

# The `spath3` package

Andrew Stacey  
[stacey@math.ntnu.no](mailto:stacey@math.ntnu.no)

v1.1 from 2016/02/19

## 1 Introduction

The `spath3` package is intended as a library for manipulating PGF's *soft paths*. In between defining a path and using it, PGF stores a path as a *soft path* where all the defining structure has been resolved into the basic operations but these have not yet been written to the output file. They can therefore still be manipulated by `TEX`, and as they have a very rigid form (and limited vocabulary), they are relatively easy to modify. This package provides some methods for working with these paths. It is not really intended for use by end users but as a foundation on which other packages can be built. As examples, the `calligraphy` package and the `knot` package are included. The first of these simulates a calligraphic pen stroking a path. The second can be used to draw knot (and similar) diagrams.

The format of a soft path is a sequence of triples of the form `\macro {dimension}{dimension}`. The macro is one of a short list, the dimensions are coordinates in points. There are certain further restrictions, particularly that every path must begin with a `move to`, and Bézier curves consist of three triples.

## 2 Implementation

### 2.1 Initialisation

Load the `LATEX3` foundation and register us as a `LATEX3` package.

```
1 \NeedsTeXFormat{LaTeX2e}
2 \RequirePackage{expl3}
3 \RequirePackage{pgf}
4 \ProvidesExplPackage {spath3} {2016/02/19} {1.1} {Functions for
5 manipulating PGF soft paths}
6 \RequirePackage{xparse}
```

We need a slew of temporary variables.

```
7 \tl_new:N \l__spath_tmpa_tl
8 \tl_new:N \l__spath_tmpb_tl
9 \tl_new:N \l__spath_tmpe_tl
10 \tl_new:N \l__spath_tmpe_tl
```

```

11 \tl_new:N \l__spath_smuggle_tl
12 \dim_new:N \l__spath_tmpa_dim
13 \dim_new:N \l__spath_tmpb_dim
14 \fp_new:N \l__spath_tmpa_fp
15 \fp_new:N \l__spath_tmpb_fp
16 \int_new:N \l__spath_tmpa_int

```

We need to be able to compare against the macros that can occur in a soft path so these token lists contain them.

```

17 \tl_new:N \g__spath_moveto_tl
18 \tl_new:N \g__spath_lineto_tl
19 \tl_new:N \g__spath_curveto_tl
20 \tl_new:N \g__spath_curvetoa_tl
21 \tl_new:N \g__spath_curvetob_tl
22 \tl_new:N \g__spath_closepath_tl
23 \tl_set:Nn \g__spath_moveto_tl {\pgfsyssoftpath@movetotoken}
24 \tl_set:Nn \g__spath_lineto_tl {\pgfsyssoftpath@linetotoken}
25 \tl_set:Nn \g__spath_curveto_tl {\pgfsyssoftpath@curvetotoken}
26 \tl_set:Nn \g__spath_curvetoa_tl {\pgfsyssoftpath@curvetosupportatoken}
27 \tl_set:Nn \g__spath_curvetob_tl {\pgfsyssoftpath@curvetosupportbtoken}
28 \tl_set:Nn \g__spath_closepath_tl {\pgfsyssoftpath@closepathtoken}

```

## 2.2 Basic Structure and Methods

A soft path is a `prop`. These are lists of the attributes that we define. The first consists of all attributes, the second of those that are “moveable” in the sense that they change if we transform the path, the third are the ones that contain actual paths.

Note that if using these attributes outside an `expl3` context, the spaces should be omitted.

```

29 \tl_new:N \g__spath_attributes
30 \tl_new:N \g__spath_moveable_attributes
31 \tl_new:N \g__spath_path_attributes
32 \tl_set:Nn \g__spath_attributes {
33   {path}
34   {reverse path}
35   {length}
36   {real length}
37   {number of components}
38   {initial point}
39   {final point}
40   {initial action}
41   {final action}
42   {min bb}
43   {max bb}
44 }
45 \tl_set:Nn \g__spath_moveable_attributes {
46   {initial point}
47   {final point}
48   {min bb}

```

```

49   {max bb}
50 }
51 \tl_set:Nn \g__spath_path_attributes {
52   {path}
53   {reverse path}
54 }

```

An `spath` object is actually a `prop`. The following functions are wrappers around the underlying `prop` functions. We prefix the names to avoid clashing with other `props` that might be lying around, this is why all the `spath` methods take argument `:n` and not `:N`. Given that `spath` objects might be created inside a group but used outside it, we work globally throughout.

`\spath_new:n`

```

55 \cs_new_nopar:Npn \spath_new:n #1
56 {
57   \prop_new:c {l__spath_#1}
58 }

```

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

`\spath_clear:n`

```

59 \cs_new_nopar:Npn \spath_clear:n #1
60 {
61   \prop_gclear:c {l__spath_#1}
62 }

```

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

`\spath_clear_new:n`

```

63 \cs_new_nopar:Npn \spath_clear_new:n #1
64 {
65   \prop_gclear_new:c {l__spath_#1}
66 }

```

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

`\spath_show:n`

```

67 \cs_new_nopar:Npn \spath_show:n #1
68 {
69   \prop_show:c {l__spath_#1}
70 }

```

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

`\spath_put:nnn`

```

71 \cs_new_nopar:Npn \spath_put:nnn #1#2#3
72 {
73   \prop_gput:cnn {l__spath_#1} {#2} {#3}
74 }

```

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

`\spath_remove:nn`

```
75 \cs_new_nopar:Npn \spath_remove:nn #1#2
76 {
77   \prop_gremove:cn {l__spath_#1} {#2}
78 }
```

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

`\__spath_get:nn` This function is an internal one since the real `get` function will generate its data if it does not already exist.

```
79 \cs_new_nopar:Npn \__spath_get:nn #1#2
80 {
81   \prop_item:cn {l__spath_#1} {#2}
82 }
```

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

`\__spath_get:nnN`

```
83 \cs_new_nopar:Npn \__spath_get:nnN #1#2#3
84 {
85   \prop_get:cnN {l__spath_#1} {#2} #3
86 }
```

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

`\spath_if_in:nn`

```
87 \prg_new_conditional:Npnn \spath_if_in:nn #1#2 {p, T, F, TF}
88 {
89   \prop_if_in:cnTF {l__spath_#1} {#2}
90   { \prg_return_true: }
91   { \prg_return_false: }
92 }
```

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

`\__spath_get:nnN`

```
93 \cs_generate_variant:Nn \__prop_split:NnTF {cnTF}
94 \prg_new_protected_conditional:Npnn \__spath_get:nnN #1#2#3 {T, F, TF}
95 {
96   \__prop_split:cnTF {l__spath_#1} {#2}
97   {
98     \tl_set:Nn #3 {##2}
99     \prg_return_true:
100   }
101   { \prg_return_false: }
102 }
103 \cs_generate_variant:Nn \spath_put:nnn {nnV, nnx, nno}
104 \cs_generate_variant:Nn \__spath_get:nn {Vn}
105 \cs_generate_variant:Nn \__spath_get:nnN {VnN}
```

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

`\spath_if_exist:n`

```
106 \prg_new_conditional:Npnn \spath_if_exist:n #1 {p,T,F,TF}
107 {
108   \prop_if_exist:cTF {l__spath_#1}
109   {
110     \prg_return_true:
111   }
112   {
113     \prg_return_false:
114   }
115 }
```

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

`\spath_clone:nn` Clones an `spath`.

```
116 \cs_new_nopar:Npn \spath_clone:nn #1 #2
117 {
118   \spath_clear_new:n {#2}
119   \tl_map_inline:Nn \g__spath_attributes
120   {
121     \spath_if_in:nnT {#1} {##1}
122     {
123       \_spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
124       \spath_put:nnV {#2} {##1} \l__spath_tmpa_tl
125     }
126   }
127 }
```

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

`\spath_get_current_path:n`

```
128 \cs_new_protected_nopar:Npn \spath_get_current_path:n #1
129 {
130   \pgfsyssoftpath@getcurrentpath\l__spath_tmpa_tl
131   \spath_clear_new:n {#1}
132   \spath_put:nnV {#1} {path} \l__spath_tmpa_tl
133 }
```

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

`\spath_set_current_path:n`

```
134 \cs_new_protected_nopar:Npn \spath_set_current_path:n #1
135 {
136   \spath_get:nnN {#1} {min bb} \l__spath_tmpa_tl
137   \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl
138
139   \spath_get:nnN {#1} {max bb} \l__spath_tmpa_tl
140   \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl

```

```

141
142 \spath_get:nnN {#1} {path} \l__spath_tmpa_tl
143 \pgfsyssoftpath@setcurrentpath\l__spath_tmpa_tl
144 \pgfsyssoftpath@flushcurrentpath
145 }

```

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

`\spath_use_path:nn`

```

146 \cs_new_protected_nopar:Npn \spath_use_path:nn #1#2
147 {
148   \spath_set_current_path:n {#1}
149   \pgfusepath{#2}
150 }

```

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

`\spath_protocol_path:n`

```

151 \cs_new_protected_nopar:Npn \spath_protocol_path:n #1
152 {
153   \spath_get:nnN {#1} {min bb} \l__spath_tmpa_tl
154   \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl
155
156   \spath_get:nnN {#1} {max bb} \l__spath_tmpa_tl
157   \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl
158 }

