

File I

Implementation

1 l3draw implementation

```

1 <*initex | package>
2 <@@=draw>
3 <*package>
4 \ProvidesExplPackage{l3draw}{2020-01-12}{}
5 {L3 Experimental core drawing support}
6 </package>
7 \RequirePackage { l3color }
8
9 Everything else is in the sub-files!
10 </initex | package>

```

2 l3draw-boxes implementation

```

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

```

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

`\l__draw_tmp_box`

```

11 \box_new:N \l__draw_tmp_box

```

(End definition for \l__draw_tmp_box.)

`\draw_box_use:N`
`__draw_box_use:Nnnnn`

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

```

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

```

```

30     \__draw_point_process:nn
31     { \__draw_path_update_limits:nn }
32     { \draw_point_transform:n { #2 , #5 } }
33 }
34 \group_begin:
35 \hbox_set:Nn \l__draw_tmp_box
36 {
37     \use:x
38     {
39         \__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_tmppb_fp 174 \fp_new:N \l__draw_path_tmpa_fp
175 \fp_new:N \l__draw_path_tmppb_fp

```

(End definition for `\l__draw_path_tmp_tl`, `\l__draw_path_tmpa_fp`, and `\l__draw_path_tmppb_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
\__draw_path_curveto:nnnn
\c__draw_path_curveto_a_fp
\c__draw_path_curveto_b_fp

```

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

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

and

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

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

```

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

```

```

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

```

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

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

`\draw_path_arc:nnnn`

`__draw_path_arc:nnnn`

`__draw_path_arc:nnNnn`

`__draw_path_arc_auxi:nnnnNnn`

`__draw_path_arc_auxi:fnnnNnn`

`__draw_path_arc_auxi:fnfnNnn`

`__draw_path_arc_auxii:nnnNnnnn`

`__draw_path_arc_auxiii:nn`

`__draw_path_arc_auxiv:nnnn`

`__draw_path_arc_auxv:nn`

`__draw_path_arc_auxvi:nn`

`__draw_path_arc_add:nnnn`

`\l__draw_path_arc_delta_fp`

`\l__draw_path_arc_start_fp`

`\c__draw_path_arc_90_fp`

`\c__draw_path_arc_60_fp`

```

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

```

```

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

```

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

```

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

```

```

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

```

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

```

419 \cs_new_protected:Npn \__draw_path_arc_auxii:nnnnnnn #1#2#3#4#5#6#7#8
420 {
421   \tl_clear:N \l__draw_path_tmp_tl
422   \__draw_point_process:nn
423   { \__draw_path_arc_auxiii:nn }
424   {
425     \__draw_point_transform_noshift:n
426     { \draw_point_polar:nnn {#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_set_false:N \l_draw_bb_update_bool
684       \bool_lazy_or:nnF
685         { \l__draw_path_use_fill_bool }
686         { \l__draw_path_use_stroke_bool }
687         { \__draw_backend_discardpath: }
688     }
689     \bool_lazy_or:nnT
690       { \l__draw_path_use_fill_bool }
691       { \l__draw_path_use_stroke_bool }
692     {
693       \use:c
694       {
695         __draw_backend_
696         \bool_if:NT \l__draw_path_use_fill_bool { fill }
697         \bool_if:NT \l__draw_path_use_stroke_bool { stroke }
698         :
699       }
700     }
701     \bool_if:NT \l__draw_path_use_clear_bool
702     { \__draw_softpath_clear: }
703   }
704   \cs_new_protected:Npn \__draw_path_use_action_draw:
705   {
706     \bool_set_true:N \l__draw_path_use_stroke_bool
707   }
708   \cs_new_protected:Npn \__draw_path_use_action_fillstroke:
709   {
710     \bool_set_true:N \l__draw_path_use_fill_bool
711     \bool_set_true:N \l__draw_path_use_stroke_bool
712   }

```

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.

```

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

```

```

724     \dim_gset:cn { g__draw_ #1#2 _dim }
725     {
726         \use:c { dim_ #2 :nn }
727         { \dim_use:c { g__draw_ #1#2 _dim } }
728         {
729             \dim_use:c { g__draw_path_ #1#2 _dim }
730             #3 0.5 \g__draw_linewidth_dim
731         }
732     }
733 }
734 }

```

(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
\l__draw_path_lasty_dim
\l__draw_path_xmax_dim
\l__draw_path_xmin_dim
\l__draw_path_ymax_dim
\l__draw_path_ymin_dim
\l__draw_softpath_corners_bool
735 \dim_new:N \l__draw_path_lastx_dim
736 \dim_new:N \l__draw_path_lasty_dim
737 \dim_new:N \l__draw_path_xmax_dim
738 \dim_new:N \l__draw_path_xmin_dim
739 \dim_new:N \l__draw_path_ymax_dim
740 \dim_new:N \l__draw_path_ymin_dim
741 \dim_new:N \l__draw_softpath_lastx_dim
742 \dim_new:N \l__draw_softpath_lasty_dim
743 \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.

```

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

```

```

765 \bool_gset_eq:NN
766 \g__draw_softpath_corners_bool
767 \l__draw_softpath_corners_bool
768 \__draw_softpath_add:o \l__draw_softpath_main_tl
769 \dim_gset_eq:NN \g__draw_softpath_lastx_dim \l__draw_softpath_lastx_dim
770 \dim_gset_eq:NN \g__draw_softpath_lasty_dim \l__draw_softpath_lasty_dim
771 \dim_gset_eq:NN \g__draw_path_xmax_dim \l__draw_path_xmax_dim
772 \dim_gset_eq:NN \g__draw_path_xmin_dim \l__draw_path_xmin_dim
773 \dim_gset_eq:NN \g__draw_path_ymax_dim \l__draw_path_ymax_dim
774 \dim_gset_eq:NN \g__draw_path_ymin_dim \l__draw_path_ymin_dim
775 \dim_gset_eq:NN \g__draw_path_lastx_dim \l__draw_path_lastx_dim
776 \dim_gset_eq:NN \g__draw_path_lasty_dim \l__draw_path_lasty_dim
777 \group_end:
778 }

```

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

```

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

```

5 l3draw-points implementation

```

786 <*initex | package>
787 <@@=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`, `\pgfpointsspherical`: The usefulness of these commands is not currently clear.
- `\pgfpointborderrectangle`, `\pgfpointborderellipse`: To be revisited once the semantics and use cases are clear.

- `\pgfqpoint`, `\pgfqpointscale`, `\pgfqpointpolar`, `\pgfqpointxy`, `\pgfqpointxyz`: The expandable approach taken in the code here, along with the absolute requirement for ε -TeX, means it is likely many use cases for these commands may be covered in other ways. This may be revisited as higher-level structures are constructed.

5.1 Support functions

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

```

\__draw_point_process:nn
  \_draw_point_process_auxi:nn
  \_draw_point_process_auxii:nw
\__draw_point_process:nnn
  \_draw_point_process_auxiii:nnn
  \_draw_point_process_auxiv:nw
\__draw_point_process:nnnn
  \_draw_point_process_auxv:nnnn
  \_draw_point_process_auxvi:nw
\__draw_point_process:nnnnn
  \_draw_point_process_auxvii:nnnnn
  \_draw_point_process_auxviii:nw

```

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.

```

788 \cs_new:Npn \__draw_point_process:nn #1#2
789 {
790   \exp_args:Nf \__draw_point_process_auxi:nn
791     { \__draw_point_to_dim:n {#2} }
792     {#1}
793 }
794 \cs_new:Npn \__draw_point_process_auxi:nn #1#2
795 { \__draw_point_process_auxii:nw {#2} #1 \q_stop }
796 \cs_new:Npn \__draw_point_process_auxii:nw #1 #2 , #3 \q_stop
797 { #1 {#2} {#3} }
798 \cs_new:Npn \__draw_point_process:nnn #1#2#3
799 {
800   \exp_args:Nff \__draw_point_process_auxiii:nnn
801     { \__draw_point_to_dim:n {#2} }
802     { \__draw_point_to_dim:n {#3} }
803     {#1}
804 }
805 \cs_new:Npn \__draw_point_process_auxiii:nnn #1#2#3
806 { \__draw_point_process_auxiv:nw {#3} #1 \q_mark #2 \q_stop }
807 \cs_new:Npn \__draw_point_process_auxiv:nw #1 #2 , #3 \q_mark #4 , #5 \q_stop
808 { #1 {#2} {#3} {#4} {#5} }
809 \cs_new:Npn \__draw_point_process:nnnn #1#2#3#4
810 {
811   \exp_args:Nfff \__draw_point_process_auxv:nnnn
812     { \__draw_point_to_dim:n {#2} }
813     { \__draw_point_to_dim:n {#3} }
814     { \__draw_point_to_dim:n {#4} }
815     {#1}
816 }
817 \cs_new:Npn \__draw_point_process_auxv:nnnn #1#2#3#4
818 { \__draw_point_process_auxvi:nw {#4} #1 \q_mark #2 \q_mark #3 \q_stop }
819 \cs_new:Npn \__draw_point_process_auxvi:nw
820   #1 #2 , #3 \q_mark #4 , #5 \q_mark #6 , #7 \q_stop
821   { #1 {#2} {#3} {#4} {#5} {#6} {#7} }
822 \cs_new:Npn \__draw_point_process:nnnnn #1#2#3#4#5
823 {
824   \exp_args:Nffff \__draw_point_process_auxvii:nnnnn
825     { \__draw_point_to_dim:n {#2} }
826     { \__draw_point_to_dim:n {#3} }
827     { \__draw_point_to_dim:n {#4} }
828     { \__draw_point_to_dim:n {#5} }
829     {#1}

```

```

830 }
831 \cs_new:Npn \__draw_point_process_auxvii:nnnnn #1#2#3#4#5
832 {
833   \__draw_point_process_auxviii:nw
834   {#5} #1 \q_mark #2 \q_mark #3 \q_mark #4 \q_stop
835 }
836 \cs_new:Npn \__draw_point_process_auxviii:nw
837 #1 #2 , #3 \q_mark #4 , #5 \q_mark #6 , #7 \q_mark #8 , #9 \q_stop
838 { #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 839 \cs_new:Npn \__draw_point_to_dim:n #1
\__draw_point_to_dim_aux:f 840 { \__draw_point_to_dim_aux:f { \fp_eval:n {#1} } }
\__draw_point_to_dim_aux:w 841 \cs_new:Npn \__draw_point_to_dim_aux:n #1
842 { \__draw_point_to_dim_aux:w #1 }
843 \cs_generate_variant:Nn \__draw_point_to_dim_aux:n { f }
844 \cs_new:Npn \__draw_point_to_dim_aux:w ( #1 , ~ #2 ) { #1pt , #2pt }

```

5.2 Polar co-ordinates

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

```

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

```

5.3 Point expression arithmetic

These functions all take point expressions as arguments.

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

```

\draw_point_unit_vector:n
\__draw_point_unit_vector:nn
\__draw_point_unit_vector:nnn

```

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

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

```

852 \cs_new:Npn \draw_point_unit_vector:n #1
853 { \__draw_point_process:nn { \__draw_point_unit_vector:nn } {#1} }
854 \cs_new:Npn \__draw_point_unit_vector:nn #1#2
855 {
856   \exp_args:Nf \__draw_point_unit_vector:nnn
857   { \fp_eval:n { (sqrt(#1 * #1 + #2 * #2)) } }
858   {#1} {#2}
859 }
860 \cs_new:Npn \__draw_point_unit_vector:nnn #1#2#3
861 {

```

```

862 \fp_compare:nNnTF {#1} = \c_zero_fp
863 { Opt, 1pt }
864 {
865   \__draw_point_to_dim:n
866   { ( #2 , #3 ) / #1 }
867 }
868 }

```

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.)

```

869 \cs_new:Npn \draw_point_intersect_lines:nnnn #1#2#3#4
870 {
871   \__draw_point_process:nnnnn
872   { \__draw_point_intersect_lines:nnnnnnnn }
873   {#1} {#2} {#3} {#4}
874 }

```

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.

```

875 \cs_new:Npn \__draw_point_intersect_lines:nnnnnnnn #1#2#3#4#5#6#7#8
876 {
877   \__draw_point_intersect_lines_aux:ffffff
878   { \fp_eval:n { #1 * #4 - #2 * #3 } }
879   { \fp_eval:n { #5 * #8 - #6 * #7 } }
880   { \fp_eval:n { #1 - #3 } }
881   { \fp_eval:n { #5 - #7 } }

```

```

882     { \fp_eval:n { #2 - #4 } }
883     { \fp_eval:n { #6 - #8 } }
884   }
885   \cs_new:Npn \__draw_point_intersect_lines_aux:nnnnnn #1#2#3#4#5#6
886   {
887     \__draw_point_to_dim:n
888     {
889       ( #2 * #3 - #1 * #4 , #2 * #5 - #1 * #6 )
890       / ( #4 * #5 - #6 * #3 )
891     }
892   }
893   \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

```

\draw_point_intersect_circles:nnnnn
\__draw_point_intersect_circles_auxi:nnnnnnn
\__draw_point_intersect_circles_auxii:nnnnnnn
\__draw_point_intersect_circles_auxiii:ffnnnnn
\__draw_point_intersect_circles_auxiiii:ffnnnnn
\__draw_point_intersect_circles_auxiv:nnnnnnnn
\__draw_point_intersect_circles_auxiv:fnnnnnnn
\__draw_point_intersect_circles_auxv:nnnnnnnnn
\__draw_point_intersect_circles_auxv:ffnnnnnnn
\__draw_point_intersect_circles_auxvi:nnnnnnnnn
\__draw_point_intersect_circles_auxvi:fnnnnnnn
\__draw_point_intersect_circles_auxvii:nnnnnnn
\__draw_point_intersect_circles_auxvii:ffnnnnn

```

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

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.

