnnnnnn
996   { \fp_eval:n { #1 * (#3) + ( 1 - #1 ) * (#2) } }
997   {#4} {#5} {#6} {#7} {#8} {#9}
998 }
999 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnnn { f }
1000 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnnn #1#2#3#4#5#6#7
1001 {
1002   \__draw_point_interpolate_arcaxes_auxiv:ffnnnnnnnn
1003   { \fp_eval:n { cosd (#1) } }
1004   { \fp_eval:n { sind (#1) } }
1005   {#2} {#3} {#4} {#5} {#6} {#7}

```

```

1006 }
1007 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnn { f }
1008 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
1009 {
1010   \__draw_point_to_dim:n
1011     { #3 + #1 * #5 + #2 * #7 , #4 + #1 * #6 + #2 * #8 }
1012 }
1013 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnn { ff }

```

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

```

\draw_point_interpolate_curve:nnnnn
\draw_point_interpolate_curve_auxi:nnnnnnnnn
\draw_point_interpolate_curve_auxii:nnnnnnnnn
\draw_point_interpolate_curve_auxiii:nnnnnnnnn
\draw_point_interpolate_curve_auxiiii:nnnnnnn
\draw_point_interpolate_curve_auxv:nnnnnnn
\draw_point_interpolate_curve_auxvi:nnnnnnn
\draw_point_interpolate_curve_auxvii:nnnnnnnnn
\draw_point_interpolate_curve_auxviii:nnnnnnn
\draw_point_interpolate_curve_auxviiii:nnnnnnn

```

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.

```

1014 \cs_new:Npn \draw_point_interpolate_curve:nnnnnn #1#2#3#4#5
1015 {
1016   \__draw_point_process:nnnnn
1017     { \__draw_point_interpolate_curve_auxi:nnnnnnnnn {#1} }
1018     {#2} {#3} {#4} {#5}
1019 }
1020 \cs_new:Npn \__draw_point_interpolate_curve_auxi:nnnnnnnnn #1#2#3#4#5#6#7#8#9
1021 {
1022   \__draw_point_interpolate_curve_auxii:nnnnnnnnn
1023     { \fp_eval:n {#1} }
1024     {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1025 }

```

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

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

In the second stage, we can drop the final point

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

and for the final stage only need one set of calculations

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

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

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

```

1026 \cs_new:Npn \__draw_point_interpolate_curve_auxiii:nnnnnnnnn
1027   #1#2#3#4#5#6#7#8#9
1028   {
1029     \__draw_point_interpolate_curve_auxiii:fnnnnn
1030     { \fp_eval:n { 1 - #1 } }
1031     {#1}
1032     { {#2} {#3} } { {#4} {#5} } { {#6} {#7} } { {#8} {#9} }
1033   }
1034 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxiii:nnnnnnnnn { f }
1035 % \begin{macrocode}
1036 % We need to do the first cycle, but haven't got enough arguments to keep
1037 % everything in play at once. So here we use a bit of argument re-ordering
1038 % and a single auxiliary to get the job done.
1039 % \begin{macrocode}
1040 \cs_new:Npn \__draw_point_interpolate_curve_auxiii:nnnnnn #1#2#3#4#5#6
1041   {
1042     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #3 #4
1043     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #4 #5
1044     \__draw_point_interpolate_curve_auxiv:nnnnnn {#1} {#2} #5 #6
1045     \prg_do_nothing:
1046     \__draw_point_interpolate_curve_auxvi:n { {#1} {#2} }
1047   }
1048 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxiii:nnnnnn { f }
1049 \cs_new:Npn \__draw_point_interpolate_curve_auxiv:nnnnnn #1#2#3#4#5#6
1050   {
1051     \__draw_point_interpolate_curve_auxv:ffw
1052     { \fp_eval:n { #1 * #3 + #2 * #5 } }
1053     { \fp_eval:n { #1 * #4 + #2 * #6 } }
1054   }
1055 \cs_new:Npn \__draw_point_interpolate_curve_auxv:nnw
1056   #1#2#3 \prg_do_nothing: #4#5
1057   {
1058     #3
1059     \prg_do_nothing:
1060     #4 { #5 {#1} {#2} }
1061   }
1062 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxv:nnw { ff }
1063 % \begin{macrocode}
1064 % Get the arguments back into the right places and to the second and
1065 % third cycles directly.
1066 % \begin{macrocode}
1067 \cs_new:Npn \__draw_point_interpolate_curve_auxvi:n #1
1068   { \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1 }
1069 \cs_new:Npn \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1#2#3#4#5#6#7#8
1070   {
1071     \__draw_point_interpolate_curve_auxviii:ffffnn
1072     { \fp_eval:n { #1 * #5 + #2 * #3 } }
1073     { \fp_eval:n { #1 * #6 + #2 * #4 } }

```

```

1074 { \fp_eval:n { #1 * #7 + #2 * #5 } }
1075 { \fp_eval:n { #1 * #8 + #2 * #6 } }
1076 {#1} {#2}
1077 }
1078 \cs_new:Npn \__draw_point_interpolate_curve_auxviii:nnnnnn #1#2#3#4#5#6
1079 {
1080   \__draw_point_to_dim:n
1081     { #5 * #3 + #6 * #1 , #5 * #4 + #6 * #2 }
1082 }
1083 \cs_generate_variant:Nn \__draw_point_interpolate_curve_auxviii:nnnnnn { ffff }

```

(End definition for \draw\_point\_interpolate\_curve:nnnn and others. These functions are documented on page ??.)

## 5.6 Vector support

As well as co-ordinates relative to the drawing

```

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

```

(End definition for \l\_\_draw\_xvec\_x\_dim and others.)

```

\draw_xvec:n Calculate the underlying position and store it.
\draw_yvec:n
\draw_zvec:n
\__draw_vec:nn
\__draw_vec:nnn

```

```

1090 \cs_new_protected:Npn \draw_xvec:n #1
1091 { \__draw_vec:nn { x } {#1} }
1092 \cs_new_protected:Npn \draw_yvec:n #1
1093 { \__draw_vec:nn { y } {#1} }
1094 \cs_new_protected:Npn \draw_zvec:n #1
1095 { \__draw_vec:nn { z } {#1} }
1096 \cs_new_protected:Npn \__draw_vec:nn #1#2
1097 {
1098   \__draw_point_process:nn { \__draw_vec:nnn {#1} } {#2}
1099 }
1100 \cs_new_protected:Npn \__draw_vec:nnn #1#2#3
1101 {
1102   \dim_set:cn { l__draw_ #1 vec_x_dim } {#2}
1103   \dim_set:cn { l__draw_ #1 vec_y_dim } {#3}
1104 }

```

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

Initialise the vectors.

```

1105 \draw_xvec:n { 1cm , 0cm }
1106 \draw_yvec:n { 0cm , 1cm }
1107 \draw_zvec:n { -0.385cm , -0.385cm }

```

```

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

```

```

1108 \cs_new:Npn \draw_point_vec:nn #1#2
1109 { \__draw_point_vec:ff { \fp_eval:n {#1} } { \fp_eval:n {#2} } }

```

```

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

```

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

```

\draw_point_vec_polar:nn
\draw_point_vec_polar:nnn
\__draw_point_vec_polar:nnn
\__draw_point_vec_polar:fnn
\cs_new:Npn \draw_point_vec_polar:nn #1#2
1138 { \draw_point_vec_polar:nnn {#1} {#1} {#2} }
1139 \cs_new:Npn \draw_point_vec_polar:nnn #1#2#3
1140 { \__draw_draw_vec_polar:fnn { \fp_eval:n {#3} } {#1} {#2} }
1141 \cs_new:Npn \__draw_draw_vec_polar:nnn #1#2#3
1142 {
1143   \__draw_point_to_dim:n
1144   {
1145     cosd(#1) * (#2) * \l__draw_xvec_x_dim ,
1146     sind(#1) * (#3) * \l__draw_yvec_y_dim
1147   }
1148 }
1149 \cs_generate_variant:Nn \__draw_draw_vec_polar:nnn { f }
1150

```

(End definition for \draw\_point\_vec\_polar:nn, \draw\_point\_vec\_polar:nnn, and \\_\_draw\_point\_vec\_polar:nnn. These functions are documented on page ??.)

## 5.7 Transformations

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

```

1151 \cs_new:Npn \draw_point_transform:n #1

```

```

1152 {
1153   \__draw_point_process:nn
1154   { \__draw_point_transform:nn } {#1}
1155 }
1156 \cs_new:Npn \__draw_point_transform:nn #1#2
1157 {
1158   \bool_if:NTF \l__draw_matrix_active_bool
1159   {
1160     \__draw_point_to_dim:n
1161     {
1162       (
1163         \l__draw_matrix_a_fp * #1
1164         + \l__draw_matrix_c_fp * #2
1165         + \l__draw_xshift_dim
1166       )
1167       ,
1168       (
1169         \l__draw_matrix_b_fp * #1
1170         + \l__draw_matrix_d_fp * #2
1171         + \l__draw_yshift_dim
1172       )
1173     }
1174   }
1175   {
1176     \__draw_point_to_dim:n
1177     {
1178       (#1, #2)
1179       + ( \l__draw_xshift_dim , \l__draw_yshift_dim )
1180     }
1181   }
1182 }

```

(End 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.

```

1183 \cs_new:Npn \__draw_point_transform_noshift:n #1
1184 {
1185   \__draw_point_process:nn
1186   { \__draw_point_transform_noshift:nn } {#1}
1187 }
1188 \cs_new:Npn \__draw_point_transform_noshift:nn #1#2
1189 {
1190   \bool_if:NTF \l__draw_matrix_active_bool
1191   {
1192     \__draw_point_to_dim:n
1193     {
1194       (
1195         \l__draw_matrix_a_fp * #1
1196         + \l__draw_matrix_c_fp * #2
1197       )
1198       ,
1199       (
1200         \l__draw_matrix_b_fp * #1

```



```

1201         + \l__draw_matrix_d_fp * #2
1202     )
1203 }
1204 }
1205 { \__draw_point_to_dim:n { (#1, #2) } }
1206 }

```

(End definition for \\_\_draw\_point\_transform\_noshift:n and \\_\_draw\_point\_transform\_noshift:nn.)

```
1207 \</initex | package>
```

## 6 l3draw-scopes implementation

```
1208 \*initex | package>
```

```
1209 \@@=draw>
```

### 6.1 Drawing environment

\g\_\_draw\_xmax\_dim Used to track the overall (official) size of the image created: may not actually be the natural size of the content.

```

\g__draw_xmin_dim
\g__draw_ymax_dim 1210 \dim_new:N \g__draw_xmax_dim
\g__draw_ymin_dim 1211 \dim_new:N \g__draw_xmin_dim
1212 \dim_new:N \g__draw_ymax_dim
1213 \dim_new:N \g__draw_ymin_dim

```

(End definition for \g\_\_draw\_xmax\_dim and others.)

\l\_draw\_bb\_update\_bool Flag to indicate that a path (or similar) should update the bounding box of the drawing.

```
1214 \bool_new:N \l_draw_bb_update_bool
```

(End definition for \l\_draw\_bb\_update\_bool. This variable is documented on page ??.)

\l\_\_draw\_layer\_main\_box Box for setting the drawing itself and the top-level layer.

```

1215 \box_new:N \l__draw_main_box
1216 \box_new:N \l__draw_layer_main_box

```

(End definition for \l\_\_draw\_layer\_main\_box.)

\g\_\_draw\_id\_int The drawing number.

```
1217 \int_new:N \g__draw_id_int
```

(End definition for \g\_\_draw\_id\_int.)

\\_\_draw\_reset\_bb: A simple auxiliary.