```

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

`\spath_tikz_path:nn`

```

159 \cs_new_protected_nopar:Npn \spath_tikz_path:nn #1#2
160 {
161   \path[#1] \pgfextra{
162     \spath_get:nnN {#2} {path} \l__spath_tmpa_tl
163     \pgfsyssoftpath@setcurrentpath \l__spath_tmpa_tl
164   };
165 }
166 \cs_generate_variant:Nn \spath_tikz_path:nn {Vn}

```

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

## 2.3 Computing Information

`\spath_get:nn` The information that we store along with a soft path can be computed from it, but computing it every time is wasteful. So this is the real `\spath_get:nn` function which checks to see if we have already computed it and then either retrieves it or computes it.

```

167 \cs_new_nopar:Npn \spath_get:nn #1#2
168 {
169   \spath_if_in:nnF {#1} {#2}
170   {

```

```

171   \cs_if_exist_use:cT {spath_generate_#2:n} {{#1}}
172   }
173   \__spath_get:nn {#1} {#2}
174 }

```

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

`\spath_get:nnN` As above but leaves the result in a token list rather than in the stream.

```

175 \cs_new_nopar:Npn \spath_get:nnN #1#2#3
176 {
177   \spath_if_in:nnF {#1} {#2}
178   {
179     \cs_if_exist_use:cT {spath_generate_#2:n} {{#1}}
180   }
181   \__spath_get:nnN {#1} {#2} #3
182 }
183 \cs_generate_variant:Nn \spath_get:nnN {nnV,VnN,VnV}

```

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

The next slew of functions generate data from the original path, storing it in the `prop` for further retrieval.

`\spath_generate_length:n` Counts the number of triples in the path.

```

184 \cs_new_nopar:Npn \spath_generate_length:n #1
185 {
186   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
187   \spath_put:nnx {#1} {length} {\int_eval:n {\tl_count:N \l__spath_tmpa_tl /3 }}
188 }

```

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

`\spath_generate_reallength:n` The real length of a path is the number of triples that actually draw something (that is, the number of lines and curves).

```

189 \cs_new_nopar:Npn \spath_generate_reallength:n #1
190 {
191   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
192   \int_set:Nn \l__spath_tmpa_int {0}
193   \tl_map_inline:Nn \l__spath_tmpa_tl {
194     \tl_if_eq:nnT {##1} {\pgfsyssoftpath@linetotoken}
195     {
196       \int_incr:N \l__spath_tmpa_int
197     }
198     \tl_if_eq:nnT {##1} {\pgfsyssoftpath@curvetotoken}
199     {
200       \int_incr:N \l__spath_tmpa_int
201     }
202   }
203   \spath_put:nnx {#1} {real length} {\int_use:N \l__spath_tmpa_int}
204 }

```

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

`\spath_generate_numberofcomponents:n` A component is a continuous segment of the path, separated by moves. Successive moves are not collapsed, and zero length moves count.

```

205 \cs_new_nopar:Npn \spath_generate_numberofcomponents:n #1
206 {
207   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
208   \int_set:Nn \l__spath_tmpa_int {0}
209   \tl_map_inline:Nn \l__spath_tmpa_tl {
210     \tl_if_eq:nnT {##1} {\pgfsyssoftpath@movetotoken}
211     {
212       \int_incr:N \l__spath_tmpa_int
213     }
214   }
215   \spath_put:nx {#1} {number of components} {\int_use:N \l__spath_tmpa_int}
216 }

```

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

`\spath_generate_initialpoint:n` The starting point of the path.

```

217 \cs_new_nopar:Npn \spath_generate_initialpoint:n #1
218 {
219   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
220   \tl_clear:N \l__spath_tmpb_tl
221   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
222   \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
223   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
224   \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
225   \spath_put:nnV {#1} {initial point} \l__spath_tmpb_tl
226 }

```

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

`\spath_generate_finalpoint:n` The final point of the path.

```

227 \cs_new_nopar:Npn \spath_generate_finalpoint:n #1
228 {
229   \tl_clear:N \l__spath_tmpb_tl
230   \spath_if_in:nnTF {#1} {reverse path}
231   {
232     \__spath_get:nnN {#1} {reverse path} \l__spath_tmpa_tl
233     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
234     \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
235     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
236     \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
237   }
238   {
239     \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
240     \tl_reverse:N \l__spath_tmpa_tl
241     \tl_put_left:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
242     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
243     \tl_put_left:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
244   }

```



```

245 \spath_put:nnV {#1} {final point} \l__spath_tmpb_tl
246 }
247 \cs_generate_variant:Nn \spath_generate_finalpoint:n {V}

```

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

`\spath_generate_reversepath:n` This computes the reverse of the path. TODO: handle closed paths, possibly rectangles.

```

248 \cs_new_nopar:Npn \spath_generate_reversepath:n #1
249 {
250   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
251   \tl_clear:N \l__spath_tmpb_tl
252   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
253   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
254   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
255   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
256   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
257   \tl_put_left:Nx \l__spath_tmpb_tl
258   {
259     {\dim_use:N \l__spath_tmpa_dim}
260     {\dim_use:N \l__spath_tmpb_dim}
261   }
262   \bool_until_do:nn {
263     \tl_if_empty_p:N \l__spath_tmpa_tl
264   }
265   {
266     \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
267     \tl_set:Nn \l__spath_tmpd_tl {}
268     \tl_case:NnF \l__spath_tmpc_tl
269     {
270       \g__spath_moveto_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_moveto_tl }
271       \g__spath_lineto_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_lineto_tl }
272       \g__spath_curveto_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curvetoa_tl }
273       \g__spath_curvetoa_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curveto_tl }
274       \g__spath_curvetob_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curvetob_tl }
275     }
276     {
277       \tl_show:N \l__spath_tmpc_tl
278     }
279     \tl_put_left:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
280     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
281     \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
282     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
283     \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
284     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
285     \tl_put_left:Nx \l__spath_tmpb_tl
286     {
287       {\dim_use:N \l__spath_tmpa_dim}
288       {\dim_use:N \l__spath_tmpb_dim}
289     }
290   }

```

```

291 \tl_put_left:NV \l__spath_tmpb_tl \g__spath_moveto_tl
292 \spath_put:nnV {#1} {reverse path} \l__spath_tmpb_tl
293 }

```

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

`\spath_generate_initialaction:n` This is the first thing that the path does (after the initial move).

```

294 \cs_new_nopar:Npn \spath_generate_initialaction:n #1
295 {
296   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
297   \tl_clear:N \l__spath_tmpb_tl
298   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
299   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
300   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
301   \tl_if_empty:NF \l__spath_tmpa_tl {
302     \tl_set:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
303   }
304   \spath_put:nnV {#1} {initial action} \l__spath_tmpb_tl
305 }

```

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

`\spath_generate_final action:n` This is the last thing that the path does.

```

306 \cs_new_nopar:Npn \spath_generate_finalaction:n #1
307 {
308   \tl_clear:N \l__spath_tmpb_tl
309   \spath_if_in:nnTF {#1} {reverse path}
310   {
311     \__spath_get:nnN {#1} {reverse path} \l__spath_tmpa_tl
312     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
313   }
314   {
315     \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
316     \tl_reverse:N \l__spath_tmpa_tl
317   }
318   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
319   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
320   \tl_if_empty:NF \l__spath_tmpa_tl {
321     \tl_set:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
322   }
323   \tl_if_eq:NNT \l__spath_tmpa_tl \g__spath_curvetoa_tl
324   {
325     \tl_set_eq:NN \l__spath_tmpa_tl \g__spath_curveto_tl
326   }
327   \spath_put:nnV {#1} {final action} \l__spath_tmpb_tl
328 }

```

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

`\spath_generate_minbb:n` This computes the minimum (bottom left) of the bounding box of the path.

```

329 \cs_new_nopar:Npn \spath_generate_minbb:n #1
330 {
331   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
332   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
333   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
334   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
335   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
336   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
337   \bool_until_do:nn {
338     \tl_if_empty_p:N \l__spath_tmpa_tl
339   }
340   {
341     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
342     \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tmpa_dim}}
343     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
344     \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tmpb_dim}}
345     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
346   }
347   \tl_clear:N \l__spath_tmpb_tl
348   \tl_put_right:Nx \l__spath_tmpb_tl
349   {
350     {\dim_use:N \l__spath_tmpa_dim}
351     {\dim_use:N \l__spath_tmpb_dim}
352   }
353   \spath_put:nnV {#1} {min bb} \l__spath_tmpb_tl
354 }

```

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

`\spath_generate_max bb:n` This computes the maximum (top right) of the bounding box of the path.

```

355 \cs_new_nopar:Npn \spath_generate_maxbb:n #1
356 {
357   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
358   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
359   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
360   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
361   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
362   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
363   \bool_until_do:nn {
364     \tl_if_empty_p:N \l__spath_tmpa_tl
365   }
366   {
367     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
368     \dim_set:Nn \l__spath_tmpa_dim {\dim_max:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tmpa_dim}}
369     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
370     \dim_set:Nn \l__spath_tmpb_dim {\dim_max:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tmpb_dim}}
371     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
372   }
373   \tl_clear:N \l__spath_tmpb_tl

```

```

374 \tl_put_right:Nx \l__spath_tmpb_tl
375 {
376   {\dim_use:N \l__spath_tmpa_dim}
377   {\dim_use:N \l__spath_tmpb_dim}
378 }
379 \spath_put:nnV {#1} {max bb} \l__spath_tmpb_tl
380 }

```

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

`\spath_generate_all:n` This function generates all of the data in one fell swoop. By traversing the path just once it is quicker than doing each one individually. However, it does need to store a lot of data as it goes.