```

894   \cs_new:Npn \draw_point_intersect_circles:nnnnn #1#2#3#4#5
895   {
896     \__draw_point_process:nnn
897     { \__draw_point_intersect_circles_auxi:nnnnnnn {#2} {#4} {#5} }
898     {#1} {#3}
899   }
900   \cs_new:Npn \__draw_point_intersect_circles_auxi:nnnnnnn #1#2#3#4#5#6#7
901   {
902     \__draw_point_intersect_circles_auxii:ffnnnnn
903     { \fp_eval:n {#1} } { \fp_eval:n {#2} } {#4} {#5} {#6} {#7} {#3}
904   }

```

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.

```

905 \cs_new:Npn \__draw_point_intersect_circles_auxii:nnnnnnn #1#2#3#4#5#6#7
906 {
907   \__draw_point_intersect_circles_auxiii:ffnnnnnn
908   { \fp_eval:n { #5 - #3 } }
909   { \fp_eval:n { #6 - #4 } }
910   {#1} {#2} {#3} {#4} {#7}
911 }
912 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxii:nnnnnnn { ff }
913 \cs_new:Npn \__draw_point_intersect_circles_auxiii:nnnnnnn #1#2#3#4#5#6#7
914 {
915   \__draw_point_intersect_circles_auxiv:fnnnnnnn
916   { \fp_eval:n { sqrt( #1 * #1 + #2 * #2 ) } }
917   {#1} {#2} {#3} {#4} {#5} {#6} {#7}
918 }
919 \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.

```

920 \cs_new:Npn \__draw_point_intersect_circles_auxiv:nnnnnnnn #1#2#3#4#5#6#7#8
921 {
922   \__draw_point_intersect_circles_auxv:ffnnnnnnnn
923   { \fp_eval:n { 1 / #1 } }
924   { \fp_eval:n { #4 * #4 } }
925   {#1} {#2} {#3} {#5} {#6} {#7} {#8}
926 }
927 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxiv:nnnnnnnn { f }
928 \cs_new:Npn \__draw_point_intersect_circles_auxv:nnnnnnnnn #1#2#3#4#5#6#7#8#9
929 {
930   \__draw_point_intersect_circles_auxvi:fnnnnnnnn
931   { \fp_eval:n { 0.5 * #1 * ( #2 + #3 * #3 - #6 * #6 ) } }
932   {#1} {#2} {#4} {#5} {#7} {#8} {#9}
933 }
934 \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.

```
935 \cs_new:Npn \__draw_point_intersect_circles_auxvi:nnnnnnnn #1#2#3#4#5#6#7#8
936 {
937   \__draw_point_intersect_circles_auxvii:fffnnnn
938   { \fp_eval:n { #1 * #2 } }
939   { \int_if_odd:nTF {#8} { 1 } { -1 } }
940   { \fp_eval:n { sqrt ( #3 - #1 * #1 ) * #2 } }
941   {#4} {#5} {#6} {#7}
942 }
943 \cs_generate_variant:Nn \__draw_point_intersect_circles_auxvi:nnnnnnnn { f }
944 \cs_new:Npn \__draw_point_intersect_circles_auxvii:nnnnnnnn #1#2#3#4#5#6#7
945 {
946   \__draw_point_to_dim:n
947   { #6 + #4 * #1 + #2 * #3 * #5 , #7 + #5 * #1 + -1 * #2 * #3 * #4 }
948 }
949 \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:nnm
\__draw_point_interpolate_line_aux:nnnnn
\__draw_point_interpolate_line_aux:fnnnn
\__draw_point_interpolate_line_aux:nnnnnn
\__draw_point_interpolate_line_aux:fnnnnn
950 \cs_new:Npn \draw_point_interpolate_line:nnn #1#2#3
951 {
952   \__draw_point_process:nnn
953   { \__draw_point_interpolate_line_aux:fnnnn { \fp_eval:n {#1} } }
954   {#2} {#3}
955 }
956 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnn #1#2#3#4#5
957 {
958   \__draw_point_interpolate_line_aux:fnnnnn { \fp_eval:n { 1 - #1 } }
959   {#1} {#2} {#3} {#4} {#5}
960 }
961 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnn { f }
962 \cs_new:Npn \__draw_point_interpolate_line_aux:nnnnnn #1#2#3#4#5#6
963 { \__draw_point_to_dim:n { #2 * #3 + #1 * #5 , #2 * #4 + #1 * #6 } }
964 \cs_generate_variant:Nn \__draw_point_interpolate_line_aux:nnnnnn { f }
```

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

```
\draw_point_interpolate_distance:nnm
\__draw_point_interpolate_distance:nnnnn
\__draw_point_interpolate_distance:nnnnnn
\__draw_point_interpolate_distance:fnnnnn
965 \cs_new:Npn \draw_point_interpolate_distance:nnn #1#2#3
966 {
967   \__draw_point_process:nn
968   { \__draw_point_interpolate_distance:nnnn {#1} {#3} }
969   {#2}
```

```

970 }
971 \cs_new:Npn \__draw_point_interpolate_distance:nnnn #1#2#3#4
972 {
973   \__draw_point_process:nn
974   {
975     \__draw_point_interpolate_distance:fnnnn
976     { \fp_eval:n {#1} } {#3} {#4}
977   }
978   { \draw_point_unit_vector:n { ( #2 ) - ( #3 , #4 ) } }
979 }
980 \cs_new:Npn \__draw_point_interpolate_distance:nnnnn #1#2#3#4#5
981 { \__draw_point_to_dim:n { #2 + #1 * #4 , #3 + #1 * #5 } }
982 \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:fnnnnnnnn
\draw_point_interpolate_arcaxes_auxiiii:nnnnnnnn
\draw_point_interpolate_arcaxes_auxiv:nnnnnnnnnn
\draw_point_interpolate_arcaxes_auxiv:fnnnnnnnn

```

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

```

983 \cs_new:Npn \draw_point_interpolate_arcaxes:nnnnnn #1#2#3#4#5#6
984 {
985   \__draw_point_process:nnnn
986   { \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn {#1} {#5} {#6} }
987   {#2} {#3} {#4}
988 }
989 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxi:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
990 {
991   \__draw_point_interpolate_arcaxes_auxiii:fnnnnnnnnnn
992   { \fp_eval:n {#1} } {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
993 }

```

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.

```

994 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
995 {
996   \__draw_point_interpolate_arcaxes_auxiiii:fnnnnnnnnnn
997   { \fp_eval:n { #1 * (#3) + ( 1 - #1 ) * (#2) } }

```

```

998     {#4} {#5} {#6} {#7} {#8} {#9}
999   }
1000 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxii:nnnnnnnnnn { f }
1001 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnnn #1#2#3#4#5#6#7
1002 {
1003   \__draw_point_interpolate_arcaxes_auxiv:ffnnnnnnnn
1004   { \fp_eval:n { cosd (#1) } } }
1005   { \fp_eval:n { sind (#1) } } }
1006   {#2} {#3} {#4} {#5} {#6} {#7}
1007 }
1008 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiii:nnnnnnnn { f }
1009 \cs_new:Npn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnnnn #1#2#3#4#5#6#7#8
1010 {
1011   \__draw_point_to_dim:n
1012   { #3 + #1 * #5 + #2 * #7 , #4 + #1 * #6 + #2 * #8 }
1013 }
1014 \cs_generate_variant:Nn \__draw_point_interpolate_arcaxes_auxiv:nnnnnnnnnn { ff }

```

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

```

\draw_point_interpolate_curve:nnnnnn
\draw_point_interpolate_curve_auxi:nnnnnnnnnn
\draw_point_interpolate_curve_auxii:nnnnnnnnnn
\draw_point_interpolate_curve_auxiii:nnnnnnnnnn
\draw_point_interpolate_curve_auxiiii:nnnnnnnnnn
\draw_point_interpolate_curve_auxiv:nnnnnnnnnn
\draw_point_interpolate_curve_auxv:nnw
\draw_point_interpolate_curve_auxv:ffw
\draw_point_interpolate_curve_auxvi:n

```

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.

```

1015 \cs_new:Npn \draw_point_interpolate_curve:nnnnnn #1#2#3#4#5
1016 {
1017   \__draw_point_process:nnnnnn
1018   { \__draw_point_interpolate_curve_auxi:nnnnnnnnnn {#1} }
1019   {#2} {#3} {#4} {#5}
1020 }
1021 \cs_new:Npn \__draw_point_interpolate_curve_auxi:nnnnnnnnnn #1#2#3#4#5#6#7#8#9
1022 {
1023   \__draw_point_interpolate_curve_auxii:ffnnnnnnnnnn
1024   { \fp_eval:n {#1} }
1025   {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9}
1026 }

```

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

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

In the second stage, we can drop the final point

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

and for the final stage only need one set of calculations

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

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

```

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

```

```

1067 % \begin{macrocode}
1068 \cs_new:Npn \__draw_point_interpolate_curve_auxvi:n #1
1069 { \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1 }
1070 \cs_new:Npn \__draw_point_interpolate_curve_auxvii:nnnnnnnn #1#2#3#4#5#6#7#8
1071 {
1072   \__draw_point_interpolate_curve_auxviii:ffffnn
1073   { \fp_eval:n { #1 * #5 + #2 * #3 } }
1074   { \fp_eval:n { #1 * #6 + #2 * #4 } }
1075   { \fp_eval:n { #1 * #7 + #2 * #5 } }
1076   { \fp_eval:n { #1 * #8 + #2 * #6 } }
1077   {#1} {#2}
1078 }
1079 \cs_new:Npn \__draw_point_interpolate_curve_auxviii:nnnnnn #1#2#3#4#5#6
1080 {
1081   \__draw_point_to_dim:n
1082   { #5 * #3 + #6 * #1 , #5 * #4 + #6 * #2 }
1083 }
1084 \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
1091 \cs_new_protected:Npn \draw_xvec:n #1
1092 { \__draw_vec:nn { x } {#1} }
1093 \cs_new_protected:Npn \draw_yvec:n #1
1094 { \__draw_vec:nn { y } {#1} }
1095 \cs_new_protected:Npn \draw_zvec:n #1
1096 { \__draw_vec:nn { z } {#1} }
1097 \cs_new_protected:Npn \__draw_vec:nn #1#2
1098 {
1099   \__draw_point_process:nn { \__draw_vec:nnn {#1} } {#2}
1100 }
1101 \cs_new_protected:Npn \__draw_vec:nnn #1#2#3
1102 {
1103   \dim_set:cn { l__draw_ #1 vec_x_dim } {#2}
1104   \dim_set:cn { l__draw_ #1 vec_y_dim } {#3}
1105 }

```

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

Initialise the vectors.