```

1218 \cs_new_protected:Npn \__draw_reset_bb:
1219 {
1220   \dim_gset:Nn \g__draw_xmax_dim { -\c_max_dim }
1221   \dim_gset:Nn \g__draw_xmin_dim { \c_max_dim }
1222   \dim_gset:Nn \g__draw_ymax_dim { -\c_max_dim }
1223   \dim_gset:Nn \g__draw_ymin_dim { \c_max_dim }
1224 }

```

(End definition for \\_\_draw\_reset\_bb:.)

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

```

1225 \cs_new_protected:Npn \draw_begin:
1226 {
1227   \group_begin:
1228     \int_gincr:N \g__draw_id_int
1229     \hbox_set:Nw \l__draw_main_box
1230       \__draw_backend_begin:
1231       \__draw_reset_bb:
1232       \__draw_path_reset_limits:
1233       \bool_set_true:N \l_draw_bb_update_bool
1234       \draw_transform_matrix_reset:
1235       \draw_transform_shift_reset:
1236       \__draw_softpath_clear:
1237       \draw_linewidth:n { \l_draw_default_linewidth_dim }
1238       \draw_color:n { . }
1239       \draw_nonzero_rule:
1240       \draw_cap_but:
1241       \draw_join_miter:
1242       \draw_miterlimit:n { 10 }
1243       \draw_dash_pattern:nn { } { 0cm }
1244       \hbox_set:Nw \l__draw_layer_main_box
1245   }
1246 \cs_new_protected:Npn \draw_end:
1247 {
1248   \exp_args:NNNV \hbox_set_end:
1249   \clist_set:Nn \l_draw_layers_clist \l_draw_layers_clist
1250   \__draw_layers_insert:
1251   \__draw_backend_end:
1252   \hbox_set_end:
1253   \dim_compare:nNnT \g__draw_xmin_dim = \c_max_dim
1254   {
1255     \dim_gzero:N \g__draw_xmax_dim
1256     \dim_gzero:N \g__draw_xmin_dim
1257     \dim_gzero:N \g__draw_ymax_dim
1258     \dim_gzero:N \g__draw_ymin_dim
1259   }
1260   \hbox_set:Nn \l__draw_main_box
1261   {
1262     \skip_horizontal:n { -\g__draw_xmin_dim }
1263     \box_move_down:nn { \g__draw_ymin_dim }
1264     { \box_use_drop:N \l__draw_main_box }
1265   }
1266   \box_set_ht:Nn \l__draw_main_box
1267   { \g__draw_ymax_dim - \g__draw_ymin_dim }
1268   \box_set_dp:Nn \l__draw_main_box { Opt }
1269   \box_set_wd:Nn \l__draw_main_box
1270   { \g__draw_xmax_dim - \g__draw_xmin_dim }
1271   \mode_leave_vertical:

```

```

1272     \box_use_drop:N \l__draw_main_box
1273     \group_end:
1274 }

```

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

## 6.2 Scopes

`\l__draw_linewidth_dim` Storage for local variables.  
`\l__draw_fill_color_tl` 1275 `\dim_new:N \l__draw_linewidth_dim`  
`\l__draw_stroke_color_tl` 1276 `\tl_new:N \l__draw_fill_color_tl`  
1277 `\tl_new:N \l__draw_stroke_color_tl`

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

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

```

\draw_scope_begin: 1278 \cs_new_protected:Npn \draw_scope_begin:
1279 {
1280     \__draw_backend_scope_begin:
1281     \group_begin:
1282     \dim_set_eq:NN \l__draw_linewidth_dim \g__draw_linewidth_dim
1283     \draw_path_scope_begin:
1284 }
1285 \cs_new_protected:Npn \draw_scope_end:
1286 {
1287     \draw_path_scope_end:
1288     \dim_gset_eq:NN \g__draw_linewidth_dim \l__draw_linewidth_dim
1289     \group_end:
1290     \__draw_backend_scope_end:
1291 }

```

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

`\l__draw_xmax_dim` Storage for the bounding box.  
`\l__draw_xmin_dim` 1292 `\dim_new:N \l__draw_xmax_dim`  
`\l__draw_ymax_dim` 1293 `\dim_new:N \l__draw_xmin_dim`  
`\l__draw_ymin_dim` 1294 `\dim_new:N \l__draw_ymax_dim`  
1295 `\dim_new:N \l__draw_ymin_dim`

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

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

```

\__draw_scope_bb_end: 1296 \cs_new_protected:Npn \__draw_scope_bb_begin:
1297 {
1298     \group_begin:
1299     \dim_set_eq:NN \l__draw_xmax_dim \g__draw_xmax_dim
1300     \dim_set_eq:NN \l__draw_xmin_dim \g__draw_xmin_dim
1301     \dim_set_eq:NN \l__draw_ymax_dim \g__draw_ymax_dim
1302     \dim_set_eq:NN \l__draw_ymin_dim \g__draw_ymin_dim
1303     \__draw_reset_bb:
1304 }
1305 \cs_new_protected:Npn \__draw_scope_bb_end:
1306 {
1307     \dim_gset_eq:NN \g__draw_xmax_dim \l__draw_xmax_dim

```

```

1308     \dim_gset_eq:NN \g__draw_xmin_dim \l__draw_xmin_dim
1309     \dim_gset_eq:NN \g__draw_ymax_dim \l__draw_ymax_dim
1310     \dim_gset_eq:NN \g__draw_ymin_dim \l__draw_ymin_dim
1311   \group_end:
1312 }

```

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

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

```

\draw_suspend_end:
1313 \cs_new_protected:Npn \draw_suspend_begin:
1314 {
1315   \__draw_scope_bb_begin:
1316   \draw_path_scope_begin:
1317   \draw_transform_matrix_reset:
1318   \draw_transform_shift_reset:
1319   \__draw_layers_save:
1320 }
1321 \cs_new_protected:Npn \draw_suspend_end:
1322 {
1323   \__draw_layers_restore:
1324   \draw_path_scope_end:
1325   \__draw_scope_bb_end:
1326 }

```

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

```

1327 </initex | package>

```

## 7 l3draw-softpath implementation

```

1328 <*initex | package>

```

```

1329 <@@=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.

```

1330 \tl_new:N \g__draw_softpath_main_tl

```

(End definition for `\g__draw_softpath_main_tl`.)

`\l__draw_softpath_internal_tl` The soft path itself.

```

1331 \tl_new:N \l__draw_softpath_internal_tl

```

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

\g\_\_draw\_softpath\_corners\_bool Allow for optimised path use.

1332 \bool\_new:N \g\_\_draw\_softpath\_corners\_bool

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

\\_\_draw\_softpath\_add:n

\\_\_draw\_softpath\_add:o

\\_\_draw\_softpath\_add:x

1333 \cs\_new\_protected:Npn \\_\_draw\_softpath\_add:n

1334 { \tl\_build\_gput\_right:Nn \g\_\_draw\_softpath\_main\_tl }

1335 \cs\_generate\_variant:Nn \\_\_draw\_softpath\_add:n { o, x }

(End definition for \\_\_draw\_softpath\_add:n.)

\\_\_draw\_softpath\_use: Using and clearing is trivial.

\\_\_draw\_softpath\_clear:

1336 \cs\_new\_protected:Npn \\_\_draw\_softpath\_use:

1337 {

1338 \tl\_build\_get:NN \g\_\_draw\_softpath\_main\_tl \l\_\_draw\_softpath\_internal\_tl

1339 \l\_\_draw\_softpath\_internal\_tl

1340 }

1341 \cs\_new\_protected:Npn \\_\_draw\_softpath\_clear:

1342 {

1343 \tl\_build\_gclear:N \g\_\_draw\_softpath\_main\_tl

1344 \bool\_gset\_false:N \g\_\_draw\_softpath\_corners\_bool

1345 }

(End definition for \\_\_draw\_softpath\_use: and \\_\_draw\_softpath\_clear:.)

\g\_\_draw\_softpath\_lastx\_dim

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

\g\_\_draw\_softpath\_lasty\_dim

1346 \dim\_new:N \g\_\_draw\_softpath\_lastx\_dim

1347 \dim\_new:N \g\_\_draw\_softpath\_lasty\_dim

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

\g\_\_draw\_softpath\_move\_bool

Track if moving a point should update the close position.

1348 \bool\_new:N \g\_\_draw\_softpath\_move\_bool

1349 \bool\_gset\_true:N \g\_\_draw\_softpath\_move\_bool

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

\\_draw\_softpath\_curveto:nnnnnn

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

\\_\_draw\_softpath\_lineto:nn

1350 \cs\_new\_protected:Npn \\_\_draw\_softpath\_closepath:

\\_\_draw\_softpath\_moveto:nn

1351 {

\\_draw\_softpath\_rectangle:nnnn

1352 \\_\_draw\_softpath\_add:x

\\_draw\_softpath\_roundpoint:nn

1353 {

\\_draw\_softpath\_roundpoint:VV

1354 \\_\_draw\_softpath\_close\_op:nn

1355 { \dim\_use:N \g\_\_draw\_softpath\_lastx\_dim }

1356 { \dim\_use:N \g\_\_draw\_softpath\_lasty\_dim }

1357 }

1358 }

1359 \cs\_new\_protected:Npn \\_\_draw\_softpath\_curveto:nnnnnn #1#2#3#4#5#6

1360 {

1361 \\_\_draw\_softpath\_add:n

1362 {

1363 \\_\_draw\_softpath\_curveto\_opi:nn {#1} {#2}

```

1364         \_draw_softpath_curveto_opii:nn {#3} {#4}
1365         \_draw_softpath_curveto_opiii:nn {#5} {#6}
1366     }
1367 }
1368 \cs_new_protected:Npn \_draw_softpath_lineto:nn #1#2
1369 {
1370     \_draw_softpath_add:n
1371     { \_draw_softpath_lineto_op:nn {#1} {#2} }
1372 }
1373 \cs_new_protected:Npn \_draw_softpath_moveto:nn #1#2
1374 {
1375     \_draw_softpath_add:n
1376     { \_draw_softpath_moveto_op:nn {#1} {#2} }
1377     \bool_if:NT \g__draw_softpath_move_bool
1378     {
1379         \dim_gset:Nn \g__draw_softpath_lastx_dim {#1}
1380         \dim_gset:Nn \g__draw_softpath_lasty_dim {#2}
1381     }
1382 }
1383 \cs_new_protected:Npn \_draw_softpath_rectangle:nnnn #1#2#3#4
1384 {
1385     \_draw_softpath_add:n
1386     {
1387         \_draw_softpath_rectangle_opi:nn {#1} {#2}
1388         \_draw_softpath_rectangle_opii:nn {#3} {#4}
1389     }
1390 }
1391 \cs_new_protected:Npn \_draw_softpath_roundpoint:nn #1#2
1392 {
1393     \_draw_softpath_add:n
1394     { \_draw_softpath_roundpoint_op:nn {#1} {#2} }
1395     \bool_gset_true:N \g__draw_softpath_corners_bool
1396 }
1397 \cs_generate_variant:Nn \_draw_softpath_roundpoint:nn { VV }

```

(End definition for \\_draw\_softpath\_curveto:nnnnnn and others.)

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

```

\_draw_softpath_curveto_opi:nn 1398 \cs_new_protected:Npn \_draw_softpath_close_op:nn #1#2
\_draw_softpath_curveto_opii:nn 1399 { \_draw_backend_closepath: }
\_draw_softpath_curveto_opiii:nn 1400 \cs_new_protected:Npn \_draw_softpath_curveto_opi:nn #1#2
\_draw_softpath_lineto_op:nn 1401 { \_draw_softpath_curveto_opi:nnNnnNnn {#1} {#2} }
\_draw_softpath_moveto_op:nn 1402 \cs_new_protected:Npn \_draw_softpath_curveto_opi:nnNnnNnn #1#2#3#4#5#6#7#8
\_draw_softpath_roundpoint_op:nn 1403 { \_draw_backend_curveto:nnnnnn {#1} {#2} {#4} {#5} {#7} {#8} }
\_draw_softpath_rectangle_opi:nn 1404 \cs_new_protected:Npn \_draw_softpath_curveto_opii:nn #1#2
\_draw_softpath_rectangle_opii:nn 1405 { \_draw_softpath_curveto_opii:nn }
\_draw_softpath_curveto_opi:nnNnnNnn 1406 \cs_new_protected:Npn \_draw_softpath_curveto_opiii:nn #1#2
\_draw_softpath_rectangle_opi:nnNnn 1407 { \_draw_softpath_curveto_opiii:nn }
1408 \cs_new_protected:Npn \_draw_softpath_lineto_op:nn #1#2
1409 { \_draw_backend_lineto:nn {#1} {#2} }
1410 \cs_new_protected:Npn \_draw_softpath_moveto_op:nn #1#2
1411 { \_draw_backend_moveto:nn {#1} {#2} }

```

```

1412 \cs_new_protected:Npn \__draw_softpath_roundpoint_op:nn #1#2 { }
1413 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nn #1#2
1414 { \__draw_softpath_rectangle_opi:nnNnn {#1} {#2} }
1415 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nnNnn #1#2#3#4#5
1416 { \__draw_backend_rectangle:nnnn {#1} {#2} {#4} {#5} }
1417 \cs_new_protected:Npn \__draw_softpath_rectangle_opii:nn #1#2 { }

```

(End definition for \\_\_draw\_softpath\_close\_op:nn and others.)