- `\l__spath_rp_tl` will hold the reversed path
- `\l__spath_l_int` will hold the length
- `\l__spath_rl_int` will hold the real length
- `\l__spath_nc_int` will hold the number of components
- `\l__spath_ip_tl` will hold the initial point
- `\l__spath_fp_tl` will hold the final point
- `\l__spath_ia_tl` will hold the initial action
- `\l__spath_fa_tl` will hold the final action
- `\l__spath_minx_dim` will hold the min x bb
- `\l__spath_miny_dim` will hold the min y bb
- `\l__spath_maxx_dim` will hold the max x bb
- `\l__spath_maxy_dim` will hold the max y bb

```

381 \tl_new:N \l__spath_rp_tl
382 \int_new:N \l__spath_l_int
383 \int_new:N \l__spath_rl_int
384 \int_new:N \l__spath_nc_int
385 \tl_new:N \l__spath_ip_tl
386 \tl_new:N \l__spath_fp_tl
387 \tl_new:N \l__spath_ia_tl
388 \tl_new:N \l__spath_fa_tl
389 \dim_new:N \l__spath_minx_dim
390 \dim_new:N \l__spath_miny_dim
391 \dim_new:N \l__spath_maxx_dim
392 \dim_new:N \l__spath_maxy_dim
393
394 \cs_new_nopar:Npn \spath_generate_all:n #1
395 {

```

```

396 \l__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
397
398 \tl_clear:N \l__spath_rp_tl
399 \int_set:Nn \l__spath_l_int {1}
400 \int_zero:N \l__spath_rl_int
401 \int_set:Nn \l__spath_nc_int {1}
402 \tl_clear:N \l__spath_ip_tl
403 \tl_clear:N \l__spath_fp_tl
404 \tl_clear:N \l__spath_ia_tl
405 \tl_clear:N \l__spath_fa_tl
406 \dim_zero:N \l__spath_minx_dim
407 \dim_zero:N \l__spath_miny_dim
408 \dim_zero:N \l__spath_maxx_dim
409 \dim_zero:N \l__spath_maxy_dim
410
411 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
412 \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
413 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
414 \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
415 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
416
417 \tl_clear:N \l__spath_ip_tl
418 \tl_put_right:Nx \l__spath_ip_tl
419 {
420   {\dim_use:N \l__spath_tmpa_dim}
421   {\dim_use:N \l__spath_tmpb_dim}
422 }
423
424 \dim_set_eq:NN \l__spath_minx_dim \l__spath_tmpa_dim
425 \dim_set_eq:NN \l__spath_miny_dim \l__spath_tmpb_dim
426 \dim_set_eq:NN \l__spath_maxx_dim \l__spath_tmpa_dim
427 \dim_set_eq:NN \l__spath_maxy_dim \l__spath_tmpb_dim
428
429 \tl_put_left:Nx \l__spath_rp_tl
430 {
431   {\dim_use:N \l__spath_tmpa_dim}
432   {\dim_use:N \l__spath_tmpb_dim}
433 }
434
435 \tl_set:Nx \l__spath_ia_tl {\tl_head:N \l__spath_tmpa_tl}
436
437 \bool_until_do:nn {
438   \tl_if_empty_p:N \l__spath_tmpa_tl
439 }
440 {
441   \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
442   \tl_set:Nn \l__spath_tmpd_tl {}
443   \tl_set_eq:NN \l__spath_fa_tl \l__spath_tmpc_tl
444   \int_incr:N \l__spath_l_int
445

```

```

446 \tl_case:NnF \l__spath_tmpc_tl
447 {
448   \g__spath_moveto_tl {
449     \tl_set_eq:NN \l__spath_tmpd_tl \g__spath_moveto_tl
450     \int_incr:N \l__spath_nc_int
451   }
452   \g__spath_lineto_tl {
453     \tl_set_eq:NN \l__spath_tmpd_tl \g__spath_lineto_tl
454     \int_incr:N \l__spath_rl_int
455   }
456   \g__spath_curveto_tl {
457     \tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curvetoa_tl
458     \int_incr:N \l__spath_rl_int
459   }
460   \g__spath_curvetoa_tl {
461     \tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curveto_tl
462   }
463   \g__spath_curvetob_tl {
464     \tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curvetob_tl
465   }
466 }
467 {
468   \tl_show:N \l__spath_tmpc_tl
469 }
470 \tl_put_left:Nv \l__spath_rp_tl \l__spath_tmpd_tl
471 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
472
473 \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
474 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
475 \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
476 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
477
478 \dim_set:Nn \l__spath_minx_dim {\dim_min:nn { \l__spath_minx_dim} {\l__spath_tmpa_dim}}
479 \dim_set:Nn \l__spath_miny_dim {\dim_min:nn { \l__spath_miny_dim} {\l__spath_tmpb_dim}}
480 \dim_set:Nn \l__spath_maxx_dim {\dim_max:nn { \l__spath_maxx_dim} {\l__spath_tmpa_dim}}
481 \dim_set:Nn \l__spath_maxy_dim {\dim_max:nn { \l__spath_maxy_dim} {\l__spath_tmpb_dim}}
482
483 \tl_put_left:Nx \l__spath_rp_tl
484 {
485   {\dim_use:N \l__spath_tmpa_dim}
486   {\dim_use:N \l__spath_tmpb_dim}
487 }
488
489 \tl_set:Nx \l__spath_fp_tl
490 {
491   {\dim_use:N \l__spath_tmpa_dim}
492   {\dim_use:N \l__spath_tmpb_dim}
493 }
494
495 }

```

```

496
497 \tl_put_left:NV \l__spath_rp_tl \g__spath_moveto_tl
498
499 \spath_put:nnV {#1} {reverse path} \l__spath_rp_tl
500 \spath_put:nnV {#1} {length} \l__spath_l_int
501 \spath_put:nnV {#1} {real length} \l__spath_rl_int
502 \spath_put:nnV {#1} {number of components} \l__spath_nc_int
503 \spath_put:nnV {#1} {initial point} \l__spath_ip_tl
504 \spath_put:nnV {#1} {final point} \l__spath_fp_tl
505 \spath_put:nnV {#1} {initial action} \l__spath_ia_tl
506 \spath_put:nnV {#1} {final action} \l__spath_fa_tl
507
508 \tl_clear:N \l__spath_tmpb_tl
509 \tl_put_right:Nx \l__spath_tmpb_tl
510 {
511   {\dim_use:N \l__spath_minx_dim}
512   {\dim_use:N \l__spath_miny_dim}
513 }
514 \spath_put:nnV {#1} {min bb} \l__spath_tmpb_tl
515
516 \tl_clear:N \l__spath_tmpb_tl
517 \tl_put_right:Nx \l__spath_tmpb_tl
518 {
519   {\dim_use:N \l__spath_maxx_dim}
520   {\dim_use:N \l__spath_maxy_dim}
521 }
522 \spath_put:nnV {#1} {max bb} \l__spath_tmpb_tl
523
524 }

```

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

## 2.4 Path Manipulation

`\spath_translate:nnn` Translates a path.

```

525 \cs_new_nopar:Npn \spath_translate:nnn #1#2#3
526 {
527   \tl_map_inline:Nn \g__spath_moveable_attributes
528   {
529     \spath_if_in:nnT {#1} {##1}
530     {
531       \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
532
533       \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl + #2}
534       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
535       \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl + #3}
536       \tl_clear:N \l__spath_tmpb_tl
537       \tl_put_right:Nx \l__spath_tmpb_tl
538       {
539         {\dim_use:N \l__spath_tmpa_dim}

```

```

540     {\dim_use:N \l__spath_tmpb_dim}
541   }
542   \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
543 }
544 }
545 \tl_map_inline:Nn \g__spath_path_attributes
546 {
547   \spath_if_in:nnT {#1} {##1}
548   {
549     \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
550     \tl_clear:N \l__spath_tmpb_tl
551     \bool_until_do:nn {
552       \tl_if_empty_p:N \l__spath_tmpa_tl
553     }
554     {
555       \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
556       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
557
558       \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl + #2}
559       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
560
561       \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl + #3}
562       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
563
564       \tl_put_right:Nx \l__spath_tmpb_tl
565       {
566         {\dim_use:N \l__spath_tmpa_dim}
567         {\dim_use:N \l__spath_tmpb_dim}
568       }
569     }
570     \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
571   }
572 }
573 }
574
575 \cs_generate_variant:Nn \spath_translate:nnn {nxx}

```

This variant allows for passing the coordinates as a single braced group as it strips off the outer braces of the second argument.

```

576 \cs_new_nopar:Npn \spath_translate:nn #1#2
577 {
578   \spath_translate:nnn {#1} #2
579 }
580
581 \cs_generate_variant:Nn \spath_translate:nn {nV}

```

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

`\spath_scale:nnn` Scale a path.

```

582 \cs_new_nopar:Npn \spath_scale:nnn #1#2#3