```
1106 \draw_xvec:n { 1cm , 0cm }
1107 \draw_yvec:n { 0cm , 1cm }
1108 \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 1109 \cs_new:Npn \draw_point_vec:nn #1#2
__draw_point_vec:ff 1110 { \__draw_point_vec:ff { \fp_eval:n {#1} } { \fp_eval:n {#2} } }
\draw_point_vec:nnn 1111 \cs_new:Npn \__draw_point_vec:nn #1#2
__draw_point_vec:nnn 1112 {
__draw_point_vec:fff 1113   \__draw_point_to_dim:n
1114   {
1115     #1 * \l__draw_xvec_x_dim + #2 * \l__draw_yvec_x_dim ,
1116     #1 * \l__draw_xvec_y_dim + #2 * \l__draw_yvec_y_dim
1117   }
1118 }
1119 \cs_generate_variant:Nn \__draw_point_vec:nn { ff }
1120 \cs_new:Npn \draw_point_vec:nnn #1#2#3
1121 {
1122   \__draw_point_vec:fff
1123   { \fp_eval:n {#1} } { \fp_eval:n {#2} } { \fp_eval:n {#3} }
1124 }
1125 \cs_new:Npn \__draw_point_vec:nnn #1#2#3
1126 {
1127   \__draw_point_to_dim:n
1128   {
1129     #1 * \l__draw_xvec_x_dim
1130     + #2 * \l__draw_yvec_x_dim
1131     + #3 * \l__draw_zvec_x_dim
1132   ,
1133     #1 * \l__draw_xvec_y_dim
1134     + #2 * \l__draw_yvec_y_dim
1135     + #3 * \l__draw_zvec_y_dim
1136   }
1137 }
1138 \cs_generate_variant:Nn \__draw_point_vec:nnn { fff }
```

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

```
\draw_point_vec_polar:nn Much the same as the core polar approach.
\draw_point_vec_polar:nnn 1139 \cs_new:Npn \draw_point_vec_polar:nn #1#2
__draw_point_vec_polar:nnn 1140 { \draw_point_vec_polar:nnn {#1} {#1} {#2} }
__draw_point_vec_polar:fnn 1141 \cs_new:Npn \draw_point_vec_polar:nnn #1#2#3
1142 { \__draw_draw_vec_polar:fnn { \fp_eval:n {#3} } {#1} {#2} }
1143 \cs_new:Npn \__draw_draw_vec_polar:nnn #1#2#3
1144 {
1145   \__draw_point_to_dim:n
1146   {
1147     cosd(#1) * (#2) * \l__draw_xvec_x_dim ,
1148     sind(#1) * (#3) * \l__draw_yvec_y_dim
1149   }
1150 }
1151 \cs_generate_variant:Nn \__draw_draw_vec_polar:nnn { f }
```

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

5.7 Transformations

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

```

1152 \cs_new:Npn \draw_point_transform:n #1
1153 {
1154   \__draw_point_process:nn
1155   { \__draw_point_transform:nn } {#1}
1156 }
1157 \cs_new:Npn \__draw_point_transform:nn #1#2
1158 {
1159   \bool_if:NTF \l__draw_matrix_active_bool
1160   {
1161     \__draw_point_to_dim:n
1162     {
1163       (
1164         \l__draw_matrix_a_fp * #1
1165         + \l__draw_matrix_c_fp * #2
1166         + \l__draw_xshift_dim
1167       )
1168       ,
1169       (
1170         \l__draw_matrix_b_fp * #1
1171         + \l__draw_matrix_d_fp * #2
1172         + \l__draw_yshift_dim
1173       )
1174     }
1175   }
1176   {
1177     \__draw_point_to_dim:n
1178     {
1179       (#1, #2)
1180       + ( \l__draw_xshift_dim , \l__draw_yshift_dim )
1181     }
1182   }
1183 }

```

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

`__draw_point_transform_noshift:n` A version with no shift: used for internal purposes.

```

1184 \cs_new:Npn \__draw_point_transform_noshift:n #1
1185 {
1186   \__draw_point_process:nn
1187   { \__draw_point_transform_noshift:nn } {#1}
1188 }
1189 \cs_new:Npn \__draw_point_transform_noshift:nn #1#2
1190 {
1191   \bool_if:NTF \l__draw_matrix_active_bool
1192   {
1193     \__draw_point_to_dim:n

```



```

1194         {
1195         (
1196             \l__draw_matrix_a_fp * #1
1197         + \l__draw_matrix_c_fp * #2
1198         )
1199         ,
1200         (
1201             \l__draw_matrix_b_fp * #1
1202         + \l__draw_matrix_d_fp * #2
1203         )
1204     }
1205 }
1206 { \__draw_point_to_dim:n { (#1, #2) } }
1207 }

```

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

```
1208 </initex | package>
```

6 l3draw-scopes implementation

```
1209 <*initex | package>
```

```
1210 <@@=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 1211 \dim_new:N \g__draw_xmax_dim
\g__draw_ymin_dim 1212 \dim_new:N \g__draw_xmin_dim
1213 \dim_new:N \g__draw_ymax_dim
1214 \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.

```
1215 \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.

```

1216 \box_new:N \l__draw_main_box
1217 \box_new:N \l__draw_layer_main_box

```

(End definition for `\l__draw_layer_main_box`.)

`\g__draw_id_int` The drawing number.

```
1218 \int_new:N \g__draw_id_int
```

(End definition for `\g__draw_id_int`.)

`__draw_reset_bb:` A simple auxiliary.

```

1219 \cs_new_protected:Npn \__draw_reset_bb:
1220 {
1221   \dim_gset:Nn \g__draw_xmax_dim { -\c_max_dim }
1222   \dim_gset:Nn \g__draw_xmin_dim { \c_max_dim }
1223   \dim_gset:Nn \g__draw_ymax_dim { -\c_max_dim }
1224   \dim_gset:Nn \g__draw_ymin_dim { \c_max_dim }
1225 }

```

(End definition for __draw_reset_bb:.)

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

`\draw_end:`

```

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

```

```

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

```

(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   1276 \dim_new:N \l__draw_linewidth_dim
\l__draw_stroke_color_tl 1277 \tl_new:N \l__draw_fill_color_tl
                          1278 \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: 1279 \cs_new_protected:Npn \draw_scope_begin:
1280 {
1281     \__draw_backend_scope_begin:
1282     \group_begin:
1283         \dim_set_eq:NN \l__draw_linewidth_dim \g__draw_linewidth_dim
1284         \draw_path_scope_begin:
1285     }
1286     \cs_new_protected:Npn \draw_scope_end:
1287     {
1288         \draw_path_scope_end:
1289         \dim_gset_eq:NN \g__draw_linewidth_dim \l__draw_linewidth_dim
1290     \group_end:
1291     \__draw_backend_scope_end:
1292     }

```

(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  1293 \dim_new:N \l__draw_xmax_dim
\l__draw_ymax_dim  1294 \dim_new:N \l__draw_xmin_dim
\l__draw_ymin_dim  1295 \dim_new:N \l__draw_ymax_dim
                  1296 \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:
1297 \cs_new_protected:Npn \__draw_scope_bb_begin:
1298 {
1299   \group_begin:
1300   \dim_set_eq:NN \l__draw_xmax_dim \g__draw_xmax_dim
1301   \dim_set_eq:NN \l__draw_xmin_dim \g__draw_xmin_dim
1302   \dim_set_eq:NN \l__draw_ymax_dim \g__draw_ymax_dim
1303   \dim_set_eq:NN \l__draw_ymin_dim \g__draw_ymin_dim
1304   \__draw_reset_bb:
1305 }
1306 \cs_new_protected:Npn \__draw_scope_bb_end:
1307 {
1308   \dim_gset_eq:NN \g__draw_xmax_dim \l__draw_xmax_dim
1309   \dim_gset_eq:NN \g__draw_xmin_dim \l__draw_xmin_dim
1310   \dim_gset_eq:NN \g__draw_ymax_dim \l__draw_ymax_dim
1311   \dim_gset_eq:NN \g__draw_ymin_dim \l__draw_ymin_dim
1312   \group_end:
1313 }

```

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

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

```

\draw_suspend_end:
1314 \cs_new_protected:Npn \draw_suspend_begin:
1315 {
1316   \__draw_scope_bb_begin:
1317   \draw_path_scope_begin:
1318   \draw_transform_matrix_reset:
1319   \draw_transform_shift_reset:
1320   \__draw_layers_save:
1321 }
1322 \cs_new_protected:Npn \draw_suspend_end:
1323 {
1324   \__draw_layers_restore:
1325   \draw_path_scope_end:
1326   \__draw_scope_bb_end:
1327 }

```

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

1328 \langle /initex | package \rangle

7 l3draw-softpath implementation

1329 \langle *initex | package \rangle

1330 \langle @@=draw \rangle

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.
1331 \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.
1332 \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.
1333 \bool_new:N \g__draw_softpath_corners_bool
(End definition for \g__draw_softpath_corners_bool.)

\__draw_softpath_add:n
\__draw_softpath_add:o 1334 \cs_new_protected:Npn \__draw_softpath_add:n
\__draw_softpath_add:x 1335 { \tl_build_gput_right:Nn \g__draw_softpath_main_tl }
1336 \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: 1337 \cs_new_protected:Npn \__draw_softpath_use:
1338 {
1339   \tl_build_get:NN \g__draw_softpath_main_tl \l__draw_softpath_internal_tl
1340   \l__draw_softpath_internal_tl
1341 }
1342 \cs_new_protected:Npn \__draw_softpath_clear:
1343 {
1344   \tl_build_gclear:N \g__draw_softpath_main_tl
1345   \bool_gset_false:N \g__draw_softpath_corners_bool
1346 }
(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 1347 \dim_new:N \g__draw_softpath_lastx_dim
1348 \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.
1349 \bool_new:N \g__draw_softpath_move_bool
1350 \bool_gset_true:N \g__draw_softpath_move_bool
(End definition for \g__draw_softpath_move_bool.)
```

| | |
|--|---|
| <pre> __draw_softpath_curveto:nnnnnn __draw_softpath_lineto:nn __draw_softpath_moveto:nn __draw_softpath_rectangle:nnnn __draw_softpath_roundpoint:nn __draw_softpath_roundpoint:VV </pre> | <p>The various parts of a path expressed as the appropriate soft path functions.</p> <pre> 1351 \cs_new_protected:Npn __draw_softpath_closepath: 1352 { 1353 __draw_softpath_add:x 1354 { 1355 __draw_softpath_close_op:nn 1356 { \dim_use:N \g__draw_softpath_lastx_dim } 1357 { \dim_use:N \g__draw_softpath_lasty_dim } 1358 } 1359 } 1360 \cs_new_protected:Npn __draw_softpath_curveto:nnnnnn #1#2#3#4#5#6 1361 { 1362 __draw_softpath_add:n 1363 { 1364 __draw_softpath_curveto_opi:nn {#1} {#2} 1365 __draw_softpath_curveto_opii:nn {#3} {#4} 1366 __draw_softpath_curveto_opiii:nn {#5} {#6} 1367 } 1368 } 1369 \cs_new_protected:Npn __draw_softpath_lineto:nn #1#2 1370 { 1371 __draw_softpath_add:n 1372 { __draw_softpath_lineto_op:nn {#1} {#2} } 1373 } 1374 \cs_new_protected:Npn __draw_softpath_moveto:nn #1#2 1375 { 1376 __draw_softpath_add:n 1377 { __draw_softpath_moveto_op:nn {#1} {#2} } 1378 \bool_if:NT \g__draw_softpath_move_bool 1379 { 1380 \dim_gset:Nn \g__draw_softpath_lastx_dim {#1} 1381 \dim_gset:Nn \g__draw_softpath_lasty_dim {#2} 1382 } 1383 } 1384 \cs_new_protected:Npn __draw_softpath_rectangle:nnnn #1#2#3#4 1385 { 1386 __draw_softpath_add:n 1387 { 1388 __draw_softpath_rectangle_opi:nn {#1} {#2} 1389 __draw_softpath_rectangle_opii:nn {#3} {#4} 1390 } 1391 } 1392 \cs_new_protected:Npn __draw_softpath_roundpoint:nn #1#2 1393 { 1394 __draw_softpath_add:n 1395 { __draw_softpath_roundpoint_op:nn {#1} {#2} } 1396 \bool_gset_true:N \g__draw_softpath_corners_bool 1397 } 1398 \cs_generate_variant:Nn __draw_softpath_roundpoint:nn { VV } </pre> |
|--|---|

(End definition for __draw_softpath_curveto:nnnnnn and others.)

| | |
|--|---|
| <pre> __draw_softpath_close_op:nn __draw_softpath_curveto_opi:nn __draw_softpath_curveto_opii:nn __draw_softpath_curveto_opiii:nn __draw_softpath_lineto_op:nn __draw_softpath_moveto_op:nn __draw_softpath_roundpoint_op:nn __draw_softpath_rectangle_opi:nn __draw_softpath_rectangle_opii:nn __draw_softpath_curveto_opi:nnNnnNnn __draw_softpath_rectangle_opi:nnNnn </pre> | <p>The markers for operations: all the top-level ones take two arguments. The support</p> |
|--|---|

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

```

1399 \cs_new_protected:Npn \__draw_softpath_close_op:nn #1#2
1400 { \__draw_backend_closepath: }
1401 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nn #1#2
1402 { \__draw_softpath_curveto_opi:nnNnnNnn {#1} {#2} }
1403 \cs_new_protected:Npn \__draw_softpath_curveto_opi:nnNnnNnn #1#2#3#4#5#6#7#8
1404 { \__draw_backend_curveto:nnnnnn {#1} {#2} {#4} {#5} {#7} {#8} }
1405 \cs_new_protected:Npn \__draw_softpath_curveto_opii:nn #1#2
1406 { \__draw_softpath_curveto_opii:nn }
1407 \cs_new_protected:Npn \__draw_softpath_curveto_opiii:nn #1#2
1408 { \__draw_softpath_curveto_opiii:nn }
1409 \cs_new_protected:Npn \__draw_softpath_lineto_op:nn #1#2
1410 { \__draw_backend_lineto:nn {#1} {#2} }
1411 \cs_new_protected:Npn \__draw_softpath_moveto_op:nn #1#2
1412 { \__draw_backend_moveto:nn {#1} {#2} }
1413 \cs_new_protected:Npn \__draw_softpath_roundpoint_op:nn #1#2 { }
1414 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nn #1#2
1415 { \__draw_softpath_rectangle_opi:nnNnn {#1} {#2} }
1416 \cs_new_protected:Npn \__draw_softpath_rectangle_opi:nnNnn #1#2#3#4#5
1417 { \__draw_backend_rectangle:nnnn {#1} {#2} {#4} {#5} }
1418 \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.