## 7.2 Rounding soft path corners

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

\l\_\_draw\_softpath\_main\_tl For constructing the updated path.

```
1418 \tl_new:N \l__draw_softpath_main_tl
```

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

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

```
1419 \tl_new:N \l__draw_softpath_part_tl
```

```
1420 \tl_new:N \l__draw_softpath_curve_end_tl
```

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

\l\_\_draw\_softpath\_lastx\_fp \l\_\_draw\_softpath\_lasty\_fp Position tracking: the token list data may be entirely empty or set to a co-ordinate.

```
1421 \fp_new:N \l__draw_softpath_lastx_fp
```

```
1422 \fp_new:N \l__draw_softpath_lasty_fp
```

```
1423 \dim_new:N \l__draw_softpath_corneri_dim
```

```
1424 \dim_new:N \l__draw_softpath_cornerii_dim
```

```
1425 \tl_new:N \l__draw_softpath_first_tl
```

```
1426 \tl_new:N \l__draw_softpath_move_tl
```

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

\c\_\_draw\_softpath\_arc\_fp The magic constant.

```
1427 \fp_const:Nn \c__draw_softpath_arc_fp { 4/3 * (sqrt(2) - 1) }
```

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

\\_\_draw\_softpath\_round\_corners:

\\_\_draw\_softpath\_round\_loop:Nnn

\\_\_draw\_softpath\_round\_action:nn

\\_\_draw\_softpath\_round\_action:Nnn

\\_\_draw\_softpath\_round\_action\_curveto:NnnNnn

\\_\_draw\_softpath\_round\_action\_close:

\\_\_draw\_softpath\_round\_lookahead:NnnNnn

\\_\_draw\_softpath\_round\_roundpoint:NnnNnnNnn

\\_\_draw\_softpath\_round\_calc:NnnNnn

\\_\_draw\_softpath\_round\_calc:nnnnnn

\\_\_draw\_softpath\_round\_calc:fVnnnn

\\_\_draw\_softpath\_round\_calc:nnnnw

\\_\_draw\_softpath\_round\_close:nn

\\_\_draw\_softpath\_round\_close:w

\\_\_draw\_softpath\_round\_end:

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

```
1428 \cs_new_protected:Npn \__draw_softpath_round_corners:
```

```
1429 {
```

```
1430 \bool_if:NT \g__draw_softpath_corners_bool
```

```
1431 {
```

```
1432 \group_begin:
```

```
1433 \tl_clear:N \l__draw_softpath_main_tl
```

```
1434 \tl_clear:N \l__draw_softpath_part_tl
```

```
1435 \fp_zero:N \l__draw_softpath_lastx_fp
```

```
1436 \fp_zero:N \l__draw_softpath_lasty_fp
```

```
1437 \tl_clear:N \l__draw_softpath_first_tl
```

```
1438 \tl_clear:N \l__draw_softpath_move_tl
```

```

1439         \tl_build_get:NN \g__draw_softpath_main_tl \l__draw_softpath_internal_tl
1440         \exp_after:wN \__draw_softpath_round_loop:Nnn
1441         \l__draw_softpath_internal_tl
1442         \q_recursion_tail ? ?
1443         \q_recursion_stop
1444     \group_end:
1445 }
1446 \bool_gset_false:N \g__draw_softpath_corners_bool
1447 }

```

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.

```

1448 \cs_new_protected:Npn \__draw_softpath_round_loop:Nnn #1#2#3
1449 {
1450     \quark_if_recursion_tail_stop_do:Nn #1 { \__draw_softpath_round_end: }
1451     \token_if_eq_meaning:NNTF #1 \__draw_softpath_roundpoint_op:nn
1452     { \__draw_softpath_round_action:nn {#2} {#3} }
1453     {
1454         \tl_if_empty:NT \l__draw_softpath_first_tl
1455         { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1456         \fp_set:Nn \l__draw_softpath_lastx_fp {#2}
1457         \fp_set:Nn \l__draw_softpath_lasty_fp {#3}
1458         \token_if_eq_meaning:NNTF #1 \__draw_softpath_moveto_op:nn
1459         {
1460             \tl_put_right:No \l__draw_softpath_main_tl
1461             \l__draw_softpath_move_tl
1462             \tl_put_right:No \l__draw_softpath_main_tl
1463             \l__draw_softpath_part_tl
1464             \tl_set:Nn \l__draw_softpath_move_tl { #1 {#2} {#3} }
1465             \tl_clear:N \l__draw_softpath_first_tl
1466             \tl_clear:N \l__draw_softpath_part_tl
1467         }
1468         { \tl_put_right:Nn \l__draw_softpath_part_tl { #1 {#2} {#3} } }
1469         \__draw_softpath_round_loop:Nnn
1470     }
1471 }
1472 \cs_new_protected:Npn \__draw_softpath_round_action:nn #1#2
1473 {
1474     \dim_set:Nn \l__draw_softpath_corneri_dim {#1}
1475     \dim_set:Nn \l__draw_softpath_cornerii_dim {#2}
1476     \bool_lazy_and:nnTF
1477     { \dim_compare_p:nNn \l__draw_softpath_corneri_dim = { 0pt } }
1478     { \dim_compare_p:nNn \l__draw_softpath_cornerii_dim = { 0pt } }
1479     { \__draw_softpath_round_loop:Nnn }
1480     { \__draw_softpath_round_action:Nnn }
1481 }

```

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.

```

1482 \cs_new_protected:Npn \__draw_softpath_round_action:Nnn #1#2#3

```



```

1483 {
1484   \tl_if_empty:NT \l__draw_softpath_first_tl
1485   { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1486   \token_if_eq_meaning:NNTF #1 \__draw_softpath_curveto_opi:nn
1487   { \__draw_softpath_round_action_curveto:NnnNnn }
1488   {
1489     \token_if_eq_meaning:NNTF #1 \__draw_softpath_close_op:nn
1490     { \__draw_softpath_round_action_close: }
1491     {
1492       \token_if_eq_meaning:NNTF #1 \__draw_softpath_lineto_op:nn
1493       { \__draw_softpath_round_lookahead:NnnNnn }
1494       { \__draw_softpath_round_loop:Nnn }
1495     }
1496   }
1497   #1 {#2} {#3}
1498 }

```

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.

```

1499 \cs_new_protected:Npn \__draw_softpath_round_action_curveto:NnnNnn
1500   #1#2#3#4#5#6
1501   {
1502     \tl_put_right:Nn \l__draw_softpath_part_tl
1503     { #1 {#2} {#3} #4 {#5} {#6} }
1504     \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1505     \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1506     \__draw_softpath_round_lookahead:NnnNnn
1507   }
1508 \cs_new_protected:Npn \__draw_softpath_round_action_close:
1509   {
1510     \bool_lazy_and:nnTF
1511     { ! \tl_if_empty_p:N \l__draw_softpath_first_tl }
1512     { ! \tl_if_empty_p:N \l__draw_softpath_move_tl }
1513     {
1514       \exp_after:wN \__draw_softpath_round_close:nn
1515       \l__draw_softpath_first_tl
1516     }
1517     { \__draw_softpath_round_loop:Nnn }
1518   }

```

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.

```

1519 \cs_new_protected:Npn \__draw_softpath_round_lookahead:NnnNnn #1#2#3#4#5#6
1520   {
1521     \bool_lazy_any:nTF
1522     {
1523       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_lineto_op:nn }
1524       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_curveto_opi:nn }
1525       { \token_if_eq_meaning_p:NN #4 \__draw_softpath_close_op:nn }
1526     }
1527     {
1528       \__draw_softpath_round_calc:NnnNnn
1529       \__draw_softpath_round_loop:Nnn
1530       {#5} {#6}

```

```

1531     }
1532     {
1533         \token_if_eq_meaning:NNTF #4 \__draw_softpath_roundpoint_op:nn
1534         { \__draw_softpath_round_roundpoint:NnnNnnNnn }
1535         { \__draw_softpath_round_loop:Nnn }
1536     }
1537     #1 {#2} {#3}
1538     #4 {#5} {#6}
1539 }
1540 \cs_new_protected:Npn \__draw_softpath_round_roundpoint:NnnNnnNnn
1541 #1#2#3#4#5#6#7#8#9
1542 {
1543     \__draw_softpath_round_calc:NnnNnn
1544     \__draw_softpath_round_loop:Nnn
1545     {#8} {#9}
1546     #1 {#2} {#3}
1547     #4 {#5} {#6} #7 {#8} {#9}
1548 }

```

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

```

1549 \cs_new_protected:Npn \__draw_softpath_round_calc:NnnNnn #1#2#3#4#5#6
1550 {
1551     \tl_set:Nx \l__draw_softpath_curve_end_tl
1552     {
1553         \draw_point_interpolate_distance:nnn
1554         \l__draw_softpath_cornerii_dim
1555         { #5 , #6 } { #2 , #3 }
1556     }
1557     \tl_put_right:Nx \l__draw_softpath_part_tl
1558     {
1559         \exp_not:N #4
1560         \__draw_softpath_round_calc:fVnnnn
1561         {
1562             \draw_point_interpolate_distance:nnn
1563             \l__draw_softpath_corneri_dim
1564             { #5 , #6 }
1565             {
1566                 \l__draw_softpath_lastx_fp ,
1567                 \l__draw_softpath_lasty_fp
1568             }
1569         }
1570         \l__draw_softpath_curve_end_tl
1571         {#5} {#6} {#2} {#3}
1572     }
1573     \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1574     \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1575     #1

```

```
1576 }
```

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

```
1577 \cs_new:Npn \__draw_softpath_round_calc:nnnnnn #1#2#3#4#5#6
1578 {
1579   \__draw_softpath_round_calc:nnnnw {#3} {#4} {#5} {#6}
1580   #1 \q_mark #2 \q_stop
1581 }
1582 \cs_generate_variant:Nn \__draw_softpath_round_calc:nnnnnn { fV }
```

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

```
1583 \cs_new:Npn \__draw_softpath_round_calc:nnnnw
1584 #1#2#3#4 #5 , #6 \q_mark #7 , #8 \q_stop
1585 {
1586   {#5} {#6}
1587   \exp_not:N \__draw_softpath_curveto_opi:nn
1588   {
1589     \fp_to_dim:n
1590     { #5 + \c__draw_softpath_arc_fp * ( #1 - #5 ) }
1591   }
1592   {
1593     \fp_to_dim:n
1594     { #6 + \c__draw_softpath_arc_fp * ( #2 - #6 ) }
1595   }
1596   \exp_not:N \__draw_softpath_curveto_opii:nn
1597   {
1598     \fp_to_dim:n
1599     { #7 + \c__draw_softpath_arc_fp * ( #1 - #7 ) }
1600   }
1601   {
1602     \fp_to_dim:n
1603     { #8 + \c__draw_softpath_arc_fp * ( #2 - #8 ) }
1604   }
1605   \exp_not:N \__draw_softpath_curveto_opiii:nn
1606   {#7} {#8}
1607 }
```

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.

```
1608 \cs_new_protected:Npn \__draw_softpath_round_close:nn #1#2
1609 {
1610   \use:x
1611   {
1612     \__draw_softpath_round_calc:NnnNnn
1613     {
1614       \tl_set:Nx \exp_not:N \l__draw_softpath_move_tl
1615       {
1616         \__draw_softpath_moveto_op:nn
1617         \exp_not:N \exp_after:wN
1618         \exp_not:N \__draw_softpath_round_close:w
1619         \exp_not:N \l__draw_softpath_curve_end_tl
```

```

1620         \exp_not:N \q_stop
1621     }
1622     \use:x
1623     {
1624         \exp_not:N \exp_not:N \exp_not:N \use_i:nnnn
1625         {
1626             \__draw_softpath_round_loop:Nnn
1627             \__draw_softpath_close_op:nn
1628             \exp_not:N \exp_after:wN
1629             \exp_not:N \__draw_softpath_round_close:w
1630             \exp_not:N \l__draw_softpath_curve_end_tl
1631             \exp_not:N \q_stop
1632         }
1633     }
1634 }
1635 {#1} {#2}
1636 \__draw_softpath_lineto_op:nn
1637 \exp_after:wN \use_none:n \l__draw_softpath_move_tl
1638 }
1639 }
1640 \cs_new:Npn \__draw_softpath_round_close:w #1 , #2 \q_stop { {#1} {#2} }

```

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

```

1641 \cs_new_protected:Npn \__draw_softpath_round_end:
1642 {
1643     \tl_put_right:No \l__draw_softpath_main_tl
1644     \l__draw_softpath_move_tl
1645     \tl_put_right:No \l__draw_softpath_main_tl
1646     \l__draw_softpath_part_tl
1647     \tl_build_gclear:N \g__draw_softpath_main_tl
1648     \__draw_softpath_add:o \l__draw_softpath_main_tl
1649 }

```

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

```

1650 </initex | package>

```

## 8 l3draw-state implementation

```

1651 <*initex | package>

```

```

1652 <@@=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.