```



```

583 {
584   \tl_map_inline:Nn \g__spath_moveable_attributes
585   {
586     \spath_if_in:nnT {#1} {##1}
587     {
588       \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
589
590       \fp_set:Nn \l__spath_tmpa_fp {\tl_head:N \l__spath_tmpa_tl * #2}
591       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
592       \fp_set:Nn \l__spath_tmpb_fp {\tl_head:N \l__spath_tmpa_tl * #3}
593       \tl_clear:N \l__spath_tmpb_tl
594       \tl_put_right:Nx \l__spath_tmpb_tl
595       {
596         {\fp_to_dim:N \l__spath_tmpa_fp}
597         {\fp_to_dim:N \l__spath_tmpb_fp}
598       }
599       \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
600     }
601   }
602   \tl_map_inline:Nn \g__spath_path_attributes
603   {
604     \spath_if_in:nnT {#1} {##1}
605     {
606       \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
607       \tl_clear:N \l__spath_tmpb_tl
608       \bool_until_do:nn {
609         \tl_if_empty_p:N \l__spath_tmpa_tl
610       }
611       {
612         \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
613         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
614
615         \fp_set:Nn \l__spath_tmpa_fp {\tl_head:N \l__spath_tmpa_tl * #2}
616         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
617
618         \fp_set:Nn \l__spath_tmpb_fp {\tl_head:N \l__spath_tmpa_tl * #3}
619         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
620
621         \tl_put_right:Nx \l__spath_tmpb_tl
622         {
623           {\fp_to_dim:N \l__spath_tmpa_fp}
624           {\fp_to_dim:N \l__spath_tmpb_fp}
625         }
626       }
627       \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
628     }
629   }
630 }
631 \cs_generate_variant:Nn \spath_scale:nnn {nxx}

```

This variant allows for passing the coordinates as a single braced group as it strips off the outer braces of the second argument.

```

632 \cs_new_nopar:Npn \spath_scale:nn #1#2
633 {
634   \spath_scale:nnn {#1} #2
635 }
636
637 \cs_generate_variant:Nn \spath_scale:nn {nV}

```

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

`\spath_transform:nnnnnnn` Applies an affine (matrix and vector) transformation to path. The matrix is specified in rows first.

```

638 \cs_new_nopar:Npn \spath_transform:nnnnnnn #1#2#3#4#5#6#7
639 {
640   \tl_map_inline:Nn \g__spath_moveable_attributes
641   {
642     \spath_if_in:nnT {#1} {##1}
643     {
644       \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
645       \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpa_tl}
646       \tl_set:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpb_tl}
647       \tl_set:Nx \l__spath_tmpa_tl {\tl_head:N \l__spath_tmpa_tl}
648       \fp_set:Nn \l__spath_tmpa_fp {\l__spath_tmpa_tl * #2 + \l__spath_tmpb_tl * #3 + #6}
649       \fp_set:Nn \l__spath_tmpb_fp {\l__spath_tmpa_tl * #4 + \l__spath_tmpb_tl * #5 + #7}
650       \tl_clear:N \l__spath_tmpb_tl
651       \tl_put_right:Nx \l__spath_tmpb_tl
652       {
653         {\fp_to_dim:N \l__spath_tmpa_fp}
654         {\fp_to_dim:N \l__spath_tmpb_fp}
655       }
656       \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
657     }
658   }
659   \tl_map_inline:Nn \g__spath_path_attributes
660   {
661     \spath_if_in:nnT {#1} {##1}
662     {
663       \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
664       \tl_clear:N \l__spath_tmpb_tl
665       \bool_until_do:nn {
666         \tl_if_empty_p:N \l__spath_tmpa_tl
667       }
668       {
669         \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
670         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
671         \tl_set:Nx \l__tmpa_tl {\tl_head:N \l__spath_tmpa_tl}
672         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
673         \tl_set:Nx \l__tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
674         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

```

```

675
676     \fp_set:Nn \l__spath_tmpa_fp {\l_tmpa_tl * #2 + \l_tmpb_tl * #3 + #6}
677     \fp_set:Nn \l__spath_tmpb_fp {\l_tmpa_tl * #4 + \l_tmpb_tl * #5 + #7}
678     \tl_put_right:Nx \l__spath_tmpb_tl
679     {
680       {\fp_to_dim:N \l__spath_tmpa_fp}
681       {\fp_to_dim:N \l__spath_tmpb_fp}
682     }
683   }
684   \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
685 }
686 }
687 }
688
689 \cs_generate_variant:Nn \spath_transform:nnnnnnn {nxxxxxx}

```

This variant allows for passing the coordinates as a single braced group as it strips off the outer braces of the second argument.

```

690 \cs_new_nopar:Npn \spath_transform:nn #1#2
691 {
692   \spath_transform:nnnnnnn {#1} #2
693 }
694
695 \cs_generate_variant:Nn \spath_transform:nn {nV}

```

*(End definition for \spath\_transform:nnnnnnn. This function is documented on page ??.)*

**\spath\_reverse:n** This reverses a path. As a lot of the data is invariant under reversing, there isn't a lot to do.

```

696 \cs_new_nopar:Npn \spath_reverse:n #1
697 {
698   \spath_if_in:nnF {#1} {reverse path} {
699     \use:c {spath_generate_reverse path:n} {#1}
700   }
701   \spath_swap:nnn {#1} {path} {reverse path}
702   \spath_swap:nnn {#1} {initial point} {final point}
703   \spath_swap:nnn {#1} {initial action} {final action}
704 }

```

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

**\spath\_swap:nnn** Swaps two entries, being careful to ensure that their existence (or otherwise) is preserved.

```

705 \cs_new_nopar:Npn \spath_swap:nnn #1#2#3
706 {
707   \__spath_get:nnNF {#1} {#2} \l__spath_tmpa_tl {\tl_clear:N \l__spath_tmpa_tl}
708   \__spath_get:nnNF {#1} {#3} \l__spath_tmpb_tl {\tl_clear:N \l__spath_tmpb_tl}
709   \tl_if_empty:NTF \l__spath_tmpb_tl
710   {\spath_remove:nn {#1} {#2}}
711   {\spath_put:nnV {#1} {#2} \l__spath_tmpb_tl}
712   \tl_if_empty:NTF \l__spath_tmpa_tl

```

```

713   {\spath_remove:nn {#1} {#3}}
714   {\spath_put:nnV {#1} {#3} \l__spath_tmpa_tl}
715 }

```

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

`\spath_weld:nn` This welds one path to another, moving the second so that its initial point coincides with the first's final point. It is called a *weld* because the initial move of the second path is removed. The first path is updated, the second is not modified.

```

716 \cs_new_nopar:Npn \spath_weld:nn #1#2
717 {
718   \spath_clone:nn {#2} {tmp_path}
719   \spath_get:nnN {#1} {final point} \l__spath_tmpa_tl
720
721   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
722   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
723   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
724
725   \spath_get:nnN {#2} {initial point} \l__spath_tmpa_tl
726
727   \dim_sub:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
728   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
729   \dim_sub:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
730
731   \spath_translate:nxx {tmp_path} {\dim_use:N \l__spath_tmpa_dim} {\dim_use:N \l__spath_tmpb_
732
733   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
734   \__spath_get:nnN {tmp_path} {path} \l__spath_tmpb_tl
735   \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
736   \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
737   \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
738   \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
739
740   \spath_put:nnV {#1} {path} \l__spath_tmpa_tl
741
742   \__spath_get:nnNTF {tmp_path} {final point} \l__spath_tmpa_tl
743   {
744     \spath_put:nnV {#1} {final point} \l__spath_tmpa_tl
745   }
746   {
747     \spath_remove:nn {#1} {final point}
748   }
749
750   \__spath_get:nnNTF {tmp_path} {final action} \l__spath_tmpa_tl
751   {
752     \spath_put:nnV {#1} {final action} \l__spath_tmpa_tl
753   }
754   {
755     \spath_remove:nn {#1} {final action}
756   }

```

```

757
758 \l__spath_get:nnNT {tmp_path} {min bb} \l__spath_tmpa_tl
759 {
760 \l__spath_get:nnNT {#1} {min bb} \l__spath_tmpb_tl
761 {
762 \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
763 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
764 \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
765
766 \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\l__spath_tmpa_dim} {\tl_head:N
767 \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
768 \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\l__spath_tmpb_dim} {\tl_head:N \l__spath_t
769
770 \tl_clear:N \l__spath_tmpb_tl
771 \tl_put_right:Nx \l__spath_tmpb_tl
772 {
773 {\dim_use:N \l__spath_tmpa_dim}
774 {\dim_use:N \l__spath_tmpb_dim}
775 }
776 \spath_put:nnV {#1} {min bb} \l__spath_tmpb_tl
777 }
778 }
779
780 \l__spath_get:nnNT {tmp_path} {max bb} \l__spath_tmpa_tl
781 {
782 \l__spath_get:nnNT {#1} {max bb} \l__spath_tmpb_tl
783 {
784 \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
785 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
786 \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
787
788 \dim_set:Nn \l__spath_tmpa_dim {\dim_max:nn {\l__spath_tmpa_dim} {\tl_head:N
789 \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
790 \dim_set:Nn \l__spath_tmpb_dim {\dim_max:nn {\l__spath_tmpb_dim} {\tl_head:N \l__spath_t
791
792 \tl_clear:N \l__spath_tmpb_tl
793 \tl_put_right:Nx \l__spath_tmpb_tl
794 {
795 {\dim_use:N \l__spath_tmpa_dim}
796 {\dim_use:N \l__spath_tmpb_dim}
797 }
798 \spath_put:nnV {#1} {max bb} \l__spath_tmpb_tl
799 }
800 }
801
802 \l__spath_get:nnNT {tmp_path} {reverse path} \l__spath_tmpa_tl
803 {
804 \l__spath_get:nnNT {#1} {reverse path} \l__spath_tmpb_tl
805 {
806 \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}