```
1419 \tl_new:N \l__draw_softpath_main_tl
```

(End definition for `\l__draw_softpath_main_tl`.)

`\l__draw_softpath_part_tl` Data structures.

```

1420 \tl_new:N \l__draw_softpath_part_tl
1421 \tl_new:N \l__draw_softpath_curve_end_tl

```

(End definition for `\l__draw_softpath_part_tl`.)

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

```

\l__draw_softpath_lasty_fp
\l__draw_softpath_corneri_dim
\l__draw_softpath_cornerii_dim
\l__draw_softpath_first_tl
\l__draw_softpath_move_tl

```

```

1422 \fp_new:N \l__draw_softpath_lastx_fp
1423 \fp_new:N \l__draw_softpath_lasty_fp
1424 \dim_new:N \l__draw_softpath_corneri_dim
1425 \dim_new:N \l__draw_softpath_cornerii_dim
1426 \tl_new:N \l__draw_softpath_first_tl
1427 \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.

```
1428 \fp_const:Nn \c__draw_softpath_arc_fp { 4/3 * (sqrt(2) - 1) }
```

(End definition for \c__draw_softpath_arc_fp.)

```

\__draw_softpath_round_corners: Rounding corners on a path means going through the entire path and adjusting it. As
\__draw_softpath_round_loop:Nnn such, we avoid this entirely if we know there are no corners to deal with. Assuming there
\__draw_softpath_round_action:nn is work to do, we recover the existing path and start a loop.
\__draw_softpath_round_action:Nnn
1429 \cs_new_protected:Npn \__draw_softpath_round_corners:
1430 {
1431   \bool_if:NT \g__draw_softpath_corners_bool
1432   {
1433     \group_begin:
1434     \tl_clear:N \l__draw_softpath_main_tl
1435     \tl_clear:N \l__draw_softpath_part_tl
1436     \fp_zero:N \l__draw_softpath_lastx_fp
1437     \fp_zero:N \l__draw_softpath_lasty_fp
1438     \tl_clear:N \l__draw_softpath_first_tl
1439     \tl_clear:N \l__draw_softpath_move_tl
1440     \tl_build_get:NN \g__draw_softpath_main_tl \l__draw_softpath_internal_tl
1441     \exp_after:wN \__draw_softpath_round_loop:Nnn
1442     \l__draw_softpath_internal_tl
1443     \q_recursion_tail ? ?
1444     \q_recursion_stop
1445   \group_end:
1446   }
1447   \bool_gset_false:N \g__draw_softpath_corners_bool
1448 }

```

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.

```

1449 \cs_new_protected:Npn \__draw_softpath_round_loop:Nnn #1#2#3
1450 {
1451   \quark_if_recursion_tail_stop_do:Nn #1 { \__draw_softpath_round_end: }
1452   \token_if_eq_meaning:NNTF #1 \__draw_softpath_roundpoint_op:nn
1453   { \__draw_softpath_round_action:nn {#2} {#3} }
1454   {
1455     \tl_if_empty:NT \l__draw_softpath_first_tl
1456     { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1457     \fp_set:Nn \l__draw_softpath_lastx_fp {#2}
1458     \fp_set:Nn \l__draw_softpath_lasty_fp {#3}
1459     \token_if_eq_meaning:NNTF #1 \__draw_softpath_moveto_op:nn
1460     {
1461       \tl_put_right:No \l__draw_softpath_main_tl
1462       \l__draw_softpath_move_tl
1463       \tl_put_right:No \l__draw_softpath_main_tl
1464       \l__draw_softpath_part_tl
1465       \tl_set:Nn \l__draw_softpath_move_tl { #1 {#2} {#3} }
1466       \tl_clear:N \l__draw_softpath_first_tl
1467       \tl_clear:N \l__draw_softpath_part_tl
1468     }
1469     { \tl_put_right:Nn \l__draw_softpath_part_tl { #1 {#2} {#3} } }
1470   \__draw_softpath_round_loop:Nnn
1471 }

```



```

1472 }
1473 \cs_new_protected:Npn \__draw_softpath_round_action:nn #1#2
1474 {
1475   \dim_set:Nn \l__draw_softpath_corneri_dim {#1}
1476   \dim_set:Nn \l__draw_softpath_cornerii_dim {#2}
1477   \bool_lazy_and:nnTF
1478     { \dim_compare_p:nNn \l__draw_softpath_corneri_dim = { 0pt } }
1479     { \dim_compare_p:nNn \l__draw_softpath_cornerii_dim = { 0pt } }
1480     { \__draw_softpath_round_loop:Nnn }
1481     { \__draw_softpath_round_action:Nnn }
1482 }

```

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.

```

1483 \cs_new_protected:Npn \__draw_softpath_round_action:Nnn #1#2#3
1484 {
1485   \tl_if_empty:NT \l__draw_softpath_first_tl
1486   { \tl_set:Nn \l__draw_softpath_first_tl { {#2} {#3} } }
1487   \token_if_eq_meaning:NNTF #1 \__draw_softpath_curveto_opi:nn
1488   { \__draw_softpath_round_action_curveto:NnnNnn }
1489   {
1490     \token_if_eq_meaning:NNTF #1 \__draw_softpath_close_op:nn
1491     { \__draw_softpath_round_action_close: }
1492     {
1493       \token_if_eq_meaning:NNTF #1 \__draw_softpath_lineto_op:nn
1494       { \__draw_softpath_round_lookahead:NnnNnn }
1495       { \__draw_softpath_round_loop:Nnn }
1496     }
1497   }
1498   #1 {#2} {#3}
1499 }

```

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.

```

1500 \cs_new_protected:Npn \__draw_softpath_round_action_curveto:NnnNnn
1501   #1#2#3#4#5#6
1502 {
1503   \tl_put_right:Nn \l__draw_softpath_part_tl
1504     { #1 {#2} {#3} #4 {#5} {#6} }
1505   \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1506   \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1507   \__draw_softpath_round_lookahead:NnnNnn
1508 }
1509 \cs_new_protected:Npn \__draw_softpath_round_action_close:
1510 {
1511   \bool_lazy_and:nnTF
1512     { ! \tl_if_empty_p:N \l__draw_softpath_first_tl }
1513     { ! \tl_if_empty_p:N \l__draw_softpath_move_tl }
1514     {
1515       \exp_after:wN \__draw_softpath_round_close:nn
1516       \l__draw_softpath_first_tl
1517     }
1518     { \__draw_softpath_round_loop:Nnn }
1519 }

```

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.

```

1520 \cs_new_protected:Npn \__draw_softpath_round_lookahead:NnnNnn #1#2#3#4#5#6
1521 {
1522   \bool_lazy_any:nTF
1523   {
1524     { \token_if_eq_meaning_p:NN #4 \__draw_softpath_lineto_op:nn }
1525     { \token_if_eq_meaning_p:NN #4 \__draw_softpath_curveto_opi:nn }
1526     { \token_if_eq_meaning_p:NN #4 \__draw_softpath_close_op:nn }
1527   }
1528   {
1529     \__draw_softpath_round_calc:NnnNnn
1530     \__draw_softpath_round_loop:Nnn
1531     {#5} {#6}
1532   }
1533   {
1534     \token_if_eq_meaning:NNTF #4 \__draw_softpath_roundpoint_op:nn
1535     { \__draw_softpath_round_roundpoint:NnnNnnNnn }
1536     { \__draw_softpath_round_loop:Nnn }
1537   }
1538   #1 {#2} {#3}
1539   #4 {#5} {#6}
1540 }
1541 \cs_new_protected:Npn \__draw_softpath_round_roundpoint:NnnNnnNnn
1542 #1#2#3#4#5#6#7#8#9
1543 {
1544   \__draw_softpath_round_calc:NnnNnn
1545   \__draw_softpath_round_loop:Nnn
1546   {#8} {#9}
1547   #1 {#2} {#3}
1548   #4 {#5} {#6} #7 {#8} {#9}
1549 }

```

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.

```

1550 \cs_new_protected:Npn \__draw_softpath_round_calc:NnnNnn #1#2#3#4#5#6
1551 {
1552   \tl_set:Nx \l__draw_softpath_curve_end_tl
1553   {
1554     \draw_point_interpolate_distance:nnn
1555     \l__draw_softpath_cornerii_dim
1556     { #5 , #6 } { #2 , #3 }
1557   }
1558   \tl_put_right:Nx \l__draw_softpath_part_tl
1559   {
1560     \exp_not:N #4

```

```

1561     \_draw_softpath_round_calc:fVnnnn
1562     {
1563         \draw_point_interpolate_distance:nnn
1564         \l__draw_softpath_corneri_dim
1565         { #5 , #6 }
1566         {
1567             \l__draw_softpath_lastx_fp ,
1568             \l__draw_softpath_lasty_fp
1569         }
1570     }
1571     \l__draw_softpath_curve_end_t1
1572     {#5} {#6} {#2} {#3}
1573 }
1574 \fp_set:Nn \l__draw_softpath_lastx_fp {#5}
1575 \fp_set:Nn \l__draw_softpath_lasty_fp {#6}
1576 #1
1577 }

```

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).

```

1578 \cs_new:Npn \_draw_softpath_round_calc:nnnnnn #1#2#3#4#5#6
1579 {
1580     \_draw_softpath_round_calc:nnnnw {#3} {#4} {#5} {#6}
1581     #1 \q_mark #2 \q_stop
1582 }
1583 \cs_generate_variant:Nn \_draw_softpath_round_calc:nnnnnn { fV }

```

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

```

1584 \cs_new:Npn \_draw_softpath_round_calc:nnnnw
1585 #1#2#3#4 #5 , #6 \q_mark #7 , #8 \q_stop
1586 {
1587     {#5} {#6}
1588     \exp_not:N \_draw_softpath_curveto_opi:nn
1589     {
1590         \fp_to_dim:n
1591         { #5 + \c__draw_softpath_arc_fp * ( #1 - #5 ) }
1592     }
1593     {
1594         \fp_to_dim:n
1595         { #6 + \c__draw_softpath_arc_fp * ( #2 - #6 ) }
1596     }
1597     \exp_not:N \_draw_softpath_curveto_opii:nn
1598     {
1599         \fp_to_dim:n
1600         { #7 + \c__draw_softpath_arc_fp * ( #1 - #7 ) }
1601     }
1602     {
1603         \fp_to_dim:n
1604         { #8 + \c__draw_softpath_arc_fp * ( #2 - #8 ) }
1605     }
1606     \exp_not:N \_draw_softpath_curveto_opiii:nn
1607     {#7} {#8}
1608 }

```

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.

```

1609 \cs_new_protected:Npn \__draw_softpath_round_close:nn #1#2
1610 {
1611   \use:x
1612   {
1613     \__draw_softpath_round_calc:NnnNnn
1614     {
1615       \tl_set:Nx \exp_not:N \l__draw_softpath_move_tl
1616       {
1617         \__draw_softpath_moveto_op:nn
1618         \exp_not:N \exp_after:wN
1619         \exp_not:N \__draw_softpath_round_close:w
1620         \exp_not:N \l__draw_softpath_curve_end_tl
1621         \exp_not:N \q_stop
1622       }
1623       \use:x
1624       {
1625         \exp_not:N \exp_not:N \exp_not:N \use_i:nnnn
1626         {
1627           \__draw_softpath_round_loop:Nnn
1628           \__draw_softpath_close_op:nn
1629           \exp_not:N \exp_after:wN
1630           \exp_not:N \__draw_softpath_round_close:w
1631           \exp_not:N \l__draw_softpath_curve_end_tl
1632           \exp_not:N \q_stop
1633         }
1634       }
1635     }
1636     {#1} {#2}
1637     \__draw_softpath_lineto_op:nn
1638     \exp_after:wN \use_none:n \l__draw_softpath_move_tl
1639   }
1640 }
1641 \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.