```

1653 \dim_new:N \g__draw_linewidth_dim

```

(End definition for `\g__draw_linewidth_dim`.)

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

```

1654 \dim_new:N \l_draw_default_linewidth_dim
1655 \dim_set:Nn \l_draw_default_linewidth_dim { 0.4pt }

```

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

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

```

1656 \cs_new_protected:Npn \draw_linewidth:n #1
1657 {
1658   \dim_gset:Nn \g__draw_linewidth_dim { \fp_to_dim:n {#1} }
1659   \__draw_backend_linewidth:n \g__draw_linewidth_dim
1660 }

```

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

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

```

\l__draw_tmp_seq
1661 \cs_new_protected:Npn \draw_dash_pattern:nn #1#2
1662 {
1663   \group_begin:
1664     \seq_set_from_clist:Nn \l__draw_tmp_seq {#1}
1665     \seq_set_map:Nn \l__draw_tmp_seq \l__draw_tmp_seq
1666       { \fp_to_dim:n {##1} }
1667     \use:x
1668     {
1669       \__draw_backend_dash_pattern:nn
1670       { \seq_use:Nn \l__draw_tmp_seq { , } }
1671       { \fp_to_dim:n {#2} }
1672     }
1673   \group_end:
1674 }
1675 \seq_new:N \l__draw_tmp_seq

```

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

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

```

1676 \cs_new_protected:Npn \draw_miterlimit:n #1
1677 { \__draw_backend_miterlimit:n { \fp_eval:n {#1} } }

```

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

`\draw_cap_but:` All straight wrappers.

```

\draw_cap_rectangle:
1678 \cs_new_protected:Npn \draw_cap_but: { \__draw_backend_cap_but: }
\draw_cap_round:
1679 \cs_new_protected:Npn \draw_cap_rectangle: { \__draw_backend_cap_rectangle: }
\draw_evenodd_rule:
1680 \cs_new_protected:Npn \draw_cap_round: { \__draw_backend_cap_round: }
\draw_nonzero_rule:
1681 \cs_new_protected:Npn \draw_evenodd_rule: { \__draw_backend_evenodd_rule: }
\draw_join_bevel:
1682 \cs_new_protected:Npn \draw_nonzero_rule: { \__draw_backend_nonzero_rule: }
\draw_join_miter:
1683 \cs_new_protected:Npn \draw_join_bevel: { \__draw_backend_join_bevel: }
\draw_join_round:
1684 \cs_new_protected:Npn \draw_join_miter: { \__draw_backend_join_miter: }
1685 \cs_new_protected:Npn \draw_join_round: { \__draw_backend_join_round: }

```

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

`\l__draw_color_tmp_tl` Scratch space.

```

1686 \tl_new:N \l__draw_color_tmp_tl

```

(End definition for \l\_\_draw\_color\_tmp\_tl.)

```

\draw_color:n Much the same as for core color support but calling the relevant driver-level function.
\draw_color_fill:n
\draw_color_stroke:n
  \__draw_color:nn { \__draw_color:nn { fill } {#1} }
\__draw_color_aux:nn
\__draw_color_aux:Vn { \__draw_color:nn { stroke } {#1} }
  \__draw_color:nw
\__draw_select_cmyk:nw
\__draw_select_gray:nw
  \__draw_select_rgb:nw
\__draw_split_select:nw
1687 \cs_new_eq:NN \draw_color:n \color_select:n
1688 \cs_new_protected:Npn \draw_color_fill:n #1
1689 { \__draw_color:nn { fill } {#1} }
1690 \cs_new_protected:Npn \draw_color_stroke:n #1
1691 { \__draw_color:nn { stroke } {#1} }
1692 \cs_new_protected:Npn \__draw_color:nn #1#2
1693 {
1694   \color_parse:nN {#2} \l__draw_color_tmp_tl
1695   \__draw_color_aux:Vn \l__draw_color_tmp_tl {#1}
1696 }
1697 \cs_new_protected:Npn \__draw_color_aux:nn #1#2
1698 { \__draw_color:nw {#2} #1 \q_stop }
1699 \cs_generate_variant:Nn \__draw_color_aux:nn { V }
1700 \cs_new_protected:Npn \__draw_color:nw #1#2 ~ #3 \q_stop
1701 { \use:c { __draw_color_ #2 :nw } {#1} #3 \q_stop }
1702 \cs_new_protected:Npn \__draw_color_cmyk:nw #1#2 ~ #3 ~ #4 ~ #5 \q_stop
1703 { \use:c { __draw_backend_color_ #1 _cmyk:nnnn } {#2} {#3} {#4} {#5} }
1704 \cs_new_protected:Npn \__draw_color_gray:nw #1#2 \q_stop
1705 { \use:c { __draw_backend_color_ #1 _gray:n } {#2} }
1706 \cs_new_protected:Npn \__draw_color_rgb:nw #1#2 ~ #3 ~ #4 \q_stop
1707 { \use:c { __draw_backend_color_ #1 _rgb:nnn } {#2} {#3} {#4} }
1708 \cs_new_protected:Npn \__draw_color_spot:nw #1#2 ~ #3 \q_stop
1709 { \use:c { __draw_backend_color_ #1 _spot:nn } {#2} {#3} }

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

1710 </initex | package>

```

## 9 l3draw-transforms implementation

```

1711 <*initex | package>
1712 <@@=draw>

```

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

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

```

\l__draw_matrix_active_bool An internal flag to avoid redundant calculations.
1713 \bool_new:N \l__draw_matrix_active_bool

(End definition for \l__draw_matrix_active_bool.)

```

\l\_\_draw\_matrix\_a\_fp The active matrix and shifts.  
\l\_\_draw\_matrix\_b\_fp 1714 \fp\_new:N \l\_\_draw\_matrix\_a\_fp  
\l\_\_draw\_matrix\_c\_fp 1715 \fp\_new:N \l\_\_draw\_matrix\_b\_fp  
\l\_\_draw\_xshift\_dim 1716 \fp\_new:N \l\_\_draw\_matrix\_c\_fp  
\l\_\_draw\_yshift\_dim 1717 \fp\_new:N \l\_\_draw\_matrix\_d\_fp  
1718 \dim\_new:N \l\_\_draw\_xshift\_dim  
1719 \dim\_new:N \l\_\_draw\_yshift\_dim

(End definition for \l\_\_draw\_matrix\_a\_fp and others.)

\draw\_transform\_matrix\_reset: Fast resetting.

\draw\_transform\_shift\_reset: 1720 \cs\_new\_protected:Npn \draw\_transform\_matrix\_reset:  
1721 {  
1722 \fp\_set:Nn \l\_\_draw\_matrix\_a\_fp { 1 }  
1723 \fp\_zero:N \l\_\_draw\_matrix\_b\_fp  
1724 \fp\_zero:N \l\_\_draw\_matrix\_c\_fp  
1725 \fp\_set:Nn \l\_\_draw\_matrix\_d\_fp { 1 }  
1726 }  
1727 \cs\_new\_protected:Npn \draw\_transform\_shift\_reset:  
1728 {  
1729 \dim\_zero:N \l\_\_draw\_xshift\_dim  
1730 \dim\_zero:N \l\_\_draw\_yshift\_dim  
1731 }  
1732 \draw\_transform\_matrix\_reset:  
1733 \draw\_transform\_shift\_reset:

(End definition for \draw\_transform\_matrix\_reset: and \draw\_transform\_shift\_reset:. These functions are documented on page ??.)

\draw\_transform\_matrix\_absolute:nmm Setting the transform matrix is straight-forward, with just a bit of expansion to sort out.

\draw\_transform\_shift\_absolute:n With the mechanism active, the identity matrix is set.

\\_draw\_transform\_shift\_absolute:nn 1734 \cs\_new\_protected:Npn \draw\_transform\_matrix\_absolute:nmmn #1#2#3#4  
1735 {  
1736 \fp\_set:Nn \l\_\_draw\_matrix\_a\_fp {#1}  
1737 \fp\_set:Nn \l\_\_draw\_matrix\_b\_fp {#2}  
1738 \fp\_set:Nn \l\_\_draw\_matrix\_c\_fp {#3}  
1739 \fp\_set:Nn \l\_\_draw\_matrix\_d\_fp {#4}  
1740 \bool\_lazy\_all:nTF  
1741 {  
1742 { \fp\_compare\_p:nNn \l\_\_draw\_matrix\_a\_fp = \c\_one\_fp }  
1743 { \fp\_compare\_p:nNn \l\_\_draw\_matrix\_b\_fp = \c\_zero\_fp }  
1744 { \fp\_compare\_p:nNn \l\_\_draw\_matrix\_c\_fp = \c\_zero\_fp }  
1745 { \fp\_compare\_p:nNn \l\_\_draw\_matrix\_d\_fp = \c\_one\_fp }  
1746 }  
1747 { \bool\_set\_false:N \l\_\_draw\_matrix\_active\_bool }  
1748 { \bool\_set\_true:N \l\_\_draw\_matrix\_active\_bool }  
1749 }  
1750 \cs\_new\_protected:Npn \draw\_transform\_shift\_absolute:n #1  
1751 {  
1752 \\_draw\_point\_process:nn  
1753 { \\_draw\_transform\_shift\_absolute:nn } {#1}  
1754 }  
1755 \cs\_new\_protected:Npn \\_draw\_transform\_shift\_absolute:nn #1#2  
1756 {

```

1757     \dim_set:Nn \l__draw_xshift_dim {#1}
1758     \dim_set:Nn \l__draw_yshift_dim {#2}
1759   }

```

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

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

```

\__draw_transform:nnnn
\draw_transform_shift:n
\__draw_transform_shift:nn
1760 \cs_new_protected:Npn \draw_transform_matrix:nnnn #1#2#3#4
1761 {
1762   \use:x
1763   {
1764     \__draw_transform:nnnn
1765     { \fp_eval:n {#1} }
1766     { \fp_eval:n {#2} }
1767     { \fp_eval:n {#3} }
1768     { \fp_eval:n {#4} }
1769   }
1770 }
1771 \cs_new_protected:Npn \__draw_transform:nnnn #1#2#3#4
1772 {
1773   \use:x
1774   {
1775     \draw_transform_matrix_absolute:nnnn
1776     { #1 * \l__draw_matrix_a_fp + #2 * \l__draw_matrix_c_fp }
1777     { #1 * \l__draw_matrix_b_fp + #2 * \l__draw_matrix_d_fp }
1778     { #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_c_fp }
1779     { #3 * \l__draw_matrix_b_fp + #4 * \l__draw_matrix_d_fp }
1780   }
1781 }
1782 \cs_new_protected:Npn \draw_transform_shift:n #1
1783 {
1784   \__draw_point_process:nn
1785   { \__draw_transform_shift:nn } {#1}
1786 }
1787 \cs_new_protected:Npn \__draw_transform_shift:nn #1#2
1788 {
1789   \dim_set:Nn \l__draw_xshift_dim { \l__draw_xshift_dim + #1 }
1790   \dim_set:Nn \l__draw_yshift_dim { \l__draw_yshift_dim + #2 }
1791 }

```

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

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