```

```

807     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
808     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
809     \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
810
811     \spath_put:nnV {#1} {reverse path} \l__spath_tmpa_tl
812   }
813 }
814
815 \__spath_get:nnNT {tmp_path} {length} \l__spath_tmpa_tl
816 {
817   \__spath_get:nnNT {#1} {length} \l__spath_tmpb_tl
818   {
819     \int_set:Nn \l__spath_tmpa_int {\l__spath_tmpa_tl + \l__spath_tmpb_tl - 1}
820     \spath_put:nnV {#1} {length} \l__spath_tmpa_int
821   }
822 }
823
824 \__spath_get:nnNT {tmp_path} {real length} \l__spath_tmpa_tl
825 {
826   \__spath_get:nnNT {#1} {real length} \l__spath_tmpb_tl
827   {
828     \int_set:Nn \l__spath_tmpa_int {\l__spath_tmpa_tl + \l__spath_tmpb_tl}
829     \spath_put:nnV {#1} {real length} \l__spath_tmpa_int
830   }
831 }
832
833 \__spath_get:nnNT {tmp_path} {number of components} \l__spath_tmpa_tl
834 {
835   \__spath_get:nnNT {#1} {number of components} \l__spath_tmpb_tl
836   {
837     \int_set:Nn \l__spath_tmpa_int {\l__spath_tmpa_tl + \l__spath_tmpb_tl - 1}
838     \spath_put:nnV {#1} {number of components} \l__spath_tmpa_int
839   }
840 }
841
842 }

```

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

`\spath_prepend_no_move:nn` Prepend the path from the second spath to the first, removing the adjoining move.

```

843 \cs_new_nopar:Npn \spath_prepend_no_move:nn #1#2
844 {
845   \spath_if_exist:nT {#2}
846   {
847     \__spath_get:nnN {#2} {path} \l__spath_tmpa_tl
848     \__spath_get:nnN {#1} {path} \l__spath_tmpb_tl
849     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
850     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
851     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
852     \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl

```

```

853 \spath_put:nnV {#1} {path} \l__spath_tmpa_tl
854
855 \spath_if_in:nnTF {#2} {initial point}
856 {
857   \__spath_get:nnN {#2} {initial point} \l__spath_tmpa_tl
858   \spath_put:nnV {#1} {initial point} \l__spath_tmpa_tl
859 }
860 {
861   \spath_remove:nn {#1} {initial point}
862 }
863
864 \spath_if_in:nnTF {#2} {initial action}
865 {
866   \__spath_get:nnN {#2} {initial action} \l__spath_tmpa_tl
867   \spath_put:nnV {#1} {initial action} \l__spath_tmpa_tl
868 }
869 {
870   \spath_remove:nn {#1} {initial action}
871 }
872
873 \bool_if:nTF
874 {
875   \spath_if_in_p:nn {#1} {length}
876   &&
877   \spath_if_in_p:nn {#2} {length}
878 }
879 {
880   \__spath_get:nnN {#1} {length} \l__spath_tmpa_tl
881   \__spath_get:nnN {#2} {length} \l__spath_tmpb_tl
882   \spath_put:nx {#1} {length} {\int_eval:n {\l__spath_tmpa_tl +
883     \l__spath_tmpb_tl - 1}}
884 }
885 {
886   \spath_remove:nn {#1} {length}
887 }
888 \bool_if:nTF
889 {
890   \spath_if_in_p:nn {#1} {real length}
891   &&
892   \spath_if_in_p:nn {#2} {real length}
893 }
894 {
895   \__spath_get:nnN {#1} {real length} \l__spath_tmpa_tl
896   \__spath_get:nnN {#2} {real length} \l__spath_tmpb_tl
897   \spath_put:nx {#1} {real length} {\int_eval:n {\l__spath_tmpa_tl +
898     \l__spath_tmpb_tl }}
899 }
900 {
901   \spath_remove:nn {#1} {real length}
902 }

```

```

903 \bool_if:nTF
904 {
905   \spath_if_in_p:nn {#1} {number of components}
906   &&
907   \spath_if_in_p:nn {#2} {number of components}
908 }
909 {
910   \__spath_get:nnN {#1} {number of components} \l__spath_tmpa_tl
911   \__spath_get:nnN {#2} {number of components} \l__spath_tmpb_tl
912   \spath_put:nxx {#1} {number of components} {\int_eval:n {\l__spath_tmpa_tl +
913     \l__spath_tmpb_tl - 1}}
914 }
915 {
916   \spath_remove:nn {#1} {number of components}
917 }
918 \bool_if:nTF
919 {
920   \spath_if_in_p:nn {#1} {min bb}
921   &&
922   \spath_if_in_p:nn {#2} {min bb}
923 }
924 {
925   \__spath_get:nnN {#1} {min bb} \l__spath_tmpa_tl
926   \__spath_get:nnN {#2} {min bb} \l__spath_tmpb_tl
927   \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_item:Nn
928     \l__spath_tmpa_tl {1}} {\tl_item:Nn
929     \l__spath_tmpb_tl {1}}}
930   \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_item:Nn
931     \l__spath_tmpa_tl {2}} {\tl_item:Nn
932     \l__spath_tmpb_tl {2}}}
933   \spath_put:nxx {#1} {min bb} {
934     {\dim_use:N \l__spath_tmpa_dim}
935     {\dim_use:N \l__spath_tmpb_dim}
936   }
937 }
938 {
939   \spath_remove:nn {#1} {min bb}
940 }
941 \bool_if:nTF
942 {
943   \spath_if_in_p:nn {#1} {max bb}
944   &&
945   \spath_if_in_p:nn {#2} {max bb}
946 }
947 {
948   \__spath_get:nnN {#1} {max bb} \l__spath_tmpa_tl
949   \__spath_get:nnN {#2} {max bb} \l__spath_tmpb_tl
950   \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_item:Nn
951     \l__spath_tmpa_tl {1}} {\tl_item:Nn
952     \l__spath_tmpb_tl {1}}}

```



```

953     \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_item:Nn
954       \l__spath_tmpa_tl {2}} {\tl_item:Nn
955         \l__spath_tmpb_tl {2}}}}
956     \spath_put:nnx {#1} {max bb} {
957       {\dim_use:N \l__spath_tmpa_dim}
958       {\dim_use:N \l__spath_tmpb_dim}
959     }
960   }
961   {
962     \spath_remove:nn {#1} {max bb}
963   }
964   \bool_if:nTF
965   {
966     \spath_if_in_p:nn {#1} {reverse path}
967     &&
968     \spath_if_in_p:nn {#2} {reverse path}
969   }
970   {
971     \__spath_get:nnN {#1} {reverse path} \l__spath_tmpa_tl
972     \__spath_get:nnN {#2} {reverse path} \l__spath_tmpb_tl
973     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
974     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
975     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
976     \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
977     \spath_put:nnV {#1} {reverse path} \l__spath_tmpb_tl
978   }
979   {
980     \spath_remove:nn {#1} {reverse path}
981   }
982 }
983 }
984 }

```

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

`\spath_append_no_move:nn` Append the path from the second `spath` to the first, removing the adjoining move.

```

985 \cs_new_nopar:Npn \spath_append_no_move:nn #1#2
986 {
987   \spath_if_exist:nT {#2}
988   {
989     \spath_get:nnN {#1} {path} \l__spath_tmpa_tl
990     \spath_get:nnN {#2} {path} \l__spath_tmpb_tl
991     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
992     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
993     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
994     \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
995     \spath_put:nnV {#1} {path} \l__spath_tmpa_tl
996     \spath_if_in:nnTF {#2} {final point}
997     {
998       \__spath_get:nnN {#2} {final point} \l__spath_tmpa_tl

```

```

999     \spath_put:nnV {#1} {final point} \l__spath_tmpa_tl
1000   }
1001   {
1002     \spath_remove:nn {#1} {final point}
1003   }
1004   \spath_if_in:nnTF {#2} {final action}
1005   {
1006     \__spath_get:nnN {#2} {final action} \l__spath_tmpa_tl
1007     \spath_put:nnV {#1} {final action} \l__spath_tmpa_tl
1008   }
1009   {
1010     \spath_remove:nn {#1} {final action}
1011   }
1012   \bool_if:nTF
1013   {
1014     \spath_if_in_p:nn {#1} {length}
1015     &&
1016     \spath_if_in_p:nn {#2} {length}
1017   }
1018   {
1019     \__spath_get:nnN {#1} {length} \l__spath_tmpa_tl
1020     \__spath_get:nnN {#2} {length} \l__spath_tmpb_tl
1021     \spath_put:nxx {#1} {length} {\int_eval:n {\l__spath_tmpa_tl +
1022       \l__spath_tmpb_tl - 1}}
1023   }
1024   {
1025     \spath_remove:nn {#1} {length}
1026   }
1027   \bool_if:nTF
1028   {
1029     \spath_if_in_p:nn {#1} {real length}
1030     &&
1031     \spath_if_in_p:nn {#2} {real length}
1032   }
1033   {
1034     \__spath_get:nnN {#1} {real length} \l__spath_tmpa_tl
1035     \__spath_get:nnN {#2} {real length} \l__spath_tmpb_tl
1036     \spath_put:nxx {#1} {real length} {\int_eval:n {\l__spath_tmpa_tl +
1037       \l__spath_tmpb_tl }}
1038   }
1039   {
1040     \spath_remove:nn {#1} {real length}
1041   }
1042   \bool_if:nTF
1043   {
1044     \spath_if_in_p:nn {#1} {number of components}
1045     &&
1046     \spath_if_in_p:nn {#2} {number of components}
1047   }
1048   {