```

1642 \cs_new_protected:Npn \__draw_softpath_round_end:
1643 {
1644   \tl_put_right:No \l__draw_softpath_main_tl
1645   \l__draw_softpath_move_tl
1646   \tl_put_right:No \l__draw_softpath_main_tl
1647   \l__draw_softpath_part_tl
1648   \tl_build_gclear:N \g__draw_softpath_main_tl
1649   \__draw_softpath_add:o \l__draw_softpath_main_tl
1650 }

```

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

```

1651 </initex | package>

```

8 l3draw-state implementation

1652 `*initex | package`

1653 `\@@=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`, `\pgfsetinnerstroke`

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.

1654 `\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.

1655 `\dim_new:N \l_draw_default_linewidth_dim`

1656 `\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.

1657 `\cs_new_protected:Npn \draw_linewidth:n #1`

1658 `{`

1659 `\dim_gset:Nn \g__draw_linewidth_dim { \fp_to_dim:n {#1} }`

1660 `__draw_backend_linewidth:n \g__draw_linewidth_dim`

1661 `}`

(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` 1662 `\cs_new_protected:Npn \draw_dash_pattern:nn #1#2`

1663 `{`

1664 `\group_begin:`

1665 `\seq_set_from_clist:Nn \l__draw_tmp_seq {#1}`

1666 `\seq_set_map:NNn \l__draw_tmp_seq \l__draw_tmp_seq`

1667 `{ \fp_to_dim:n {##1} }`

1668 `\use:x`

1669 `{`

1670 `__draw_backend_dash_pattern:nn`

1671 `{ \seq_use:Nn \l__draw_tmp_seq { , } }`

1672 `{ \fp_to_dim:n {#2} }`

1673 `}`

1674 `\group_end:`

1675 `}`

1676 `\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.

1677 `\cs_new_protected:Npn \draw_miterlimit:n #1`

1678 `{ __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: 1679 \cs_new_protected:Npn \draw_cap_but: { \__draw_backend_cap_but: }
\draw_cap_round: 1680 \cs_new_protected:Npn \draw_cap_rectangle: { \__draw_backend_cap_rectangle: }
\draw_evenodd_rule: 1681 \cs_new_protected:Npn \draw_cap_round: { \__draw_backend_cap_round: }
\draw_nonzero_rule: 1682 \cs_new_protected:Npn \draw_evenodd_rule: { \__draw_backend_evenodd_rule: }
\draw_join_bevel: 1683 \cs_new_protected:Npn \draw_nonzero_rule: { \__draw_backend_nonzero_rule: }
\draw_join_miter: 1684 \cs_new_protected:Npn \draw_join_bevel: { \__draw_backend_join_bevel: }
\draw_join_round: 1685 \cs_new_protected:Npn \draw_join_miter: { \__draw_backend_join_miter: }
1686 \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.

```
1687 \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 1688 \cs_new_eq:NN \draw_color:n \color_select:n
\draw_color_stroke:n 1689 \cs_new_protected:Npn \draw_color_fill:n #1
\__draw_color:nn 1690 { \__draw_color:nn { fill } {#1} }
\__draw_color_aux:nn 1691 \cs_new_protected:Npn \draw_color_stroke:n #1
\__draw_color_aux:Vn 1692 { \__draw_color:nn { stroke } {#1} }
\__draw_color:nw 1693 \cs_new_protected:Npn \__draw_color:nn #1#2
1694 {
1695   \color_parse:nN {#2} \l__draw_color_tmp_tl
1696   \__draw_color_aux:Vn \l__draw_color_tmp_tl {#1}
1697 }
\__draw_select_cmyk:nw 1698 \cs_new_protected:Npn \__draw_color_aux:nn #1#2
1699 { \__draw_color:nw {#2} #1 \q_stop }
\__draw_select_gray:nw 1700 \cs_generate_variant:Nn \__draw_color_aux:nn { V }
\__draw_select_rgb:nw 1701 \cs_new_protected:Npn \__draw_color:nw #1#2 ~ #3 \q_stop
1702 { \use:c { \__draw_color_ #2 :nw } {#1} #3 \q_stop }
\__draw_split_select:nw 1703 \cs_new_protected:Npn \__draw_color_cmyk:nw #1#2 ~ #3 ~ #4 ~ #5 \q_stop
1704 { \use:c { \__draw_backend_color_ #1_cmyk:nnnn } {#2} {#3} {#4} {#5} }
1705 \cs_new_protected:Npn \__draw_color_gray:nw #1#2 \q_stop
1706 { \use:c { \__draw_backend_color_ #1_gray:n } {#2} }
1707 \cs_new_protected:Npn \__draw_color_rgb:nw #1#2 ~ #3 ~ #4 \q_stop
1708 { \use:c { \__draw_backend_color_ #1_rgb:nnn } {#2} {#3} {#4} }
1709 \cs_new_protected:Npn \__draw_color_spot:nw #1#2 ~ #3 \q_stop
1710 { \use:c { \__draw_backend_color_ #1_spot:nn } {#2} {#3} }

```

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

```
1711 \</initex | package>
```

9 l3draw-transforms implementation

```
1712 \< *initex | package>
```

```
1713 \< @@=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.
- `\pgfflowlevelsynccm`, `\pgfflowlevel`: 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.

```
1714 \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 1715 \fp_new:N \l__draw_matrix_a_fp
\l__draw_matrix_c_fp 1716 \fp_new:N \l__draw_matrix_b_fp
\l__draw_xshift_dim 1717 \fp_new:N \l__draw_matrix_c_fp
\l__draw_yshift_dim 1718 \fp_new:N \l__draw_matrix_d_fp
1719 \dim_new:N \l__draw_xshift_dim
1720 \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: 1721 \cs_new_protected:Npn \draw_transform_matrix_reset:
1722 {
1723   \fp_set:Nn \l__draw_matrix_a_fp { 1 }
1724   \fp_zero:N \l__draw_matrix_b_fp
1725   \fp_zero:N \l__draw_matrix_c_fp
1726   \fp_set:Nn \l__draw_matrix_d_fp { 1 }
1727 }
1728 \cs_new_protected:Npn \draw_transform_shift_reset:
1729 {
1730   \dim_zero:N \l__draw_xshift_dim
1731   \dim_zero:N \l__draw_yshift_dim
1732 }
1733 \draw_transform_matrix_reset:
1734 \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:n` 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 1735 \cs_new_protected:Npn \draw_transform_matrix_absolute:nnnn #1#2#3#4
1736 {
1737   \fp_set:Nn \l__draw_matrix_a_fp {#1}
1738   \fp_set:Nn \l__draw_matrix_b_fp {#2}
1739   \fp_set:Nn \l__draw_matrix_c_fp {#3}
1740   \fp_set:Nn \l__draw_matrix_d_fp {#4}
1741   \bool_lazy_all:nTF
1742   {
1743     { \fp_compare_p:nNn \l__draw_matrix_a_fp = \c_one_fp }
1744     { \fp_compare_p:nNn \l__draw_matrix_b_fp = \c_zero_fp }
1745     { \fp_compare_p:nNn \l__draw_matrix_c_fp = \c_zero_fp }

```

```

1746         { \fp_compare_p:nNn \l__draw_matrix_d_fp = \c_one_fp }
1747     }
1748     { \bool_set_false:N \l__draw_matrix_active_bool }
1749     { \bool_set_true:N \l__draw_matrix_active_bool }
1750 }
1751 \cs_new_protected:Npn \draw_transform_shift_absolute:n #1
1752 {
1753     \__draw_point_process:nn
1754     { \__draw_transform_shift_absolute:nn } {#1}
1755 }
1756 \cs_new_protected:Npn \__draw_transform_shift_absolute:nn #1#2
1757 {
1758     \dim_set:Nn \l__draw_xshift_dim {#1}
1759     \dim_set:Nn \l__draw_yshift_dim {#2}
1760 }

```

(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
1761 \cs_new_protected:Npn \draw_transform_matrix:nnnn #1#2#3#4
1762 {
1763     \use:x
1764     {
1765         \__draw_transform:nnnn
1766         { \fp_eval:n {#1} }
1767         { \fp_eval:n {#2} }
1768         { \fp_eval:n {#3} }
1769         { \fp_eval:n {#4} }
1770     }
1771 }
1772 \cs_new_protected:Npn \__draw_transform:nnnn #1#2#3#4
1773 {
1774     \use:x
1775     {
1776         \draw_transform_matrix_absolute:nnnn
1777         { #1 * \l__draw_matrix_a_fp + #2 * \l__draw_matrix_c_fp }
1778         { #1 * \l__draw_matrix_b_fp + #2 * \l__draw_matrix_d_fp }
1779         { #3 * \l__draw_matrix_a_fp + #4 * \l__draw_matrix_c_fp }
1780         { #3 * \l__draw_matrix_b_fp + #4 * \l__draw_matrix_d_fp }
1781     }
1782 }
1783 \cs_new_protected:Npn \draw_transform_shift:n #1
1784 {
1785     \__draw_point_process:nn
1786     { \__draw_transform_shift:nn } {#1}
1787 }
1788 \cs_new_protected:Npn \__draw_transform_shift:nn #1#2
1789 {
1790     \dim_set:Nn \l__draw_xshift_dim { \l__draw_xshift_dim + #1 }
1791     \dim_set:Nn \l__draw_yshift_dim { \l__draw_yshift_dim + #2 }
1792 }

```


(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 1793 \cs_new_protected:Npn \draw_transform_matrix_invert:
\__draw_transform_invert:f 1794 {
\draw_transform_shift_invert: 1795 \bool_if:NT \l__draw_matrix_active_bool
1796 {
1797 \__draw_transform_invert:f
1798 {
1799 \fp_eval:n
1800 {
1801 1 /
1802 (
1803 \l__draw_matrix_a_fp * \l__draw_matrix_d_fp
1804 - \l__draw_matrix_b_fp * \l__draw_matrix_c_fp
1805 )
1806 }
1807 }
1808 }
1809 }
1810 \cs_new_protected:Npn \__draw_transform_invert:n #1
1811 {
1812 \fp_set:Nn \l__draw_matrix_a_fp
1813 { \l__draw_matrix_d_fp * #1 }
1814 \fp_set:Nn \l__draw_matrix_b_fp
1815 { -\l__draw_matrix_b_fp * #1 }
1816 \fp_set:Nn \l__draw_matrix_c_fp
1817 { -\l__draw_matrix_c_fp * #1 }
1818 \fp_set:Nn \l__draw_matrix_d_fp
1819 { \l__draw_matrix_a_fp * #1 }
1820 }
1821 \cs_generate_variant:Nn \__draw_transform_invert:n { f }
1822 \cs_new_protected:Npn \draw_transform_shift_invert:
1823 {
1824 \dim_set:Nn \l__draw_xshift_dim { -\l__draw_xshift_dim }
1825 \dim_set:Nn \l__draw_yshift_dim { -\l__draw_yshift_dim }
1826 }

```

(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.

```

1827 \cs_new_protected:Npn \draw_transform_triangle:nnn #1#2#3
1828 {
1829 \__draw_point_process:nnn
1830 {
1831 \__draw_point_process:nn
1832 { \__draw_tranform_triangle:nnnnnn }
1833 {#1}
1834 }
1835 {#2} {#3}
1836 }
1837 \cs_new_protected:Npn \__draw_tranform_triangle:nnnnnn #1#2#3#4#5#6

```

```

1838 {
1839   \use:x
1840   {
1841     \draw_transform_matrix_absolute:nnnn
1842     { #3 - #1 }
1843     { #4 - #2 }
1844     { #5 - #1 }
1845     { #6 - #2 }
1846     \draw_transform_shift_absolute:n { #1 , #2 }
1847   }
1848 }