```

\__draw_transform_invert:n
\__draw_transform_invert:f
\draw_transform_shift_invert:
1792 \cs_new_protected:Npn \draw_transform_matrix_invert:
1793 {
1794   \bool_if:NT \l__draw_matrix_active_bool
1795   {
1796     \__draw_transform_invert:f
1797     {
1798       \fp_eval:n
1799       {
1800         1 /

```



```

1801         (
1802             \l__draw_matrix_a_fp * \l__draw_matrix_d_fp
1803             - \l__draw_matrix_b_fp * \l__draw_matrix_c_fp
1804         )
1805     }
1806 }
1807 }
1808 }
1809 \cs_new_protected:Npn \__draw_transform_invert:n #1
1810 {
1811     \fp_set:Nn \l__draw_matrix_a_fp
1812     { \l__draw_matrix_d_fp * #1 }
1813     \fp_set:Nn \l__draw_matrix_b_fp
1814     { -\l__draw_matrix_b_fp * #1 }
1815     \fp_set:Nn \l__draw_matrix_c_fp
1816     { -\l__draw_matrix_c_fp * #1 }
1817     \fp_set:Nn \l__draw_matrix_d_fp
1818     { \l__draw_matrix_a_fp * #1 }
1819 }
1820 \cs_generate_variant:Nn \__draw_transform_invert:n { f }
1821 \cs_new_protected:Npn \draw_transform_shift_invert:
1822 {
1823     \dim_set:Nn \l__draw_xshift_dim { -\l__draw_xshift_dim }
1824     \dim_set:Nn \l__draw_yshift_dim { -\l__draw_yshift_dim }
1825 }

```

(End definition for \draw\_transform\_matrix\_invert:, \\_\_draw\_transform\_invert:n, and \draw\_transform\_shift\_invert:. These functions are documented on page ??.)

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

```

1826 \cs_new_protected:Npn \draw_transform_triangle:nnn #1#2#3
1827 {
1828     \__draw_point_process:nnn
1829     {
1830         \__draw_point_process:nn
1831         { \__draw_tranform_triangle:nnnnnn }
1832         {#1}
1833     }
1834     {#2} {#3}
1835 }
1836 \cs_new_protected:Npn \__draw_tranform_triangle:nnnnnn #1#2#3#4#5#6
1837 {
1838     \use:x
1839     {
1840         \draw_transform_matrix_absolute:nnnn
1841         { #3 - #1 }
1842         { #4 - #2 }
1843         { #5 - #1 }
1844         { #6 - #2 }
1845         \draw_transform_shift_absolute:n { #1 , #2 }
1846     }
1847 }

```

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

`\draw_transform_scale:n` Lots of shortcuts.

```

\draw_transform_xscale:n 1848 \cs_new_protected:Npn \draw_transform_scale:n #1
\draw_transform_yscale:n 1849 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { #1 } }
\draw_transform_xshift:n 1850 \cs_new_protected:Npn \draw_transform_xscale:n #1
\draw_transform_yshift:n 1851 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { 1 } }
\draw_transform_xslant:n 1852 \cs_new_protected:Npn \draw_transform_yscale:n #1
\draw_transform_yslant:n 1853 { \draw_transform_matrix:nnnn { 1 } { 0 } { 0 } { #1 } }
1854 \cs_new_protected:Npn \draw_transform_xshift:n #1
1855 { \draw_transform_shift:n { #1 , Opt } }
1856 \cs_new_protected:Npn \draw_transform_yshift:n #1
1857 { \draw_transform_shift:n { Opt , #1 } }
1858 \cs_new_protected:Npn \draw_transform_xslant:n #1
1859 { \draw_transform_matrix:nnnn { 1 } { 0 } { #1 } { 1 } }
1860 \cs_new_protected:Npn \draw_transform_yslant:n #1
1861 { \draw_transform_matrix:nnnn { 1 } { #1 } { 0 } { 1 } }

```

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

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