```

```

1049     \_spath_get:nnN {#1} {number of components} \l__spath_tmpa_tl
1050     \_spath_get:nnN {#2} {number of components} \l__spath_tmpb_tl
1051     \spath_put:nx {#1} {number of components} {\int_eval:n {\l__spath_tmpa_tl +
1052         \l__spath_tmpb_tl - 1}}
1053 }
1054 {
1055     \spath_remove:nn {#1} {number of components}
1056 }
1057 \bool_if:nTF
1058 {
1059     \spath_if_in_p:nn {#1} {min bb}
1060     &&
1061     \spath_if_in_p:nn {#2} {min bb}
1062 }
1063 {
1064     \_spath_get:nnN {#1} {min bb} \l__spath_tmpa_tl
1065     \_spath_get:nnN {#2} {min bb} \l__spath_tmpb_tl
1066     \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_item:Nn
1067         \l__spath_tmpa_tl {1}} {\tl_item:Nn
1068         \l__spath_tmpb_tl {1}}}
1069     \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_item:Nn
1070         \l__spath_tmpa_tl {2}} {\tl_item:Nn
1071         \l__spath_tmpb_tl {2}}}
1072     \spath_put:nx {#1} {min bb} {
1073         {\dim_use:N \l__spath_tmpa_dim}
1074         {\dim_use:N \l__spath_tmpb_dim}
1075     }
1076 }
1077 {
1078     \spath_remove:nn {#1} {min bb}
1079 }
1080 \bool_if:nTF
1081 {
1082     \spath_if_in_p:nn {#1} {max bb}
1083     &&
1084     \spath_if_in_p:nn {#2} {max bb}
1085 }
1086 {
1087     \_spath_get:nnN {#1} {max bb} \l__spath_tmpa_tl
1088     \_spath_get:nnN {#2} {max bb} \l__spath_tmpb_tl
1089     \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_item:Nn
1090         \l__spath_tmpa_tl {1}} {\tl_item:Nn
1091         \l__spath_tmpb_tl {1}}}
1092     \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_item:Nn
1093         \l__spath_tmpa_tl {2}} {\tl_item:Nn
1094         \l__spath_tmpb_tl {2}}}
1095     \spath_put:nx {#1} {max bb} {
1096         {\dim_use:N \l__spath_tmpa_dim}
1097         {\dim_use:N \l__spath_tmpb_dim}
1098     }

```

```

1099 }
1100 {
1101   \spath_remove:nn {#1} {max bb}
1102 }
1103 \bool_if:nTF
1104 {
1105   \spath_if_in_p:nn {#1} {reverse path}
1106   &&
1107   \spath_if_in_p:nn {#2} {reverse path}
1108 }
1109 {
1110   \__spath_get:nnN {#2} {reverse path} \l__spath_tmpa_tl
1111   \__spath_get:nnN {#1} {reverse path} \l__spath_tmpb_tl
1112   \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
1113   \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
1114   \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
1115   \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
1116   \spath_put:nnV {#1} {reverse path} \l__spath_tmpb_tl
1117 }
1118 {
1119   \spath_remove:nn {#1} {reverse path}
1120 }
1121 }
1122 }

```

*(End definition for \spath\_append\_no\_move:nn. This function is documented on page ??.)*

`\spath_bake_round:n` Ought to clear the reverse path, if set.

```

1123 \cs_new_nopar:Npn \spath_bake_round:n #1
1124 {
1125   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
1126   \pgf@@processround\l__spath_tmpa_tl\l__spath_tmpb_tl
1127   \spath_put:nnV {#1} {path} \l__spath_tmpb_tl
1128 }

```

*(End definition for \spath\_bake\_round:n. This function is documented on page ??.)*

`\spath_close_path:n` Appends a close path to the end of the path, and to the end of the reverse path. For now, the point is the initial or final point (respectively). To be future proof, it ought to be the point of the adjacent move to.

```

1129 \cs_new_nopar:Npn \spath_close_path:n #1
1130 {
1131   \spath_get:nnN {#1} {initial point} \l__spath_tmpb_tl
1132   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
1133   \tl_put_right:NV \l__spath_tmpa_tl \g__spath_closepath_tl
1134   \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
1135   \spath_put:nnV {#1} {path} \l__spath_tmpa_tl
1136   \spath_if_in:nnT {#1} {reverse path}
1137   {
1138     \spath_get:nnN {#1} {final point} \l__spath_tmpb_tl

```

```

1139 \l__spath_get:nnN {#1} {reverse path} \l__spath_tmpa_tl
1140 \tl_put_right:NV \l__spath_tmpa_tl \g__spath_closepath_tl
1141 \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
1142 \spath_put:nnV {#1} {reverse path} \l__spath_tmpa_tl
1143 }
1144 }

```

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

## 2.5 Iteration Functions

`\spath_map_component:Nn` This iterates through the components of a path, applying the inline function to each.

```

1145 \cs_new_nopar:Npn \spath_map_component:Nn #1#2
1146 {
1147   \int_gincr:N \g__prg_map_int
1148   \cs_gset:cpn { __prg_map_ \int_use:N \g__prg_map_int :w } ##1 {#2}
1149   \tl_set:NV \l__spath_tmpa_tl #1
1150   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1151   \tl_put_right:NV \l__spath_tmpa_tl \g__spath_moveto_tl
1152   \tl_set_eq:NN \l__spath_tmpb_tl \g__spath_moveto_tl
1153   \bool_until_do:nn {
1154     \tl_if_empty_p:N \l__spath_tmpa_tl
1155   }
1156   {
1157     \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
1158     \tl_if_eq:NNT \l__spath_tmpc_tl \g__spath_moveto_tl
1159     {
1160       \exp_args:NnV \use:c { __prg_map_ \int_use:N \g__prg_map_int :w } \l__spath_tmpb_tl
1161     }
1162     \tl_clear:N \l__spath_tmpb_tl
1163     \tl_if_single:NTF \l__spath_tmpc_tl
1164     {
1165       \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpc_tl
1166     }
1167     {
1168       \tl_put_right:Nx \l__spath_tmpb_tl {\l__spath_tmpc_tl}
1169     }
1170     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1171   }
1172 }

```

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

`\spath_map_segment_inline:Nn` This iterates through the segments of the path, applying the inline function to each.

```

1173 \cs_new_nopar:Npn \spath_map_segment_inline:Nn #1#2
1174 {
1175   \int_gincr:N \g__prg_map_int
1176   \cs_gset:cpn { __prg_map_ \int_use:N \g__prg_map_int :w } ##1 ##2 {#2}
1177   \spath_map_segment_function:Nc #1 { __prg_map_ \int_use:N \g__prg_map_int :w }
1178 }

```

(End definition for \spath\_map\_segment\_inline:Nn. This function is documented on page ??.)

\spath\_map\_segment\_inline:nn This iterates through the segments of the path of the `spath` object, applying the inline function to each.

```
1179 \cs_new_nopar:Npn \spath_map_segment_inline:nn #1#2
1180 {
1181   \int_gincr:N \g__prg_map_int
1182   \cs_gset:cpn { __prg_map_ \int_use:N \g__prg_map_int :w } ##1 ##2 {#2}
1183   \spath_get:nnN {#1} {path} \l__spath_tmpd_tl
1184   \spath_map_segment_function:Nc \l__spath_tmpd_tl { __prg_map_ \int_use:N \g__prg_map_int :w
1185 }
```

(End definition for \spath\_map\_segment\_inline:nn. This function is documented on page ??.)

\spath\_map\_segment\_function:nn This iterates through the segments of the path of the `spath` object, applying the specified function to each. The specified function should take two N type arguments. The first is a token representing the type of path segment, the second is the path segment itself.

```
1186 \cs_new_nopar:Npn \spath_map_segment_function:nn #1#2
1187 {
1188   \spath_get:nnN {#1} {path} \l__spath_tmpd_tl
1189   \spath_map_segment_function:NN \l__spath_tmpd_tl #2
1190 }

1191 \cs_new_nopar:Npn \spath_map_segment_function:NN #1#2
1192 {
1193   \tl_set_eq:NN \l__spath_tmpa_tl #1
1194   \tl_clear:N \l__spath_tmpb_tl
1195   \dim_zero:N \l__spath_tmpa_dim
1196   \dim_zero:N \l__spath_tmpb_dim
1197
1198   \bool_until_do:nn {
1199     \tl_if_empty_p:N \l__spath_tmpa_tl
1200   }
1201   {
1202     \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
1203     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1204     \tl_case:NnF \l__spath_tmpc_tl
1205     {
1206       \g__spath_lineto_tl
1207       {
1208         \tl_set_eq:NN \l__spath_tmpb_tl \g__spath_moveto_tl
1209         \tl_put_right:Nx \l__spath_tmpb_tl
1210         {
1211           {\dim_use:N \l__spath_tmpa_dim}
1212           {\dim_use:N \l__spath_tmpb_dim}
1213         }
1214         \tl_put_right:NV \l__spath_tmpb_tl \g__spath_lineto_tl
1215
1216         \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1217         \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}

```

```

1218     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1219
1220     \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1221     \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
1222     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1223
1224 }
1225
1226 \g__spath_curvetoa_tl
1227 {
1228     \tl_set_eq:NN \l__spath_tmpb_tl \g__spath_moveto_tl
1229     \tl_put_right:Nx \l__spath_tmpb_tl
1230     {
1231         {\dim_use:N \l__spath_tmpa_dim}
1232         {\dim_use:N \l__spath_tmpb_dim}
1233     }
1234     \tl_put_right:NV \l__spath_tmpb_tl \g__spath_curvetoa_tl
1235
1236     \prg_replicate:nn {2} {
1237         \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1238         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1239         \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1240         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1241         \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
1242         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1243     }
1244
1245     \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1246     \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
1247     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1248
1249     \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1250     \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
1251     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1252
1253 }
1254 }
1255 {
1256
1257     \tl_set_eq:NN \l__spath_tmpb_tl \l__spath_tmpc_tl
1258     \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1259     \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
1260     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1261
1262     \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1263     \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
1264     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1265
1266 }
1267

```

```

1268     #2 \l__spath_tmpc_tl \l__spath_tmpb_tl
1269     \tl_clear:N \l__spath_tmpb_tl
1270
1271   }
1272 }
1273 \cs_generate_variant:Nn \spath_map_segment_function:NN {Nc}

```

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

## 2.6 Public Commands

The next functions are more “public” than the previous lot. That said, they aren’t intended for direct use in a normal document.