```

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

```

\draw_transform_scale:n Lots of shortcuts.
\draw_transform_xscale:n 1849 \cs_new_protected:Npn \draw_transform_scale:n #1
\draw_transform_yscale:n 1850 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { #1 } }
\draw_transform_xshift:n 1851 \cs_new_protected:Npn \draw_transform_xscale:n #1
\draw_transform_yshift:n 1852 { \draw_transform_matrix:nnnn { #1 } { 0 } { 0 } { 1 } }
\draw_transform_xslant:n 1853 \cs_new_protected:Npn \draw_transform_yscale:n #1
\draw_transform_yslant:n 1854 { \draw_transform_matrix:nnnn { 1 } { 0 } { 0 } { #1 } }
1855 \cs_new_protected:Npn \draw_transform_xshift:n #1
1856 { \draw_transform_shift:n { #1 , Opt } }
1857 \cs_new_protected:Npn \draw_transform_yshift:n #1
1858 { \draw_transform_shift:n { Opt , #1 } }
1859 \cs_new_protected:Npn \draw_transform_xslant:n #1
1860 { \draw_transform_matrix:nnnn { 1 } { 0 } { #1 } { 1 } }
1861 \cs_new_protected:Npn \draw_transform_yslant:n #1
1862 { \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 1863 \cs_new_protected:Npn \draw_transform_rotate:n #1
\_draw_transform_rotate:f 1864 { \_draw_transform_rotate:f { \fp_eval:n {#1} } }
\_draw_transform_rotate:nn 1865 \cs_new_protected:Npn \_draw_transform_rotate:n #1
\_draw_transform_rotate:ff 1866 {
1867   \_draw_transform_rotate:ff
1868   { \fp_eval:n { cosd(#1) } }
1869   { \fp_eval:n { sind(#1) } }
1870 }
1871 \cs_generate_variant:Nn \_draw_transform_rotate:n { f }
1872 \cs_new_protected:Npn \_draw_transform_rotate:nn #1#2
1873 { \draw_transform_matrix:nnnn {#1} {#2} { -#2 } { #1 } }
1874 \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 ??.)

```

1875 </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, 784, 1036, 1040, 1064, 1067

bool commands:

`\bool_gset_eq:Nn` 765

`\bool_gset_false:N` 1345, 1447

`\bool_gset_true:N` 1350, 1396

`\bool_if:NTF`

. 19, 113, 192, 231, 680, 696,

697, 701, 1159, 1191, 1378, 1431, 1795

`\bool_lazy_all:nTF` 1741

`\bool_lazy_and:nnTF`

. 223, 675, 1477, 1511

`\bool_lazy_any:nTF` 1522

`\bool_lazy_or:nnTF` . 556, 651, 684, 689

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

642, 643, 743, 1215, 1333, 1349, 1714

`\bool_set_eq:Nn` 757

`\bool_set_false:N`

. 94, 226, 662, 663, 664, 683, 1748

`\bool_set_true:N` 96,

227, 668, 706, 710, 711, 1234, 1749

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` 1264

`\box_move_up:nn` 49

`\box_new:N` 11, 78, 79, 1216, 1217

`\box_set_dp:Nn` 53, 1269

`\box_set_eq:Nn` 143

`\box_set_ht:Nn` 52, 1267

`\box_set_wd:Nn` 54, 126, 1270

`\box_use_drop:N`

. 50, 55, 100, 127, 132, 1265, 1273

`\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, 1250

coffin commands:

`\coffin_typeset:Nnnnn` 62

`\coffin_wd:N` 64

color commands:

`\color_parse:nN` 1695

`\color_select:n` 1688

cs commands:

`\cs_generate_variant:Nn`

. 418, 605, 638, 843, 851, 893, 912,

919, 927, 934, 943, 949, 961, 964,

982, 1000, 1008, 1014, 1035, 1049,

1063, 1084, 1119, 1138, 1151, 1336,

1398, 1583, 1700, 1821, 1871, 1874

`\cs_if_exist:NTF` 667

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

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

788, 794, 796, 798, 805, 807, 809,

817, 819, 822, 831, 836, 839, 841,

844, 845, 847, 849, 852, 854, 860,

869, 875, 885, 894, 900, 905, 913,

920, 928, 935, 944, 950, 956, 962,

965, 971, 980, 983, 989, 994, 1001,

1009, 1015, 1021, 1027, 1041, 1050,

1056, 1068, 1070, 1079, 1109, 1111,

1120, 1125, 1139, 1141, 1143, 1152,

1157, 1184, 1189, 1578, 1584, 1641

`\cs_new_eq:Nn` 1688

`\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, 704, 708, 713, 720, 744,

762, 1091, 1093, 1095, 1097, 1101,

1219, 1226, 1247, 1279, 1286, 1297,

1306, 1314, 1322, 1334, 1337, 1342,

1351, 1360, 1369, 1374, 1384, 1392,

1399, 1401, 1403, 1405, 1407, 1409,

1411, 1413, 1414, 1416, 1418, 1429,

1449, 1473, 1483, 1500, 1509, 1520,

1541, 1550, 1609, 1642, 1657, 1662,

1677, 1679, 1680, 1681, 1682, 1683,

1684, 1685, 1686, 1689, 1691, 1693,

1698, 1701, 1703, 1705, 1707, 1709,

1721, 1728, 1735, 1751, 1756, 1761,

1772, 1783, 1788, 1793, 1810, 1822,

1827, 1837, 1849, 1851, 1853, 1855,

1857, 1859, 1861, 1863, 1865, 1872

D

dim commands:

| | |
|--|--|
| <code>\dim_abs:n</code> | 594, 595 |
| <code>\dim_compare:nNnTF</code> | 601, 608, 722, 1254 |
| <code>\dim_compare_p:nNn</code> | 224, 225, 1478, 1479 |
| <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, 724, 1221, 1222, 1223, 1224, 1380, 1381, 1659 |
| <code>\dim_gset_eq:NN</code> | 769, 770, 771, 772, 773, 774, 775, 776, 1289, 1308, 1309, 1310, 1311 |
| <code>\dim_gzero:N</code> .. | 1256, 1257, 1258, 1259 |
| <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, 735, 736, 737, 738, 739, 740, 741, 742, 1085, 1086, 1087, 1088, 1089, 1090, 1211, 1212, 1213, 1214, 1276, 1293, 1294, 1295, 1296, 1347, 1348, 1424, 1425, 1654, 1655, 1719, 1720 |
| <code>\dim_set:Nn</code> | 221, 222, 1103, 1104, 1475, 1476, 1656, 1758, 1759, 1790, 1791, 1824, 1825 |
| <code>\dim_set_eq:NN</code> | 747, 748, 749, 750, 751, 752, 753, 754, 1283, 1300, 1301, 1302, 1303 |
| <code>\dim_step_inline:nnnn</code> | 621, 629 |
| <code>\dim_use:N</code> .. | 722, 727, 729, 1356, 1357 |
| <code>\dim_zero:N</code> | 1730, 1731 |
| <code>\c_max_dim</code> | 206, 207, 208, 209, 722, 1221, 1222, 1223, 1224, 1254 |

draw commands:

| | |
|--|-------------------------------|
| <code>\l_draw_bb_update_bool</code> | 19, 192, 676, 683, 1215, 1234 |
| <code>\draw_begin:</code> | 1226 |
| <code>\draw_box_use:N</code> | 12 |
| <code>\draw_cap_but:</code> | 1241, 1679 |
| <code>\draw_cap_rectangle:</code> | 1679 |
| <code>\draw_cap_round:</code> | 1679 |
| <code>\draw_coffin_use:Nnn</code> | 58 |
| <code>\draw_color:n</code> | 1239, 1688 |
| <code>\draw_color_fill:n</code> | 1688 |
| <code>\draw_color_stroke:n</code> | 1688 |
| <code>\draw_dash_pattern:nn</code> ... | 1244, 1662 |
| <code>\l_draw_default_linewidth_dim</code> ... | 103, 1238, 1655 |
| <code>\draw_end:</code> | 1226 |
| <code>\draw_evenodd_rule:</code> | 1679 |
| <code>\draw_join_bevel:</code> | 1679 |
| <code>\draw_join_miter:</code> | 1242, 1679 |
| <code>\draw_join_round:</code> | 1679 |

| | |
|---|------------------------------|
| <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, 1250 |
| <code>\draw_linewidth:n</code> | 103, 1238, 1657 |
| <code>\draw_miterlimit:n</code> | 1243, 1677 |
| <code>\draw_nonzero_rule:</code> | 1240, 1679 |
| <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> | 744, 1284, 1317 |
| <code>\draw_path_scope_end:</code> | 744, 1288, 1325 |
| <code>\draw_path_use:n</code> | 644 |
| <code>\draw_path_use_clear:n</code> | 644 |
| <code>\draw_point_interpolate_arcaxes:nnnnnn</code> | 983 |
| <code>\draw_point_interpolate_curve:nnnnnn</code> | 1015 |
| <code>\draw_point_interpolate_curve:nnnnnnnn</code> | 1015 |
| <code>\draw_point_interpolate_curve_-auxi:nnnnnnnnnn</code> | 1015 |
| <code>\draw_point_interpolate_curve_-auxii:nnnnnnnnnn</code> | 1015 |
| <code>\draw_point_interpolate_curve_-auxiii:nnnnnnnn</code> | 1015 |
| <code>\draw_point_interpolate_curve_-auxiv:nnnnnnnn</code> | 1015 |
| <code>\draw_point_interpolate_curve_-auxv:nnw</code> | 1015 |
| <code>\draw_point_interpolate_curve_-auxvi:n</code> | 1015 |
| <code>\draw_point_interpolate_curve_-auxvii:nnnnnnnnnn</code> | 1015 |
| <code>\draw_point_interpolate_curve_-auxviii:nnnnnnnn</code> | 1015 |