```

\__draw_transform_rotate:n 1862 \cs_new_protected:Npn \draw_transform_rotate:n #1
\__draw_transform_rotate:f 1863 { \__draw_transform_rotate:f { \fp_eval:n {#1} } }
\__draw_transform_rotate:nn 1864 \cs_new_protected:Npn \__draw_transform_rotate:n #1
\__draw_transform_rotate:ff 1865 {
1866   \__draw_transform_rotate:ff
1867   { \fp_eval:n { cosd(#1) } }
1868   { \fp_eval:n { sind(#1) } }
1869 }
1870 \cs_generate_variant:Nn \__draw_transform_rotate:n { f }
1871 \cs_new_protected:Npn \__draw_transform_rotate:nn #1#2
1872 { \draw_transform_matrix:nnnn {#1} {#2} { -#2 } { #1 } }
1873 \cs_generate_variant:Nn \__draw_transform_rotate:nn { ff }

```

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

1874 `\</initex | package>`

# Index

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

## B

`\begin` . . . 169, 783, 1035, 1039, 1063, 1066

bool commands:

`\bool_gset_eq:NN` . . . . . 764

`\bool_gset_false:N` . . . . . 1344, 1446

`\bool_gset_true:N` . . . . . 1349, 1395

`\bool_if:NTF` . . . . .

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

696, 700, 1158, 1190, 1377, 1430, 1794

`\bool_lazy_all:nTF` . . . . . 1740

`\bool_lazy_and:nnTF` . . . . .

. . . . . 223, 675, 1476, 1510

`\bool_lazy_any:nTF` . . . . . 1521

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

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

642, 643, 742, 1214, 1332, 1348, 1713

`\bool_set_eq:NN` . . . . . 756

`\bool_set_false:N` . . . . .

. . . . . 94, 226, 662, 663, 664, 1747

`\bool_set_true:N` . . . . . 96,

227, 668, 705, 709, 710, 1233, 1748

box commands:

`\box_dp:N` . . . . . 15, 65

`\box_gset_eq:NN` . . . . . 154

`\box_gset_wd:Nn` . . . . . 98, 131

`\box_ht:N` . . . . . 15, 67

`\box_if_exist:NTF` . . . . . 91

`\box_move_down:nn` . . . . . 1263

`\box_move_up:nn` . . . . . 49

`\box_new:N` . . . . . 11, 78, 79, 1215, 1216

`\box_set_dp:Nn` . . . . . 53, 1268

`\box_set_eq:NN` . . . . . 143

`\box_set_ht:Nn` . . . . . 52, 1266

`\box_set_wd:Nn` . . . . . 54, 126, 1269

`\box_use_drop:N` . . . . .

. . . . . 50, 55, 100, 127, 132, 1264, 1272

`\box_wd:N` . . . . . 15, 64, 66

## C

clist commands:

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

`\clist_map_inline:nn` . . . . . 665

`\clist_new:N` . . . . . 85, 87

`\clist_set:Nn` . . . . . 86, 1249

coffin commands:

`\coffin_typeset:Nnnnn` . . . . . 62

`\coffin_wd:N` . . . . . 64

color commands:

`\color_parse:nN` . . . . . 1694

`\color_select:n` . . . . . 1687

cs commands:

`\cs_generate_variant:Nn` . . . . .

. 418, 605, 638, 842, 850, 892, 911,

918, 926, 933, 942, 948, 960, 963,

981, 999, 1007, 1013, 1034, 1048,

1062, 1083, 1118, 1137, 1150, 1335,

1397, 1582, 1699, 1820, 1870, 1873

`\cs_if_exist:NTF` . . . . . 667

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

`\cs_new:Npn` . . . . 509, 519, 529, 539,

787, 793, 795, 797, 804, 806, 808,

816, 818, 821, 830, 835, 838, 840,

843, 844, 846, 848, 851, 853, 859,

868, 874, 884, 893, 899, 904, 912,

919, 927, 934, 943, 949, 955, 961,

964, 970, 979, 982, 988, 993, 1000,

1008, 1014, 1020, 1026, 1040, 1049,

1055, 1067, 1069, 1078, 1108, 1110,

1119, 1124, 1138, 1140, 1142, 1151,

1156, 1183, 1188, 1577, 1583, 1640

`\cs_new_eq:NN` . . . . . 1687

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

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

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

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

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

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

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

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

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

761, 1090, 1092, 1094, 1096, 1100,

1218, 1225, 1246, 1278, 1285, 1296,

1305, 1313, 1321, 1333, 1336, 1341,

1350, 1359, 1368, 1373, 1383, 1391,

1398, 1400, 1402, 1404, 1406, 1408,

1410, 1412, 1413, 1415, 1417, 1428,

1448, 1472, 1482, 1499, 1508, 1519,

1540, 1549, 1608, 1641, 1656, 1661,

1676, 1678, 1679, 1680, 1681, 1682,

1683, 1684, 1685, 1688, 1690, 1692,

1697, 1700, 1702, 1704, 1706, 1708,

1720, 1727, 1734, 1750, 1755, 1760,

1771, 1782, 1787, 1792, 1809, 1821,

1826, 1836, 1848, 1850, 1852, 1854,

1856, 1858, 1860, 1862, 1864, 1871

## D

### dim commands:

<code>\dim_abs:n</code> .....	594, 595
<code>\dim_compare:nNnTF</code> .....	601, 608, 721, 1253
<code>\dim_compare_p:nNn</code> .....	224, 225, 1477, 1478
<code>\dim_eval:n</code> .....	594, 595
<code>\dim_gset:Nn</code> .....	184, 186, 188, 190, 194, 196, 198, 200, 206, 207, 208, 209, 213, 214, 723, 1220, 1221, 1222, 1223, 1379, 1380, 1658
<code>\dim_gset_eq:NN</code> .....	768, 769, 770, 771, 772, 773, 774, 775, 1288, 1307, 1308, 1309, 1310
<code>\dim_gzero:N</code> ..	1255, 1256, 1257, 1258
<code>\dim_max:nn</code> .....	185, 189, 195, 199
<code>\dim_min:nn</code> .....	187, 191, 197, 201
<code>\dim_new:N</code> .....	176, 177, 178, 179, 180, 181, 216, 217, 734, 735, 736, 737, 738, 739, 740, 741, 1084, 1085, 1086, 1087, 1088, 1089, 1210, 1211, 1212, 1213, 1275, 1292, 1293, 1294, 1295, 1346, 1347, 1423, 1424, 1653, 1654, 1718, 1719
<code>\dim_set:Nn</code> .....	221, 222, 1102, 1103, 1474, 1475, 1655, 1757, 1758, 1789, 1790, 1823, 1824
<code>\dim_set_eq:NN</code> .....	746, 747, 748, 749, 750, 751, 752, 753, 1282, 1299, 1300, 1301, 1302
<code>\dim_step_inline:nnnn</code> .....	621, 629
<code>\dim_use:N</code> ..	721, 726, 728, 1355, 1356
<code>\dim_zero:N</code> .....	1729, 1730
<code>\c_max_dim</code> .....	206, 207, 208, 209, 721, 1220, 1221, 1222, 1223, 1253

### draw commands:

<code>\l_draw_bb_update_bool</code> .....	19, 192, 676, 1214, 1233
<code>\draw_begin:</code> .....	1225
<code>\draw_box_use:N</code> .....	12
<code>\draw_cap_but:</code> .....	1240, 1678
<code>\draw_cap_rectangle:</code> .....	1678
<code>\draw_cap_round:</code> .....	1678
<code>\draw_coffin_use:Nnn</code> .....	58
<code>\draw_color:n</code> .....	1238, 1687
<code>\draw_color_fill:n</code> .....	1687
<code>\draw_color_stroke:n</code> .....	1687
<code>\draw_dash_pattern:nn</code> ...	1243, 1661
<code>\l_draw_default_linewidth_dim</code> ...	103, 1237, 1654
<code>\draw_end:</code> .....	1225
<code>\draw_evenodd_rule:</code> .....	1678
<code>\draw_join_bevel:</code> .....	1678
<code>\draw_join_miter:</code> .....	1241, 1678
<code>\draw_join_round:</code> .....	1678

<code>\draw_layer_begin:n</code> .....	88
<code>\draw_layer_end:</code> .....	88
<code>\draw_layer_new:n</code> .....	73
<code>\l_draw_layers_clist</code> .....	85, 122, 139, 150, 1249
<code>\draw_linewidth:n</code> ....	103, 1237, 1656
<code>\draw_miterlimit:n</code> .....	1242, 1676
<code>\draw_nonzero_rule:</code> .....	1239, 1678
<code>\draw_path_arc:nnn</code> .....	343, 485
<code>\draw_path_arc:nnnn</code> .....	343
<code>\draw_path_arc_axes:nnnn</code> .....	482
<code>\draw_path_canvas_curveto:nnn</code> ..	287
<code>\draw_path_canvas_lineto:n</code> ....	287
<code>\draw_path_canvas_moveto:n</code> ....	287
<code>\draw_path_circle:nn</code> .....	550
<code>\draw_path_close:</code> .....	282, 578
<code>\draw_path_corner_arc:nn</code> .....	219
<code>\draw_path_curveto:nn</code> .....	300
<code>\draw_path_curveto:nnn</code> .....	238
<code>\draw_path_ellipse:nnn</code> .....	487, 551
<code>\draw_path_grid:nnnn</code> .....	589
<code>\draw_path_lineto:n</code> .....	238, 575, 576, 577, 627, 635
<code>\draw_path_moveto:n</code> .....	238, 574, 579, 626, 634
<code>\draw_path_rectangle:nn</code> ....	552, 588
<code>\draw_path_rectangle_corners:nn</code> ..	581
<code>\draw_path_scope_begin:</code> .....	743, 1283, 1316
<code>\draw_path_scope_end:</code> .....	743, 1287, 1324
<code>\draw_path_use:n</code> .....	644
<code>\draw_path_use_clear:n</code> .....	644
<code>\draw_point_interpolate_arcaxes:nnnnnn</code> .....	982
<code>\draw_point_interpolate_curve:nnnnnn</code> .....	1014
<code>\draw_point_interpolate_curve:nnnnnnnn</code> .....	1014
<code>\draw_point_interpolate_curve_-auxi:nnnnnnnnnn</code> .....	1014
<code>\draw_point_interpolate_curve_-auxii:nnnnnnnnnn</code> .....	1014
<code>\draw_point_interpolate_curve_-auxiii:nnnnnnnn</code> .....	1014
<code>\draw_point_interpolate_curve_-auxiv:nnnnnnnn</code> .....	1014
<code>\draw_point_interpolate_curve_-auxv:nnw</code> .....	1014
<code>\draw_point_interpolate_curve_-auxvi:n</code> .....	1014
<code>\draw_point_interpolate_curve_-auxvii:nnnnnnnnnn</code> .....	1014
<code>\draw_point_interpolate_curve_-auxviii:nnnnnnnn</code> .....	1014

\draw_point_interpolate_distance:nnn	682
..... 964, 1553, 1562	
\draw_point_interpolate_line:nnn	1399
..... 949	
\draw_point_intersect_circles:nnnnn	1403
..... 893	
\draw_point_intersect_lines:nnnn	1669
..... 868	
\draw_point_polar:nn	686
..... 844	
\draw_point_polar:nnn	1251
..... 426, 432, 436, 442, 844	
\draw_point_transform:n	1681
..... 23, 26, 29, 32, 242, 254, 270, 271,	
272, 304, 305, 431, 435, 491, 562, 1151	
\draw_point_unit_vector:n	1683
..... 851, 977	
\draw_point_vec:nn	1684
..... 1108	
\draw_point_vec:nnn	1685
..... 1108	
\draw_point_vec_polar:nn	1409
..... 1138	
\draw_point_vec_polar:nnn	1659
..... 1138	
\draw_scope_begin:	1677
..... 1278	
\draw_scope_end:	1411
..... 1285	
\draw_suspend_begin:	1682
..... 1313	
\draw_suspend_end:	1416
..... 1313	
\draw_transform_matrix:nnnn	130, 1280
..... 1760,	
1849, 1851, 1853, 1859, 1861, 1872	
\draw_transform_matrix_absolute:nnnn	133, 1290
..... 1734, 1775, 1840	
\draw_transform_matrix_invert:	12, 63
..... 1792	
\draw_transform_matrix_reset:	1687
..... 1234, 1317, 1720	
\draw_transform_rotate:n	1687
..... 1862	
\draw_transform_scale:n	1687
..... 1848	
\draw_transform_shift:n	1702
..... 1760, 1855, 1857	
\draw_transform_shift_absolute:n	1704
..... 1734, 1845	
\draw_transform_shift_invert:	1706
..... 1792	
\draw_transform_shift_reset:	1708
..... 1235, 1318, 1720	
\draw_transform_triangle:nnn	1686, 1694, 1695
..... 484, 1826	
\draw_transform_xscale:n	218, 226, 227, 231, 557
..... 1848	
\draw_transform_xshift:n	216, 221, 224, 234
..... 1848	
\draw_transform_xslant:n	216, 222, 225, 235
..... 1848	
\draw_transform_yscale:n	844
..... 1848	
\draw_transform_yshift:n	1141, 1142, 1150
..... 1848	
\draw_transform_yslant:n	1275
..... 1090, 1105	
\draw_xvec:n	1217, 1228
..... 1090, 1106	
\draw_yvec:n	84, 94, 96, 113
..... 1090, 1107	
\draw_zvec:n	126, 127, 1215, 1244
draw internal commands:	82, 93, 97
\_draw_backend_begin:	85
\_draw_backend_box_use:Nnnnn	120, 1250
\_draw_backend_cap_but:	137, 1323
\_draw_backend_cap_rectangle:	137, 1319
\_draw_backend_cap_round:	729, 1282, 1288, 1653, 1658, 1659
\_draw_backend_clip:	1275, 1282, 1288
\_draw_backend_closepath:	1215, 1229,
\_draw_backend_curveto:nnnnnn	1260, 1264, 1266, 1268, 1269, 1272
\_draw_backend_dash_pattern:nn	
\_draw_backend_discardpath:	
\_draw_backend_end:	