Most are just wrappers around internal functions.

`\MakeSPath` Constructs an `spath` object out of the given name and path.

```

1274 \NewDocumentCommand \MakeSPath { m m }
1275 {
1276   \spath_clear_new:n {#1}
1277   \spath_put:nno {#1} {path} {#2}
1278 }

```

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

`\MakeSPathList` This constructs a list of `spath` objects from a single path by splitting it into components.

```

1279 \NewDocumentCommand \MakeSPathList { m m }
1280 {
1281   \tl_gclear_new:c {l__spath_list_#1}
1282   \int_zero:N \l__spath_tmpa_int
1283   \spath_map_component:Nn #2 {
1284     \spath_clear_new:n {#1 _ \int_use:N \l__spath_tmpa_int}
1285     \spath_put:nnn {#1 _ \int_use:N \l__spath_tmpa_int} {path} {##1}
1286     \tl_gput_right:cx {l__spath_list_#1} {{#1 _ \int_use:N \l__spath_tmpa_int}}
1287     \int_incr:N \l__spath_tmpa_int
1288   }
1289 }

```

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

`\CloneSPath`

```

1290 \NewDocumentCommand \CloneSPath { m m }
1291 {
1292   \spath_clone:nn {#1} {#2}
1293 }

```

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



### `\SPathInfo`

```
1294 \NewDocumentCommand \SPathInfo { m m }
1295 {
1296   \spath_get:nn {#1} {#2}
1297 }
```

*(End definition for \SPathInfo. This function is documented on page ??.)*

### `\SPathPrepare`

```
1298 \NewDocumentCommand \SPathPrepare { m }
1299 {
1300   \spath_generate_all:n {#1}
1301 }
```

*(End definition for \SPathPrepare. This function is documented on page ??.)*

### `\SPathListPrepare`

```
1302 \NewDocumentCommand \SPathListPrepare { m }
1303 {
1304   \tl_map_inline:cn {l__spath_list_#1}
1305   {
1306     \spath_generate_all:n {##1}
1307   }
1308 }
```

*(End definition for \SPathListPrepare. This function is documented on page ??.)*

### `\SPathInfoInto`

```
1309 \NewDocumentCommand \SPathInfoInto { m m m }
1310 {
1311   \tl_set:Nn \l_tmpa_tl #3
1312   \spath_get:nnV {#1} {#2} \l_tmpa_tl
1313 }
```

*(End definition for \SPathInfoInto. This function is documented on page ??.)*

### `\SPathShow`

```
1314 \NewDocumentCommand \SPathShow { m }
1315 {
1316   \spath_show:n {#1}
1317 }
```

*(End definition for \SPathShow. This function is documented on page ??.)*

### `\SPathTranslate`

```
1318 \NewDocumentCommand \SPathTranslate { m m m }
1319 {
1320   \spath_translate:nnn {#1} {#2} {#3}
1321 }
```

*(End definition for \SPathTranslate. This function is documented on page ??.)*

`\SPathTranslateInto` Clones the path before translating it.

```

1322 \NewDocumentCommand \SPathTranslateInto { m m m m }
1323 {
1324   \spath_clone:nn {#1} {#2}
1325   \spath_translate:nnn {#2} {#3} {#4}
1326 }

```

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

`\SPathScale`

```

1327 \NewDocumentCommand \SPathScale { m m m }
1328 {
1329   \spath_scale:nnn {#1} {#2} {#3}
1330 }

```

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

`\SPathScaleInto` Clones the path first.

```

1331 \NewDocumentCommand \SPathScaleInto { m m m m }
1332 {
1333   \spath_clone:nn {#1} {#2}
1334   \spath_scale:nnn {#2} {#3} {#4}
1335 }

```

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

`\SPathWeld`

```

1336 \NewDocumentCommand \SPathWeld { m m }
1337 {
1338   \spath_weld:nn {#1} {#2}
1339 }

```

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

`\SPathWeldInto`

```

1340 \NewDocumentCommand \SPathWeldInto { m m m }
1341 {
1342   \spath_clone:nn {#1} {#2}
1343   \spath_weld:nn {#2} {#3}
1344 }

```

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

Interfaces via TikZ keys.

```

1345 \tikzset{
1346   save~spath/.code={
1347     \tikz@addmode{
1348       \spath_get_current_path:n {#1}
1349     }
1350   },
1351   restore~spath/.code={
1352     \spath_set_current_path:n {#1}
1353   }
1354 }

```

## 2.7 Miscellaneous Commands

`\spath_split_curve:nnNN` Splits a Bezier cubic into pieces.

```

1355 \cs_new_nopar:Npn \spath_split_curve:nnNN #1#2#3#4
1356 {
1357   \group_begin:
1358   \tl_gclear:N \l__spath_smuggle_tl
1359   \tl_set_eq:NN \l__spath_tmpa_tl \g__spath_moveto_tl
1360   \tl_put_right:Nx \l__spath_tmpa_tl {
1361     {\tl_item:nn {#2} {3}}
1362     {\tl_item:nn {#2} {4}}
1363   }
1364   \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curvetoa_tl
1365   \tl_put_right:Nx \l__spath_tmpa_tl
1366   {
1367     {\fp_to_dim:n
1368       {
1369         (1 - #1) * \tl_item:nn {#2} {2} + (#1) * \tl_item:nn {#2} {5}
1370       }}
1371     {\fp_to_dim:n
1372       {
1373         (1 - #1) * \tl_item:nn {#2} {3} + (#1) * \tl_item:nn {#2} {6}
1374       }}
1375   }
1376   \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curvetob_tl
1377   \tl_put_right:Nx \l__spath_tmpa_tl
1378   {
1379     {\fp_to_dim:n
1380       {
1381         (1 - #1)^2 * \tl_item:nn {#2} {2} + 2 * (1 - #1) * (#1) * \tl_item:nn {#2} {5} + (#1)^2 *
1382       }}
1383     {\fp_to_dim:n
1384       {
1385         (1 - #1)^2 * \tl_item:nn {#2} {3} + 2 * (1 - #1) * (#1) * \tl_item:nn {#2} {6} + (#1)^2 *
1386       }}
1387   }
1388   \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curveto_tl
1389   \tl_put_right:Nx \l__spath_tmpa_tl
1390   {
1391     {\fp_to_dim:n
1392       {
1393         (1 - #1)^3 * \tl_item:nn {#2} {2} + 3 * (1 - #1)^2 * (#1) * \tl_item:nn {#2} {5} + 3 *
1394       }}
1395     {\fp_to_dim:n
1396       {
1397         (1 - #1)^3 * \tl_item:nn {#2} {3} + 3 * (1 - #1)^2 * (#1) * \tl_item:nn {#2} {6} + 3 *
1398       }}
1399   }
1400   \tl_gset_eq:NN \l__spath_smuggle_tl \l__spath_tmpa_tl
1401   \group_end:

```

```

1402 \tl_set_eq:NN #3 \l__spath_smuggle_tl
1403 \group_begin:
1404 \tl_set_eq:NN \l__spath_tmpa_tl \g__spath_moveto_tl
1405 \tl_put_right:Nx \l__spath_tmpa_tl
1406 {
1407   {\fp_to_dim:n
1408     {
1409       (1 - #1)^3 * \tl_item:nn {#2} {2} + 3 * (1 - #1)^2 * (#1) * \tl_item:nn {#2} {5} + 3 *
1410     }}
1411   {\fp_to_dim:n
1412     {
1413       (1 - #1)^3 * \tl_item:nn {#2} {3} + 3 * (1 - #1)^2 * (#1) * \tl_item:nn {#2} {6} + 3 *
1414     }}
1415   }
1416 \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curvetoa_tl
1417 \tl_put_right:Nx \l__spath_tmpa_tl
1418 {
1419   {\fp_to_dim:n
1420     {
1421       (1 - #1)^2 * \tl_item:nn {#2} {5} + 2 * (1 - #1) * (#1) * \tl_item:nn {#2} {8} + (#1)^2
1422     }}
1423   {\fp_to_dim:n
1424     {
1425       (1 - #1)^2 * \tl_item:nn {#2} {6} + 2 * (1 - #1) * (#1) * \tl_item:nn {#2} {9} + (#1)^2
1426     }}
1427   }
1428 \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curvetob_tl
1429 \tl_put_right:Nx \l__spath_tmpa_tl
1430 {
1431   {\fp_to_dim:n
1432     {
1433       (1 - #1) * \tl_item:nn {#2} {8} + (#1) * \tl_item:nn {#2} {11}
1434     }}
1435   {\fp_to_dim:n
1436     {
1437       (1 - #1) * \tl_item:nn {#2} {9} + (#1) * \tl_item:nn {#2} {12}
1438     }}
1439   }
1440 \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curveto_tl
1441 \tl_put_right:Nx \l__spath_tmpa_tl {
1442   {\tl_item:nn {#2} {11}}
1443   {\tl_item:nn {#2} {12}}
1444 }
1445 \tl_gset_eq:NN \l__spath_smuggle_tl \l__spath_tmpa_tl
1446 \group_end:
1447 \tl_set_eq:NN #4 \l__spath_smuggle_tl
1448 }
1449
1450 \cs_generate_variant:Nn \spath_split_curve:nnNN {nVNN, VVNN}