| | |
|---|-----------------------------------|
| <code>\draw_point_interpolate_distance:nnn</code> | 682 |
| 965, 1554, 1563 | |
| <code>\draw_point_interpolate_line:nnn</code> | 1400 |
| 950 | |
| <code>\draw_point_intersect_circles:nnnnn</code> | 1404 |
| 894 | |
| <code>\draw_point_intersect_lines:nnnn</code> | 1670 |
| 869 | |
| <code>\draw_point_polar:nn</code> | 687 |
| 845 | |
| <code>\draw_point_polar:nnn</code> | 1252 |
| 426, 432, 436, 442, 845 | |
| <code>\draw_point_transform:n</code> | 1682 |
| 23, 26, 29, 32, 242, 254, 270, 271, | |
| 272, 304, 305, 431, 435, 491, 562, 1152 | |
| <code>\draw_point_unit_vector:n</code> | 1684 |
| 852, 978 | |
| <code>\draw_point_vec:nn</code> | 1685 |
| 1109 | |
| <code>\draw_point_vec:nnn</code> | 1686 |
| 1109 | |
| <code>\draw_point_vec_polar:nn</code> | 1410 |
| 1139 | |
| <code>\draw_point_vec_polar:nnn</code> | 1660 |
| 1139 | |
| <code>\draw_scope_begin:</code> | 1678 |
| 1279 | |
| <code>\draw_scope_end:</code> | 1412 |
| 1286 | |
| <code>\draw_suspend_begin:</code> | 1683 |
| 1314 | |
| <code>\draw_suspend_end:</code> | 1417 |
| 1314 | |
| <code>\draw_transform_matrix:nnnn</code> | 130, 1281 |
| 1761, | |
| 1850, 1852, 1854, 1860, 1862, 1873 | |
| <code>\draw_transform_matrix_absolute:nnnn</code> | 133, 1291 |
| 1735, 1776, 1841 | |
| <code>\draw_transform_matrix_invert:</code> | 12, 63 |
| 1793 | |
| <code>\draw_transform_matrix_reset:</code> | 1688 |
| 1235, 1318, 1721 | |
| <code>\draw_transform_rotate:n</code> | 1688 |
| 1863 | |
| <code>\draw_transform_scale:n</code> | 1688 |
| 1849 | |
| <code>\draw_transform_shift:n</code> | 1688 |
| 1761, 1856, 1858 | |
| <code>\draw_transform_shift_absolute:n</code> | 1703 |
| 1735, 1846 | |
| <code>\draw_transform_shift_invert:</code> | 1705 |
| 1793 | |
| <code>\draw_transform_shift_reset:</code> | 1707 |
| 1236, 1319, 1721 | |
| <code>\draw_transform_triangle:nnn</code> | 1709 |
| 484, 1827 | |
| <code>\draw_transform_xscale:n</code> | 1687, 1695, 1696 |
| 1849 | |
| <code>\draw_transform_xshift:n</code> | 1687, 1695, 1696 |
| 1849 | |
| <code>\draw_transform_xslant:n</code> | 1687, 1695, 1696 |
| 1849 | |
| <code>\draw_transform_yscale:n</code> | 1687, 1695, 1696 |
| 1849 | |
| <code>\draw_transform_yshift:n</code> | 1687, 1695, 1696 |
| 1849 | |
| <code>\draw_transform_yslant:n</code> | 1687, 1695, 1696 |
| 1849 | |
| <code>\draw_xvec:n</code> | 1091, 1106 |
| 1091, 1106 | |
| <code>\draw_yvec:n</code> | 1091, 1107 |
| 1091, 1107 | |
| <code>\draw_zvec:n</code> | 1091, 1108 |
| 1091, 1108 | |
| draw internal commands: | |
| <code>__draw_backend_begin:</code> | 1231 |
| 39 | |
| <code>__draw_backend_box_use:Nnnnn</code> | 1679 |
| 1679 | |
| <code>__draw_backend_cap_but:</code> | 1680 |
| 1680 | |
| <code>__draw_backend_cap_rectangle:</code> | 1681 |
| 1681 | |
| <code>__draw_backend_cap_round:</code> | 1681 |
| 1681 | |
| <code>__draw_backend_clip:</code> | 682 |
| 682 | |
| <code>__draw_backend_closepath:</code> | 1400 |
| 1400 | |
| <code>__draw_backend_curveto:nnnnnn</code> | 1404 |
| 1404 | |
| <code>__draw_backend_dash_pattern:nn</code> | 1670 |
| 1670 | |
| <code>__draw_backend_discardpath:</code> | 687 |
| 687 | |
| <code>__draw_backend_end:</code> | 1252 |
| 1252 | |
| <code>__draw_backend_evenodd_rule:</code> | 1682 |
| 1682 | |
| <code>__draw_backend_join_bevel:</code> | 1684 |
| 1684 | |
| <code>__draw_backend_join_miter:</code> | 1685 |
| 1685 | |
| <code>__draw_backend_join_round:</code> | 1686 |
| 1686 | |
| <code>__draw_backend_lineto:nn</code> | 1410 |
| 1410 | |
| <code>__draw_backend_linewidth:n</code> | 1660 |
| 1660 | |
| <code>__draw_backend_miterlimit:n</code> | 1678 |
| 1678 | |
| <code>__draw_backend_moveto:nn</code> | 1412 |
| 1412 | |
| <code>__draw_backend_nonzero_rule:</code> | 1683 |
| 1683 | |
| <code>__draw_backend_rectangle:nnnn</code> | 1417 |
| 1417 | |
| <code>__draw_backend_scope_begin:</code> | 130, 1281 |
| 130, 1281 | |
| <code>__draw_backend_scope_end:</code> | 133, 1291 |
| 133, 1291 | |
| <code>__draw_box_use:Nnnnn</code> | 12, 63 |
| 12, 63 | |
| <code>__draw_color:nn</code> | 1688 |
| 1688 | |
| <code>__draw_color:nw</code> | 1688 |
| 1688 | |
| <code>__draw_color_aux:nn</code> | 1688 |
| 1688 | |
| <code>__draw_color_cmyk:nw</code> | 1703 |
| 1703 | |
| <code>__draw_color_gray:nw</code> | 1705 |
| 1705 | |
| <code>__draw_color_rgb:nw</code> | 1707 |
| 1707 | |
| <code>__draw_color_spot:nw</code> | 1709 |
| 1709 | |
| <code>\l__draw_color_tmp_tl</code> | 1687, 1695, 1696 |
| 1687, 1695, 1696 | |
| <code>\l__draw_corner_arc_bool</code> | 218, 226, 227, 231, 557 |
| 218, 226, 227, 231, 557 | |
| <code>\l__draw_corner_xarc_dim</code> | 216, 221, 224, 234 |
| 216, 221, 224, 234 | |
| <code>\l__draw_corner_yarc_dim</code> | 216, 222, 225, 235 |
| 216, 222, 225, 235 | |
| <code>__draw_draw_polar:nnn</code> | 845 |
| 845 | |
| <code>__draw_draw_vec_polar:nnn</code> | 1142, 1143, 1151 |
| 1142, 1143, 1151 | |
| <code>\l__draw_fill_color_tl</code> | 1276 |
| 1276 | |
| <code>\g__draw_id_int</code> | 1218, 1229 |
| 1218, 1229 | |
| <code>\l__draw_layer_close_bool</code> | 84, 94, 96, 113 |
| 84, 94, 96, 113 | |
| <code>\l__draw_layer_main_box</code> | 126, 127, 1216, 1245 |
| 126, 127, 1216, 1245 | |
| <code>\l__draw_layer_tl</code> | 82, 93, 97 |
| 82, 93, 97 | |
| <code>\g__draw_layers_clist</code> | 85 |
| 85 | |
| <code>__draw_layers_insert:</code> | 120, 1251 |
| 120, 1251 | |
| <code>__draw_layers_restore:</code> | 137, 1324 |
| 137, 1324 | |
| <code>__draw_layers_save:</code> | 137, 1320 |
| 137, 1320 | |
| <code>\g__draw_linewidth_dim</code> | 730, 1283, 1289, 1654, 1659, 1660 |
| 730, 1283, 1289, 1654, 1659, 1660 | |
| <code>\l__draw_linewidth_dim</code> | 1276, 1283, 1289 |
| 1276, 1283, 1289 | |
| <code>\l__draw_main_box</code> | 1216, 1230, |
| 1216, 1230, | |
| 1261, 1265, 1267, 1269, 1270, 1273 | |

| | |
|---|---|
| \l__draw_matrix_a_fp | \l__draw_path_lasty_dim 735 , 748 , 776 |
| . 40 , 1164 , 1196 , 1715 , 1723 , 1737 , | __draw_path_lineto:nn 238 , 290 |
| 1743 , 1777 , 1779 , 1803 , 1812 , 1819 | __draw_path_mark_corner: |
| \l__draw_matrix_active_bool | 229 , 258 , 267 , 284 , 295 , 313 , 384 |
| 558 , 1159 , 1191 , 1714 , 1748 , 1749 , 1795 | __draw_path_moveto:nn |
| \l__draw_matrix_b_fp | 238 , 288 , 499 , 507 |
| . 41 , 1170 , 1201 , 1715 , 1724 , 1738 , | __draw_path_rectangle:nnnn ... 552 |
| 1744 , 1778 , 1780 , 1804 , 1814 , 1815 | __draw_path_rectangle_corners:nnnn |
| \l__draw_matrix_c_fp | 581 |
| . 42 , 1165 , 1197 , 1715 , 1725 , 1739 , | __draw_path_rectangle_corners:nnnnn |
| 1745 , 1777 , 1779 , 1804 , 1816 , 1817 | 584 , 587 |
| \l__draw_matrix_d_fp | __draw_path_rectangle_rounded:nnnn |
| . 43 , 1171 , 1202 , 1718 , 1726 , 1740 , | 552 |
| 1746 , 1778 , 1780 , 1803 , 1813 , 1818 | __draw_path_reset_limits: |
| __draw_path_arc:nnnn 343 | 182 , 656 , 755 , 1233 |
| __draw_path_arc:nnNnn 343 | \l__draw_path_tmp_tl |
| \c__draw_path_arc_60_fp 343 | 173 , 421 , 444 , 463 , 467 , 471 , 475 |
| \c__draw_path_arc_90_fp 343 | \l__draw_path_tmpa_fp |
| __draw_path_arc_add:nnnn 343 | 173 , 309 , 319 , 331 |
| __draw_path_arc_aux_add:nn | \l__draw_path_tmpb_fp |
| 450 , 456 , 468 , 473 | 173 , 310 , 326 , 335 |
| __draw_path_arc_auxi:nnnnNnn ... | __draw_path_update_last:nn |
| 343 , 370 , 377 | 211 , 248 , 261 , 280 , 570 |
| __draw_path_arc_auxii:nnnNnnnn 343 | __draw_path_update_limits:nn ... |
| __draw_path_arc_auxiii:nn 343 | 22 , 25 , 28 , 31 , |
| __draw_path_arc_auxiv:nnnn ... 343 | 182 , 246 , 259 , 276 , 277 , 278 , 567 , 568 |
| __draw_path_arc_auxv:nn 343 | __draw_path_use:n 644 |
| __draw_path_arc_auxvi:nn 343 | __draw_path_use_action_draw: .. 644 |
| \l__draw_path_arc_delta_fp 343 | __draw_path_use_action_fillstroke: |
| \l__draw_path_arc_start_fp 343 | 644 |
| __draw_path_curveto:nnnn 300 | \l__draw_path_use_bb_bool 642 |
| __draw_path_curveto:nnnnnn 238 , 296 , 314 , 444 , 511 , 521 , 531 , 541 | \l__draw_path_use_clear_bool 642 , 701 |
| \c__draw_path_curveto_a_fp 300 | \l__draw_path_use_clip_bool |
| \c__draw_path_curveto_b_fp 300 | 639 , 662 , 680 |
| __draw_path_ellipse:nnnnnn ... 487 | \l__draw_path_use_fill_bool 639 , 663 , 685 , 690 , 696 , 710 |
| __draw_path_ellipse_arci:nnnnnn 487 | __draw_path_use_stroke_bb: ... 644 |
| __draw_path_ellipse_arcii:nnnnnn | __draw_path_use_stroke_bb- |
| 487 | aux:NnN 644 |
| __draw_path_ellipse_arciiii:nnnnnn | \l__draw_path_use_stroke_bool ... |
| 487 | 639 , 664 , 677 , 686 , 691 , 697 , 706 , 711 |
| __draw_path_ellipse_arciv:nnnnnn | \g__draw_path_xmax_dim |
| 487 | 178 , 184 , 185 , 206 , 749 , 771 |
| \c__draw_path_ellipse_fp 487 | \l__draw_path_xmax_dim . 735 , 749 , 771 |
| __draw_path_grid_auxi:nnnnnn .. 589 | \g__draw_path_xmin_dim |
| __draw_path_grid_auxii:nnnnnn . 589 | 178 , 186 , 187 , 207 , 750 , 772 |
| __draw_path_grid_auxiii:nnnnnn 589 | \l__draw_path_xmin_dim . 735 , 750 , 772 |
| __draw_path_grid_auxiiii:nnnnnn 589 | \g__draw_path_ymax_dim |
| __draw_path_grid_auxiv:nnnnnnnn 589 | 178 , 188 , 189 , 208 , 751 , 773 |
| \g__draw_path_lastx_dim | \l__draw_path_ymax_dim . 735 , 751 , 773 |
| 176 , 213 , 318 , 451 , 457 , 747 , 775 | \g__draw_path_ymin_dim |
| \l__draw_path_lastx_dim 735 , 747 , 775 | 178 , 190 , 191 , 209 , 752 , 774 |
| \g__draw_path_lasty_dim | \l__draw_path_ymin_dim . 735 , 752 , 774 |
| 176 , 214 , 325 , 452 , 458 , 748 , 776 | |