\_draw_backend_evenodd_rule:	
\_draw_backend_join_bevel:	
\_draw_backend_join_miter:	
\_draw_backend_join_round:	
\_draw_backend_lineto:nn	
\_draw_backend_linewidth:n	
\_draw_backend_miterlimit:n	
\_draw_backend_moveto:nn	
\_draw_backend_nonzero_rule:	
\_draw_backend_rectangle:nnnn	
\_draw_backend_scope_begin:	
\_draw_backend_scope_end:	
\_draw_box_use:Nnnnn	
\_draw_color:nn	
\_draw_color:nw	
\_draw_color_aux:nn	
\_draw_color_cmyk:nw	
\_draw_color_gray:nw	
\_draw_color_rgb:nw	
\_draw_color_spot:nw	
\l_draw_color_tmp_tl	
\l_draw_corner_arc_bool	
\l_draw_corner_xarc_dim	
\l_draw_corner_yarc_dim	
\_draw_draw_polar:nnn	
\_draw_draw_vec_polar:nnn	
\l_draw_fill_color_tl	
\g_draw_id_int	
\l_draw_layer_close_bool	
\l_draw_layer_main_box	
\l_draw_layer_tl	
\g_draw_layers_clist	
\_draw_layers_insert:	
\_draw_layers_restore:	
\_draw_layers_save:	
\g_draw_linewidth_dim	
\l_draw_linewidth_dim	
\l_draw_main_box	

\l__draw_matrix_a_fp .....	\l__draw_path_lasty_dim <a href="#">734</a> , <a href="#">747</a> , <a href="#">775</a>
. <a href="#">40</a> , <a href="#">1163</a> , <a href="#">1195</a> , <a href="#">1714</a> , <a href="#">1722</a> , <a href="#">1736</a> ,	\__draw_path_lineto:nn ..... <a href="#">238</a> , <a href="#">290</a>
<a href="#">1742</a> , <a href="#">1776</a> , <a href="#">1778</a> , <a href="#">1802</a> , <a href="#">1811</a> , <a href="#">1818</a>	\__draw_path_mark_corner: .....
\l__draw_matrix_active_bool ....	..... <a href="#">229</a> , <a href="#">258</a> , <a href="#">267</a> , <a href="#">284</a> , <a href="#">295</a> , <a href="#">313</a> , <a href="#">384</a>
<a href="#">558</a> , <a href="#">1158</a> , <a href="#">1190</a> , <a href="#">1713</a> , <a href="#">1747</a> , <a href="#">1748</a> , <a href="#">1794</a>	\__draw_path_moveto:nn .....
\l__draw_matrix_b_fp .....	..... <a href="#">238</a> , <a href="#">288</a> , <a href="#">499</a> , <a href="#">507</a>
. <a href="#">41</a> , <a href="#">1169</a> , <a href="#">1200</a> , <a href="#">1714</a> , <a href="#">1723</a> , <a href="#">1737</a> ,	\__draw_path_rectangle:nnnn ... <a href="#">552</a>
<a href="#">1743</a> , <a href="#">1777</a> , <a href="#">1779</a> , <a href="#">1803</a> , <a href="#">1813</a> , <a href="#">1814</a>	\__draw_path_rectangle_corners:nnnn
\l__draw_matrix_c_fp .....	..... <a href="#">581</a>
. <a href="#">42</a> , <a href="#">1164</a> , <a href="#">1196</a> , <a href="#">1714</a> , <a href="#">1724</a> , <a href="#">1738</a> ,	\__draw_path_rectangle_corners:nnnnn
<a href="#">1744</a> , <a href="#">1776</a> , <a href="#">1778</a> , <a href="#">1803</a> , <a href="#">1815</a> , <a href="#">1816</a>	..... <a href="#">584</a> , <a href="#">587</a>
\l__draw_matrix_d_fp .....	\__draw_path_rectangle_rounded:nnnn
. <a href="#">43</a> , <a href="#">1170</a> , <a href="#">1201</a> , <a href="#">1717</a> , <a href="#">1725</a> , <a href="#">1739</a> ,	..... <a href="#">552</a>
<a href="#">1745</a> , <a href="#">1777</a> , <a href="#">1779</a> , <a href="#">1802</a> , <a href="#">1812</a> , <a href="#">1817</a>	\__draw_path_reset_limits: .....
\__draw_path_arc:nnnn ..... <a href="#">343</a>	..... <a href="#">182</a> , <a href="#">656</a> , <a href="#">754</a> , <a href="#">1232</a>
\__draw_path_arc:nnNnn ..... <a href="#">343</a>	\l__draw_path_tmp_tl .....
\c__draw_path_arc_60_fp ..... <a href="#">343</a>	..... <a href="#">173</a> , <a href="#">421</a> , <a href="#">444</a> , <a href="#">463</a> , <a href="#">467</a> , <a href="#">471</a> , <a href="#">475</a>
\c__draw_path_arc_90_fp ..... <a href="#">343</a>	\l__draw_path_tmpa_fp .....
\__draw_path_arc_add:nnnn ..... <a href="#">343</a>	..... <a href="#">173</a> , <a href="#">309</a> , <a href="#">319</a> , <a href="#">331</a>
\__draw_path_arc_aux_add:nn ....	\l__draw_path_tmpb_fp .....
..... <a href="#">450</a> , <a href="#">456</a> , <a href="#">468</a> , <a href="#">473</a>	..... <a href="#">173</a> , <a href="#">310</a> , <a href="#">326</a> , <a href="#">335</a>
\__draw_path_arc_auxi:nnnnNnn ...	\__draw_path_update_last:nn ....
..... <a href="#">343</a> , <a href="#">370</a> , <a href="#">377</a>	..... <a href="#">211</a> , <a href="#">248</a> , <a href="#">261</a> , <a href="#">280</a> , <a href="#">570</a>
\__draw_path_arc_auxii:nnnNnnnn <a href="#">343</a>	\__draw_path_update_limits:nn ...
\__draw_path_arc_auxiii:nn .... <a href="#">343</a>	..... <a href="#">22</a> , <a href="#">25</a> , <a href="#">28</a> , <a href="#">31</a> ,
\__draw_path_arc_auxiv:nnnn ... <a href="#">343</a>	<a href="#">182</a> , <a href="#">246</a> , <a href="#">259</a> , <a href="#">276</a> , <a href="#">277</a> , <a href="#">278</a> , <a href="#">567</a> , <a href="#">568</a>
\__draw_path_arc_auxv:nn ..... <a href="#">343</a>	\__draw_path_use:n ..... <a href="#">644</a>
\__draw_path_arc_auxvi:nn ..... <a href="#">343</a>	\__draw_path_use_action_draw: .. <a href="#">644</a>
\l__draw_path_arc_delta_fp .... <a href="#">343</a>	\__draw_path_use_action_fillstroke:
\l__draw_path_arc_start_fp .... <a href="#">343</a>	..... <a href="#">644</a>
\__draw_path_curveto:nnnn ..... <a href="#">300</a>	\l__draw_path_use_bb_bool ..... <a href="#">642</a>
\__draw_path_curveto:nnnnnn ..... <a href="#">238</a> , <a href="#">296</a> , <a href="#">314</a> , <a href="#">444</a> , <a href="#">511</a> , <a href="#">521</a> , <a href="#">531</a> , <a href="#">541</a>	\l__draw_path_use_clear_bool <a href="#">642</a> , <a href="#">700</a>
\c__draw_path_curveto_a_fp .... <a href="#">300</a>	\l__draw_path_use_clip_bool ....
\c__draw_path_curveto_b_fp .... <a href="#">300</a>	..... <a href="#">639</a> , <a href="#">662</a> , <a href="#">680</a>
\__draw_path_ellipse:nnnnnn ... <a href="#">487</a>	\l__draw_path_use_fill_bool ....
\__draw_path_ellipse_arci:nnnnnn <a href="#">487</a>	..... <a href="#">639</a> , <a href="#">663</a> , <a href="#">684</a> , <a href="#">689</a> , <a href="#">695</a> , <a href="#">709</a>
\__draw_path_ellipse_arcii:nnnnnn	\__draw_path_use_stroke_bb: ... <a href="#">644</a>
..... <a href="#">487</a>	\__draw_path_use_stroke_bb-
\__draw_path_ellipse_arciiii:nnnnnn	aux:NnN ..... <a href="#">644</a>
..... <a href="#">487</a>	\l__draw_path_use_stroke_bool ...
\__draw_path_ellipse_arciiv:nnnnnn	<a href="#">639</a> , <a href="#">664</a> , <a href="#">677</a> , <a href="#">685</a> , <a href="#">690</a> , <a href="#">696</a> , <a href="#">705</a> , <a href="#">710</a>
..... <a href="#">487</a>	\g__draw_path_xmax_dim .....
\c__draw_path_ellipse_fp ..... <a href="#">487</a>	..... <a href="#">178</a> , <a href="#">184</a> , <a href="#">185</a> , <a href="#">206</a> , <a href="#">748</a> , <a href="#">770</a>
\__draw_path_grid_auxi:nnnnnn .. <a href="#">589</a>	\l__draw_path_xmax_dim . <a href="#">734</a> , <a href="#">748</a> , <a href="#">770</a>
\__draw_path_grid_auxii:nnnnnn . <a href="#">589</a>	\g__draw_path_xmin_dim .....
\__draw_path_grid_auxiii:nnnnnn <a href="#">589</a>	..... <a href="#">178</a> , <a href="#">186</a> , <a href="#">187</a> , <a href="#">207</a> , <a href="#">749</a> , <a href="#">771</a>
\__draw_path_grid_auxiiii:nnnnnn <a href="#">589</a>	\l__draw_path_xmin_dim . <a href="#">734</a> , <a href="#">749</a> , <a href="#">771</a>
\__draw_path_grid_auxiv:nnnnnnnn <a href="#">589</a>	\g__draw_path_ymax_dim .....
\g__draw_path_lastx_dim .....	..... <a href="#">178</a> , <a href="#">188</a> , <a href="#">189</a> , <a href="#">208</a> , <a href="#">750</a> , <a href="#">772</a>
..... <a href="#">176</a> , <a href="#">213</a> , <a href="#">318</a> , <a href="#">451</a> , <a href="#">457</a> , <a href="#">746</a> , <a href="#">774</a>	\l__draw_path_ymax_dim . <a href="#">734</a> , <a href="#">750</a> , <a href="#">772</a>
\l__draw_path_lastx_dim <a href="#">734</a> , <a href="#">746</a> , <a href="#">774</a>	\g__draw_path_ymin_dim .....
\g__draw_path_lasty_dim .....	..... <a href="#">178</a> , <a href="#">190</a> , <a href="#">191</a> , <a href="#">209</a> , <a href="#">751</a> , <a href="#">773</a>
..... <a href="#">176</a> , <a href="#">214</a> , <a href="#">325</a> , <a href="#">452</a> , <a href="#">458</a> , <a href="#">747</a> , <a href="#">775</a>	\l__draw_path_ymin_dim . <a href="#">734</a> , <a href="#">751</a> , <a href="#">773</a>

<code>\__draw_point_interpolate_-</code>	<code>\__draw_point_process:nn</code>
<code>arcaxes_auxi:nnnnnnnn</code>	21, 24, 27, 30, 240, 252,
<code>\__draw_point_interpolate_-</code>	288, 290, 422, 438, 787, 852, 966,
<code>arcaxes_auxii:nnnnnnnn</code>	972, 1098, 1153, 1185, 1752, 1784, 1830
<code>\__draw_point_interpolate_-</code>	<code>\__draw_point_process:nnn</code>
<code>arcaxes_auxiii:nnnnnn</code>	302,
<code>\__draw_point_interpolate_-</code>	428, 554, 583, 591, 787, 895, 951, 1828
<code>arcaxes_auxiv:nnnnnnnn</code>	<code>\__draw_point_process:nnnn</code>
<code>\__draw_point_interpolate_curve_-</code>	265, 293, 489, 787, 984
<code>auxi:nnnnnnnn</code>	<code>\__draw_point_process:nnnnn</code>
<code>\__draw_point_interpolate_curve_-</code>	787, 870, 1016
<code>auxii:nnnnnnnn</code>	<code>\__draw_point_process_auxi:nn</code>
<code>\__draw_point_interpolate_curve_-</code>	787
<code>auxiii:nnnnnn</code>	<code>\__draw_point_process_auxii:nw</code>
<code>\__draw_point_interpolate_curve_-</code>	787
<code>auxiv:nnnnnn</code>	<code>\__draw_point_process_auxiii:nnn</code>
<code>\__draw_point_interpolate_curve_-</code>	787
<code>auxv:nnw</code>	<code>\__draw_point_process_auxiv:nw</code>
<code>\__draw_point_interpolate_curve_-</code>	787
<code>auxvi:n</code>	<code>\__draw_point_process_auxvi:nw</code>
<code>\__draw_point_interpolate_curve_-</code>	787
<code>auxvii:nnnnnnnn</code>	<code>\__draw_point_to_dim:n</code>
<code>\__draw_point_interpolate_curve_-</code>	790,
<code>auxviii:nnnnnn</code>	800, 801, 811, 812, 813, 824, 825,
<code>\__draw_point_interpolate_-</code>	826, 827, 838, 1010, 1080, 1112,
<code>distance:nnnn</code>	1126, 1144, 1160, 1176, 1192, 1205
<code>\__draw_point_interpolate_-</code>	<code>\__draw_point_to_dim_aux:n</code>
<code>distance:nnnnnn</code>	838
<code>\__draw_point_interpolate_line_-</code>	<code>\__draw_point_to_dim_aux:w</code>
<code>aux:nnnnnn</code>	838
<code>\__draw_point_interpolate_line_-</code>	<code>\__draw_point_transform:nn</code>
<code>aux:nnnnnn</code>	1151
<code>\__draw_point_intersect_circles_-</code>	<code>\__draw_point_transform_noshift:n</code>
<code>auxi:nnnnnnnn</code>	425, 441, 492, 493, 1183
<code>\__draw_point_intersect_circles_-</code>	<code>\__draw_point_transform_noshift:nn</code>
<code>auxii:nnnnnnnn</code>	1183
<code>\__draw_point_intersect_circles_-</code>	<code>\__draw_point_unit_vector:nn</code>
<code>auxiii:nnnnnnnn</code>	851
<code>\__draw_point_intersect_circles_-</code>	<code>\__draw_point_unit_vector:nnn</code>
<code>auxiv:nnnnnnnn</code>	851
<code>\__draw_point_intersect_circles_-</code>	<code>\__draw_point_vec:nn</code>
<code>auxv:nnnnnnnnnn</code>	1108
<code>\__draw_point_intersect_circles_-</code>	<code>\__draw_point_vec:nnn</code>
<code>auxvi:nnnnnnnnnn</code>	1108
<code>\__draw_point_intersect_circles_-</code>	<code>\__draw_point_vec_polar:nnn</code>
<code>auxvii:nnnnnnnn</code>	1138
<code>\__draw_point_intersect_lines:nnnnnn</code>	<code>\__draw_reset_bb:</code>
<code>\__draw_point_intersect_lines:nnnnnnnn</code>	1218, 1231, 1303
<code>\__draw_point_intersect_lines_-</code>	<code>\__draw_scope_bb_begin:</code>
<code>aux:nnnnnn</code>	1296, 1315
	<code>\__draw_scope_bb_end:</code>
	1296, 1325
	<code>\__draw_select_cmyk:nw</code>
	1687
	<code>\__draw_select_gray:nw</code>
	1687
	<code>\__draw_select_rgb:nw</code>
	1687
	<code>\__draw_softpath_add:n</code>
	767, 1333, 1352,
	1361, 1370, 1375, 1385, 1393, 1648
	<code>\c__draw_softpath_arc_fp</code>
	1427, 1590, 1594, 1599, 1603
	<code>\__draw_softpath_clear:</code>
	655, 701, 759, 763, 1236, 1336
	<code>\__draw_softpath_close_op:nn</code>
	1354, 1398, 1489, 1525, 1627
	<code>\__draw_softpath_closepath:</code>
	285, 506, 1350
	<code>\l__draw_softpath_corneri_dim</code>
	1421, 1474, 1477, 1563

\l__draw_softpath_cornerii_dim ..	\__draw_softpath_rectangle:nnnn ..
..... <a href="#">1421</a> , <a href="#">1475</a> , <a href="#">1478</a> , <a href="#">1554</a>	..... <a href="#">569</a> , <a href="#">1350</a>
\g__draw_softpath_corners_bool ..	\__draw_softpath_rectangle_-
..... <a href="#">758</a> , <a href="#">765</a> , <a href="#">1332</a> , <a href="#">1344</a> , <a href="#">1395</a> , <a href="#">1430</a> , <a href="#">1446</a>	opi:nn ..... <a href="#">1387</a> , <a href="#">1398</a>
\l__draw_softpath_corners_bool ..	\__draw_softpath_rectangle_-
..... <a href="#">734</a> , <a href="#">757</a> , <a href="#">766</a>	opi:nnNnn ..... <a href="#">1398</a>
\l__draw_softpath_curve_end_tl ..	\__draw_softpath_rectangle_-
..... <a href="#">1420</a> , <a href="#">1551</a> , <a href="#">1570</a> , <a href="#">1619</a> , <a href="#">1630</a>	opii:nn ..... <a href="#">1388</a> , <a href="#">1398</a>
\__draw_softpath_curveto:nnnnnn ..	\__draw_softpath_round_action:nn
..... <a href="#">279</a> , <a href="#">1350</a>	..... <a href="#">1428</a>
\__draw_softpath_curveto_opi:nn ..	\__draw_softpath_round_action:Nnn
..... <a href="#">1363</a> , <a href="#">1398</a> , <a href="#">1486</a> , <a href="#">1524</a> , <a href="#">1587</a>	..... <a href="#">1428</a>
\__draw_softpath_curveto_-	\__draw_softpath_round_action_-
opi:nnNnnNnn ..... <a href="#">1398</a>	close: ..... <a href="#">1428</a>
\__draw_softpath_curveto_opii:nn	\__draw_softpath_round_action_-
..... <a href="#">1364</a> , <a href="#">1398</a> , <a href="#">1596</a>	curveto:NnnNnn ..... <a href="#">1428</a>
\__draw_softpath_curveto_-	\__draw_softpath_round_calc:NnnNnn
opiii:nn ..... <a href="#">1365</a> , <a href="#">1398</a> , <a href="#">1605</a>	..... <a href="#">1428</a>
\l__draw_softpath_first_tl .....	\__draw_softpath_round_calc:nnnnnn
..... <a href="#">1421</a> , <a href="#">1437</a> , <a href="#">1454</a> ,	..... <a href="#">1428</a>
<a href="#">1455</a> , <a href="#">1465</a> , <a href="#">1484</a> , <a href="#">1485</a> , <a href="#">1511</a> , <a href="#">1515</a>	\__draw_softpath_round_calc:nnnnnw
\l__draw_softpath_internal_tl .....	..... <a href="#">1428</a>
..... <a href="#">1331</a> , <a href="#">1338</a> , <a href="#">1339</a> , <a href="#">1439</a> , <a href="#">1441</a>	\__draw_softpath_round_close:nn <a href="#">1428</a>
\g__draw_softpath_lastx_dim .....	\__draw_softpath_round_close:w <a href="#">1428</a>
..... <a href="#">752</a> , <a href="#">768</a> , <a href="#">1346</a> , <a href="#">1355</a> , <a href="#">1379</a>	\__draw_softpath_round_corners: ..
\l__draw_softpath_lastx_dim .....	..... <a href="#">674</a> , <a href="#">1428</a>
..... <a href="#">740</a> , <a href="#">752</a> , <a href="#">768</a>	\__draw_softpath_round_end: .. <a href="#">1428</a>
\l__draw_softpath_lastx_fp .....	\__draw_softpath_round_lookahead:NnnNnn
.. <a href="#">1421</a> , <a href="#">1435</a> , <a href="#">1456</a> , <a href="#">1504</a> , <a href="#">1566</a> , <a href="#">1573</a>	..... <a href="#">1428</a>
\g__draw_softpath_lasty_dim .....	\__draw_softpath_round_loop:Nnn <a href="#">1428</a>
..... <a href="#">753</a> , <a href="#">769</a> , <a href="#">1346</a> , <a href="#">1356</a> , <a href="#">1380</a>	\__draw_softpath_round_roundpoint:NnnNnnNnn
\l__draw_softpath_lasty_dim .....	..... <a href="#">1428</a>
..... <a href="#">741</a> , <a href="#">753</a> , <a href="#">769</a>	\__draw_softpath_roundpoint:nn ..
\l__draw_softpath_lasty_fp .....	..... <a href="#">233</a> , <a href="#">1350</a>
.. <a href="#">1421</a> , <a href="#">1436</a> , <a href="#">1457</a> , <a href="#">1505</a> , <a href="#">1567</a> , <a href="#">1574</a>	\__draw_softpath_roundpoint_-
\__draw_softpath_lineto:nn <a href="#">260</a> , <a href="#">1350</a>	op:nn ..... <a href="#">1394</a> , <a href="#">1398</a> , <a href="#">1451</a> , <a href="#">1533</a>
\__draw_softpath_lineto_op:nn ..	\__draw_softpath_use: ..... <a href="#">679</a> , <a href="#">1336</a>
..... <a href="#">1371</a> , <a href="#">1398</a> , <a href="#">1492</a> , <a href="#">1523</a> , <a href="#">1636</a>	\__draw_split_select:nw ..... <a href="#">1687</a>
\g__draw_softpath_main_tl .....	\l__draw_stroke_color_tl ..... <a href="#">1275</a>
..... <a href="#">755</a> , <a href="#">1330</a> , <a href="#">1334</a> , <a href="#">1338</a> , <a href="#">1343</a> , <a href="#">1439</a> , <a href="#">1647</a>	\l__draw_tmp_box ..... <a href="#">11</a> , <a href="#">35</a> , <a href="#">46</a> ,
\l__draw_softpath_main_tl .....	<a href="#">50</a> , <a href="#">52</a> , <a href="#">53</a> , <a href="#">54</a> , <a href="#">55</a> , <a href="#">61</a> , <a href="#">63</a> , <a href="#">64</a> , <a href="#">65</a> , <a href="#">66</a> , <a href="#">67</a>
..... <a href="#">19</a> , <a href="#">755</a> , <a href="#">767</a> , <a href="#">1418</a> ,	\l__draw_tmp_seq ..... <a href="#">1661</a>
<a href="#">1433</a> , <a href="#">1460</a> , <a href="#">1462</a> , <a href="#">1643</a> , <a href="#">1645</a> , <a href="#">1648</a>	\__draw_tranform_triangle:nnnnnn
\g__draw_softpath_move_bool .....	..... <a href="#">1831</a> , <a href="#">1836</a>
..... <a href="#">1348</a> , <a href="#">1377</a>	\__draw_transform:nnnn ..... <a href="#">1760</a>
\l__draw_softpath_move_tl .....	\__draw_transform_invert:n ... <a href="#">1792</a>
..... <a href="#">1421</a> , <a href="#">1438</a> ,	\__draw_transform_rotate:n ... <a href="#">1862</a>
<a href="#">1461</a> , <a href="#">1464</a> , <a href="#">1512</a> , <a href="#">1614</a> , <a href="#">1637</a> , <a href="#">1644</a>	\__draw_transform_rotate:nn ... <a href="#">1862</a>
\__draw_softpath_moveto:nn <a href="#">247</a> , <a href="#">1350</a>	\__draw_transform_shift:nn ... <a href="#">1760</a>
\__draw_softpath_moveto_op:nn ..	\__draw_transform_shift_absolute:nn
..... <a href="#">1376</a> , <a href="#">1398</a> , <a href="#">1458</a> , <a href="#">1616</a>	..... <a href="#">1734</a>
\l__draw_softpath_part_tl .....	\__draw_vec:nn ..... <a href="#">1090</a>
..... <a href="#">1419</a> , <a href="#">1434</a> ,	\__draw_vec:nnn ..... <a href="#">1090</a>
<a href="#">1463</a> , <a href="#">1466</a> , <a href="#">1468</a> , <a href="#">1502</a> , <a href="#">1557</a> , <a href="#">1646</a>	



`\g__draw_xmax_dim` . . . . . 194,  
 195, 1210, 1220, 1255, 1270, 1299, 1307  
`\l__draw_xmax_dim` . . . 1292, 1299, 1307  
`\g__draw_xmin_dim` . . . . .  
 . . . . . 196, 197, 1210, 1221,  
 1253, 1256, 1262, 1270, 1300, 1308  
`\l__draw_xmin_dim` . . . 1292, 1300, 1308  
`\l__draw_xshift_dim` . . . . 48, 1165,  
 1179, 1714, 1729, 1757, 1789, 1823  
`\l__draw_xvec_x_dim` . . . . .  
 . . . . . 1084, 1114, 1128, 1146  
`\l__draw_xvec_y_dim` . 1084, 1115, 1132  
`\g__draw_ymax_dim` . . . . . 198,  
 199, 1210, 1222, 1257, 1267, 1301, 1309  
`\l__draw_ymax_dim` . . . 1292, 1301, 1309  
`\g__draw_ymin_dim` . 200, 201, 1210,  
 1223, 1258, 1263, 1267, 1302, 1310  
`\l__draw_ymin_dim` . . . 1292, 1302, 1310  
`\l__draw_yshift_dim` . . . . 49, 1171,  
 1179, 1714, 1730, 1758, 1790, 1824  
`\l__draw_yvec_x_dim` . 1084, 1114, 1129  
`\l__draw_yvec_y_dim` . . . . .  
 . . . . . 1084, 1115, 1133, 1147  
`\l__draw_zvec_x_dim` . . . . 1084, 1130  
`\l__draw_zvec_y_dim` . . . . 1084, 1134

**E**

`\end` . . . . . 167, 781  
 exp commands:  
`\exp_after:wN` . . . . .  
 444, 462, 1440, 1514, 1617, 1628, 1637  
`\exp_args:Nf` . . . . . 789, 855  
`\exp_args:Nff` . . . . . 799  
`\exp_args:Nfff` . . . . . 810  
`\exp_args:Nffff` . . . . . 823  
`\exp_args:NNNV` . . . . . 1248  
`\exp_not:N` . . . . . 1559, 1587,  
 1596, 1605, 1614, 1617, 1618, 1619,  
 1620, 1624, 1628, 1629, 1630, 1631

**F**

fp commands:  
`\fp_compare:nNnTF` . . . . 358, 368, 861  
`\fp_compare_p:nNn` . . . . .  
 . . . . . 1742, 1743, 1744, 1745  
`\fp_const:Nn` . . . . .  
 . . . . . 341, 342, 480, 481, 549, 1427  
`\fp_eval:n` . 350, 351, 372, 379, 388,  
 839, 847, 856, 877, 878, 879, 880,  
 881, 882, 902, 907, 908, 915, 922,  
 923, 930, 937, 939, 952, 957, 975,  
 991, 996, 1003, 1004, 1023, 1030,  
 1052, 1053, 1072, 1073, 1074, 1075,

1109, 1122, 1141, 1677, 1765, 1766,  
 1767, 1768, 1798, 1863, 1867, 1868  
`\fp_new:N` . . . . . 174, 175, 478,  
 479, 1421, 1422, 1714, 1715, 1716, 1717  
`\fp_set:Nn` . . . . . 309, 310, 364, 365,  
 445, 446, 1456, 1457, 1504, 1505,  
 1573, 1574, 1722, 1725, 1736, 1737,  
 1738, 1739, 1811, 1813, 1815, 1817  
`\fp_to_decimal:N` . . . . . 371, 378, 386  
`\fp_to_dim:n` . . . . . 316, 323,  
 330, 334, 352, 353, 399, 408, 476,  
 500, 512, 513, 514, 515, 516, 517,  
 522, 523, 524, 525, 526, 527, 532,  
 533, 534, 535, 536, 537, 542, 543,  
 544, 545, 546, 547, 615, 616, 1589,  
 1593, 1598, 1602, 1658, 1666, 1671  
`\fp_use:N` . . . . . 40, 41, 42, 43, 549  
`\fp_while_do:nNnn` . . . . . 366  
`\fp_zero:N` . . . . 1435, 1436, 1723, 1724  
`\c_one_fp` . . . . . 1742, 1745  
`\c_zero_fp` . . . . . 861, 1743, 1744

**G**

group commands:  
`\group_begin:` . . . . . 34, 60, 90,  
 101, 745, 1227, 1281, 1298, 1432, 1663  
`\group_end:` . . . . . 56, 68, 115,  
 118, 776, 1273, 1289, 1311, 1444, 1673

**H**

hbox commands:  
`\hbox_gset:Nw` . . . . . 99  
`\hbox_gset_end:` . . . . . 116  
`\hbox_set:Nn` . . . . . 35, 46, 61, 1260  
`\hbox_set:Nw` . . . . . 1229, 1244  
`\hbox_set_end:` . . . . . 1248, 1252

**I**

int commands:  
`\int_gincr:N` . . . . . 1228  
`\int_if_odd:nTF` . . . . . 938  
`\int_new:N` . . . . . 1217

**M**

mode commands:  
`\mode_leave_vertical:` . . . . . 1271  
 msg commands:  
`\msg_error:nnn` . . . . . 76, 107, 108, 671  
`\msg_new:nnn` . . . . . 162  
`\msg_new:nnnn` . . . . . 159, 164, 778

**P**

`\pgfextractx` . . . . . 20  
`\pgfextracty` . . . . . 20  
`\pgfgetlastxy` . . . . . 20