| | |
|---------------------------------------|---|
| _draw_point_interpolate_- | _draw_point_process:nn |
| arcaxes_auxi:nnnnnnnnn | 21, 24, 27, 30, 240, 252, |
| 983 | 288, 290, 422, 438, 788, 853, 967, |
| _draw_point_interpolate_- | 973, 1099, 1154, 1186, 1753, 1785, 1831 |
| arcaxes_auxii:nnnnnnnnn | _draw_point_process:nnn |
| 983 | 302, |
| _draw_point_interpolate_- | 428, 554, 583, 591, 788, 896, 952, 1829 |
| arcaxes_auxiii:nnnnnnn | _draw_point_process:nnnn |
| 983 | 265, 293, 489, 788, 985 |
| _draw_point_interpolate_- | _draw_point_process:nnnnn |
| arcaxes_auxiv:nnnnnnnnn | 788, 871, 1017 |
| 983 | _draw_point_process_auxi:nn |
| _draw_point_interpolate_curve_- | 788 |
| auxi:nnnnnnnnn | _draw_point_process_auxii:nw |
| 1018, 1021 | 788 |
| _draw_point_interpolate_curve_- | _draw_point_process_auxiii:nnn |
| auxii:nnnnnnnnn | 788 |
| 1023, 1027, 1035 | _draw_point_process_auxiv:nw |
| _draw_point_interpolate_curve_- | 788 |
| auxiii:nnnnnnn | _draw_point_process_auxv:nnnn |
| 1030, 1041, 1049 | 788 |
| _draw_point_interpolate_curve_- | _draw_point_process_auxvi:nw |
| auxiv:nnnnnnn | 788 |
| 1043, 1044, 1045, 1050 | _draw_point_process_auxvii:nnnnn |
| _draw_point_interpolate_curve_- | 788 |
| auxv:nnw | _draw_point_process_auxviii:nw |
| 1052, 1056, 1063 | 788 |
| _draw_point_interpolate_curve_- | _draw_point_to_dim:n |
| auxvi:n | 791, |
| 1047, 1068 | 801, 802, 812, 813, 814, 825, 826, |
| _draw_point_interpolate_curve_- | 827, 828, 839, 1011, 1081, 1113, |
| auxvii:nnnnnnnn | 1127, 1145, 1161, 1177, 1193, 1206 |
| 1069, 1070 | _draw_point_to_dim_aux:n |
| _draw_point_interpolate_curve_- | 839 |
| auxviii:nnnnnnn | _draw_point_to_dim_aux:w |
| 1072, 1079, 1084 | 839 |
| _draw_point_interpolate_- | _draw_point_transform:nn |
| distance:nnnn | 1152 |
| 968, 971 | _draw_point_transform_noshift:n |
| _draw_point_interpolate_- | 425, 441, 492, 493, 1184 |
| distance:nnnnn | _draw_point_transform_noshift:nn |
| 965, 975 | 1184 |
| _draw_point_interpolate_- | _draw_point_unit_vector:nn |
| distance:nnnnnn | 852 |
| 965 | _draw_point_unit_vector:nnn |
| _draw_point_interpolate_line_- | 852 |
| aux:nnnnn | _draw_point_vec:nn |
| 950 | 1109 |
| _draw_point_interpolate_line_- | _draw_point_vec:nnn |
| aux:nnnnnn | 1109 |
| 950 | _draw_point_vec_polar:nnn |
| _draw_point_intersect_circles_- | 1139 |
| auxi:nnnnnnnn | _draw_reset_bb: |
| 894 | 1219, 1232, 1304 |
| _draw_point_intersect_circles_- | _draw_scope_bb_begin: |
| auxii:nnnnnnnn | 1297, 1316 |
| 894 | _draw_scope_bb_end: |
| _draw_point_intersect_circles_- | 1297, 1326 |
| auxiii:nnnnnnnn | _draw_select_cmyk:nw |
| 894 | 1688 |
| _draw_point_intersect_circles_- | _draw_select_gray:nw |
| auxiv:nnnnnnnnn | 1688 |
| 894 | _draw_select_rgb:nw |
| _draw_point_intersect_circles_- | 1688 |
| auxv:nnnnnnnnnn | _draw_softpath_add:n |
| 894 | 768, 1334, 1353, |
| _draw_point_intersect_circles_- | 1362, 1371, 1376, 1386, 1394, 1649 |
| auxvi:nnnnnnnnn | _c_draw_softpath_arc_fp |
| 894 | 1428, 1591, 1595, 1600, 1604 |
| _draw_point_intersect_circles_- | _draw_softpath_clear: |
| auxvii:nnnnnnnn | 655, 702, 760, 764, 1237, 1337 |
| 894 | _draw_softpath_close_op:nn |
| _draw_point_intersect_lines:nnnnnn | 1355, 1399, 1490, 1526, 1628 |
| 869 | _draw_softpath_closepath: |
| _draw_point_intersect_lines:nnnnnnnn | 285, 506, 1351 |
| 869 | _l_draw_softpath_corneri_dim |
| _draw_point_intersect_lines_- | 1422, 1475, 1478, 1564 |
| aux:nnnnnnn | |
| 869 | |

| | |
|---|--|
| \l__draw_softpath_cornerii_dim .. | __draw_softpath_rectangle:nnnn .. |
| 1422 , 1476 , 1479 , 1555 | 569 , 1351 |
| \g__draw_softpath_corners_bool .. | __draw_softpath_rectangle_- |
| 759 , 766 , 1333 , 1345 , 1396 , 1431 , 1447 | opi:nn 1388 , 1399 |
| \l__draw_softpath_corners_bool .. | __draw_softpath_rectangle_- |
| 735 , 758 , 767 | opi:nnNnn 1399 |
| \l__draw_softpath_curve_end_tl .. | __draw_softpath_rectangle_- |
| 1421 , 1552 , 1571 , 1620 , 1631 | opii:nn 1389 , 1399 |
| __draw_softpath_curveto:nnnnnn .. | __draw_softpath_round_action:nn |
| 279 , 1351 | 1429 |
| __draw_softpath_curveto_opi:nn .. | __draw_softpath_round_action:Nnn |
| 1364 , 1399 , 1487 , 1525 , 1588 | 1429 |
| __draw_softpath_curveto_- | __draw_softpath_round_action_- |
| opi:nnNnnNnn 1399 | close: 1429 |
| __draw_softpath_curveto_opii:nn | __draw_softpath_round_action_- |
| 1365 , 1399 , 1597 | curveto:NnnNnn 1429 |
| __draw_softpath_curveto_- | __draw_softpath_round_calc:NnnNnn |
| opiii:nn 1366 , 1399 , 1606 | 1429 |
| \l__draw_softpath_first_tl | __draw_softpath_round_calc:nnnnnn |
| 1422 , 1438 , 1455 , | 1429 |
| 1456 , 1466 , 1485 , 1486 , 1512 , 1516 | __draw_softpath_round_calc:nnnnnw |
| \l__draw_softpath_internal_tl | 1429 |
| 1332 , 1339 , 1340 , 1440 , 1442 | __draw_softpath_round_close:nn 1429 |
| \g__draw_softpath_lastx_dim | __draw_softpath_round_close:w 1429 |
| 753 , 769 , 1347 , 1356 , 1380 | __draw_softpath_round_corners: .. |
| \l__draw_softpath_lastx_dim | 674 , 1429 |
| 741 , 753 , 769 | __draw_softpath_round_end: .. 1429 |
| \l__draw_softpath_lastx_fp | __draw_softpath_round_lookahead:NnnNnn |
| .. 1422 , 1436 , 1457 , 1505 , 1567 , 1574 | 1429 |
| \g__draw_softpath_lasty_dim | __draw_softpath_round_loop:Nnn 1429 |
| 754 , 770 , 1347 , 1357 , 1381 | __draw_softpath_round_roundpoint:NnnNnnNnn |
| \l__draw_softpath_lasty_dim | 1429 |
| 742 , 754 , 770 | __draw_softpath_roundpoint:nn .. |
| \l__draw_softpath_lasty_fp | 233 , 1351 |
| .. 1422 , 1437 , 1458 , 1506 , 1568 , 1575 | __draw_softpath_roundpoint_- |
| __draw_softpath_lineto:nn 260 , 1351 | op:nn 1395 , 1399 , 1452 , 1534 |
| __draw_softpath_lineto_op:nn | __draw_softpath_use: 679 , 1337 |
| 1372 , 1399 , 1493 , 1524 , 1637 | __draw_split_select:nw 1688 |
| \g__draw_softpath_main_tl | \l__draw_stroke_color_tl 1276 |
| 756 , 1331 , 1335 , 1339 , 1344 , 1440 , 1648 | \l__draw_tmp_box 11 , 35 , 46 , |
| \l__draw_softpath_main_tl | 50 , 52 , 53 , 54 , 55 , 61 , 63 , 64 , 65 , 66 , 67 |
| 19 , 756 , 768 , 1419 , | \l__draw_tmp_seq 1662 |
| 1434 , 1461 , 1463 , 1644 , 1646 , 1649 | __draw_tranform_triangle:nnnnnn |
| \g__draw_softpath_move_bool | 1832 , 1837 |
| 1349 , 1378 | __draw_transform:nnnn 1761 |
| \l__draw_softpath_move_tl | __draw_transform_invert:n ... 1793 |
| 1422 , 1439 , | __draw_transform_rotate:n ... 1863 |
| 1462 , 1465 , 1513 , 1615 , 1638 , 1645 | __draw_transform_rotate:nn ... 1863 |
| __draw_softpath_moveto:nn 247 , 1351 | __draw_transform_shift:nn ... 1761 |
| __draw_softpath_moveto_op:nn | __draw_transform_shift_absolute:nn |
| 1377 , 1399 , 1459 , 1617 | 1735 |
| \l__draw_softpath_part_tl | __draw_vec:nn 1091 |
| 1420 , 1435 , | __draw_vec:nnn 1091 |
| 1464 , 1467 , 1469 , 1503 , 1558 , 1647 | |

`\g__draw_xmax_dim` 194,
 195, 1211, 1221, 1256, 1271, 1300, 1308
`\l__draw_xmax_dim` . . . 1293, 1300, 1308
`\g__draw_xmin_dim`
 196, 197, 1211, 1222,
 1254, 1257, 1263, 1271, 1301, 1309
`\l__draw_xmin_dim` . . . 1293, 1301, 1309
`\l__draw_xshift_dim` 48, 1166,
 1180, 1715, 1730, 1758, 1790, 1824
`\l__draw_xvec_x_dim`
 1085, 1115, 1129, 1147
`\l__draw_xvec_y_dim` . 1085, 1116, 1133
`\g__draw_ymax_dim` 198,
 199, 1211, 1223, 1258, 1268, 1302, 1310
`\l__draw_ymax_dim` . . . 1293, 1302, 1310
`\g__draw_ymin_dim` . 200, 201, 1211,
 1224, 1259, 1264, 1268, 1303, 1311
`\l__draw_ymin_dim` . . . 1293, 1303, 1311
`\l__draw_yshift_dim` 49, 1172,
 1180, 1715, 1731, 1759, 1791, 1825
`\l__draw_yvec_x_dim` . 1085, 1115, 1130
`\l__draw_yvec_y_dim`
 1085, 1116, 1134, 1148
`\l__draw_zvec_x_dim` 1085, 1131
`\l__draw_zvec_y_dim` 1085, 1135

E

`\end` 167, 782
 exp commands:
`\exp_after:wN`
 444, 462, 1441, 1515, 1618, 1629, 1638
`\exp_args:Nf` 790, 856
`\exp_args:Nff` 800
`\exp_args:Nfff` 811
`\exp_args:Nffff` 824
`\exp_args:NNNV` 1249
`\exp_not:N` 1560, 1588,
 1597, 1606, 1615, 1618, 1619, 1620,
 1621, 1625, 1629, 1630, 1631, 1632

F

fp commands:

`\fp_compare:nNnTF` 358, 368, 862
`\fp_compare_p:nNn`
 1743, 1744, 1745, 1746
`\fp_const:Nn`
 341, 342, 480, 481, 549, 1428
`\fp_eval:n` . 350, 351, 372, 379, 388,
 840, 848, 857, 878, 879, 880, 881,
 882, 883, 903, 908, 909, 916, 923,
 924, 931, 938, 940, 953, 958, 976,
 992, 997, 1004, 1005, 1024, 1031,
 1053, 1054, 1073, 1074, 1075, 1076,
 1110, 1123, 1142, 1678, 1766, 1767,
 1768, 1769, 1799, 1864, 1868, 1869
`\fp_new:N` 174, 175, 478,
 479, 1422, 1423, 1715, 1716, 1717, 1718
`\fp_set:Nn` 309, 310, 364, 365,
 445, 446, 1457, 1458, 1505, 1506,
 1574, 1575, 1723, 1726, 1737, 1738,
 1739, 1740, 1812, 1814, 1816, 1818
`\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, 1590,
 1594, 1599, 1603, 1659, 1667, 1672
`\fp_use:N` 40, 41, 42, 43, 549
`\fp_while_do:nNnn` 366
`\fp_zero:N` 1436, 1437, 1724, 1725
`\c_one_fp` 1743, 1746
`\c_zero_fp` 862, 1744, 1745

G

group commands:

`\group_begin:` 34, 60, 90,
 101, 746, 1228, 1282, 1299, 1433, 1664
`\group_end:` 56, 68, 115,
 118, 777, 1274, 1290, 1312, 1445, 1674

H

hbox commands:

`\hbox_gset:Nw` 99
`\hbox_gset_end:` 116
`\hbox_set:Nn` 35, 46, 61, 1261
`\hbox_set:Nw` 1230, 1245
`\hbox_set_end:` 1249, 1253

I

int commands:

`\int_gincr:N` 1229
`\int_if_odd:nTF` 939
`\int_new:N` 1218

M

mode commands:

`\mode_leave_vertical:` 1272

msg commands:

`\msg_error:nnn` 76, 107, 108, 671
`\msg_new:nnn` 162
`\msg_new:nnnn` 159, 164, 779

P

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