```

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

## 3 The Calligraphy Package

### 3.1 Initialisation

```
1451 \RequirePackage{spath3}
1452 \ExplSyntaxOn
1453
1454 \tl_new:N \l__cal_tmpa_tl
1455 \tl_new:N \l__cal_tmpb_tl
1456 \int_new:N \l__cal_tmpa_int
1457 \int_new:N \l__cal_tmpb_int
1458 \int_new:N \l__cal_path_component_int
1459 \int_new:N \l__cal_label_int
1460 \dim_new:N \l__cal_tmpa_dim
1461 \dim_new:N \l__cal_tmpb_dim
1462 \dim_new:N \l__cal_tmpe_dim
1463 \dim_new:N \l__cal_tmpe_dim
1464 \dim_new:N \l__cal_tmpe_dim
1465 \dim_new:N \l__cal_tmpe_dim
1466 \dim_new:N \l__cal_tmpe_dim
1467 \dim_new:N \l__cal_tmpe_dim
1468 \bool_new:N \l__cal_annotate_bool
1469 \bool_new:N \l__cal_taper_start_bool
1470 \bool_new:N \l__cal_taper_end_bool
1471 \bool_new:N \l__cal_taperable_bool
1472 \dim_new:N \l__cal_taper_width_dim
1473 \dim_new:N \l__cal_line_width_dim
1474
1475 \bool_set_true:N \l__cal_taper_start_bool
1476 \bool_set_true:N \l__cal_taper_end_bool
1477
1478 \cs_generate_variant:Nn \tl_put_right:Nn {Nv}
```

### 3.2 TikZ Keys

The public interface to this package is through TikZ keys and styles.

```
1479 \tikzset{
1480   define~pen/.code={
1481     \tikzset{pen-name=#1}
1482     \pgf@relevantforpicturesizefalse
1483     \tikz@addmode{
1484       \pgfsyssoftpath@getcurrentpath\l__cal_tmpa_tl
1485       \MakeSPathList{calligraphy pen \pgfkeysvalueof{/tikz/pen-name}}{\l__cal_tmpa_tl}
1486       \SPathListPrepare{calligraphy pen \pgfkeysvalueof{/tikz/pen-name}}
1487       \pgfusepath{discard}%
1488     }
1489   },
```

```

1490 define~pen/.default={default},
1491 use~pen/.code={
1492   \tikzset{pen~name=#1}
1493   \int_gzero:N \l__cal_path_component_int
1494   \cs_set_eq:NN \pgfpathmoveto \cal_moveto:n
1495   \tikz@addmode{
1496     \pgfsyssoftpath@getcurrentpath\l__cal_tmpa_tl
1497     \MakeSPathList{calligraphy path}{\l__cal_tmpa_tl}
1498     \SPathListPrepare{calligraphy path}
1499     \CalligraphyPathCreate{calligraphy path}{\pgfkeysvalueof{/tikz/pen~name}}
1500   }
1501 },
1502 use~pen/.default={default},
1503 pen~name/.initial={default},
1504 copperplate/.style={pen~name=copperplate},
1505 pen~colour/.initial={black},
1506 weight/.is~choice,
1507 weight/heavy/.style={
1508   line~width=\pgfkeysvalueof{/tikz/heavy~line~width},
1509   taper~width=\pgfkeysvalueof{/tikz/light~line~width},
1510 },
1511 weight/light/.style={
1512   line~width=\pgfkeysvalueof{/tikz/light~line~width},
1513   taper~width=0pt,
1514 },
1515 heavy/.style={
1516   weight=heavy
1517 },
1518 light/.style={
1519   weight=light
1520 },
1521 heavy~line~width/.initial=2pt,
1522 light~line~width/.initial=1pt,
1523 taper/.is~choice,
1524 taper/.default=both,
1525 taper/none/.style={
1526   taper~start=false,
1527   taper~end=false,
1528 },
1529 taper/both/.style={
1530   taper~start=true,
1531   taper~end=true,
1532 },
1533 taper/start/.style={
1534   taper~start=true,
1535   taper~end=false,
1536 },
1537 taper/end/.style={
1538   taper~start=false,
1539   taper~end=true,

```

```

1540 },
1541 taper~start/.code={
1542   \tl_if_eq:nnTF {#1} {true}
1543   {
1544     \bool_set_true:N \l__cal_taper_start_bool
1545   }
1546   {
1547     \bool_set_false:N \l__cal_taper_start_bool
1548   }
1549 },
1550 taper~start/.default={true},
1551 taper~end/.code={
1552   \tl_if_eq:nnTF {#1} {true}
1553   {
1554     \bool_set_true:N \l__cal_taper_end_bool
1555   }
1556   {
1557     \bool_set_false:N \l__cal_taper_end_bool
1558   }
1559 },
1560 taper~end/.default={true},
1561 taper~width/.code={\dim_set:Nn \l__cal_taper_width_dim {#1}},
1562 nib~style/.code~2~args={
1563   \tl_clear_new:c {l__cal_nib_style_#1}
1564   \tl_set:cn {l__cal_nib_style_#1} {#2}
1565 },
1566 stroke~style/.code~2~args={
1567   \tl_clear_new:c {l__cal_stroke_style_#1}
1568   \tl_set:cn {l__cal_stroke_style_#1} {#2}
1569 },
1570 this~stroke~style/.code={
1571   \tl_clear_new:c {l__cal_stroke_inline_style_ \int_use:N \l__cal_path_component_int}
1572   \tl_set:cn {l__cal_stroke_inline_style_ \int_use:N \l__cal_path_component_int} {#1}
1573 },
1574 annotate/.style={
1575   annotate~if,
1576   annotate~reset,
1577   annotation~style/.update~value={#1},
1578 },
1579 annotate~if/.default={true},
1580 annotate~if/.code={
1581   \tl_if_eq:nnTF {#1} {true}
1582   {
1583     \bool_set_true:N \l__cal_annotate_bool
1584   }
1585   {
1586     \bool_set_false:N \l__cal_annotate_bool
1587   }
1588 },
1589 annotate~reset/.code={

```

```

1590   \int_gzero:N \l__cal_label_int
1591 },
1592 annotation~style/.initial={draw,->},
1593 annotation~shift/.initial={(0,1ex)},
1594 every~annotation~node/.initial={anchor=south-west},
1595 annotation~node~style/.code~2~args={
1596   \tl_set:cn {l__cal_annotation_style_ #1 _tl}{#2}
1597 },
1598 tl~use:N/.code={
1599   \exp_args:Nv \pgfkeysalso #1
1600 },
1601 tl~use:c/.code={
1602   \tl_if_exist:cT {#1}
1603   {
1604     \exp_args:Nv \pgfkeysalso {#1}
1605   }
1606 },
1607 /handlers/.update~style/.code={
1608   \tl_if_eq:nnF {#1} {\pgfkeysnovalue}
1609   {
1610     \pgfkeys{\pgfkeyscurrentpath/.code=\pgfkeysalso{#1}}
1611   }
1612 },
1613 /handlers/.update~value/.code={
1614   \tl_if_eq:nnF {#1} {\pgfkeysnovalue}
1615   {
1616     \pgfkeyssetvalue{\pgfkeyscurrentpath}{#1}
1617   }
1618 }
1619 }

```

Some wrappers around the TikZ keys.

```

1620 \NewDocumentCommand \pen { 0{ } }
1621 {
1622   \path[define~ pen,every~ calligraphy~ pen/.try,#1]
1623 }
1624
1625 \NewDocumentCommand \definepen { 0{ } }
1626 {
1627   \tikz \path[define~ pen,every~ calligraphy~ pen/.try,#1]
1628 }
1629
1630 \NewDocumentCommand \calligraphy { 0{ } }
1631 {
1632   \path[use~ pen,every~ calligraphy/.try,#1]
1633 }

```



### 3.3 The Path Creation

`\CalligraphyPathCreate` This is the main command for creating the calligraphic paths.

```

1634 \NewDocumentCommand \CalligraphyPathCreate { m m }
1635 {
1636   \int_zero:N \l__cal_tmpa_int
1637   \tl_map_inline:cn {l__spath_list_#1}
1638   {
1639     \int_incr:N \l__cal_tmpa_int
1640     \int_zero:N \l__cal_tmpb_int
1641     \tl_map_inline:cn {l__spath_list_calligraphy pen #2}
1642     {
1643       \int_incr:N \l__cal_tmpb_int
1644       \group_begin:
1645       \pgfsys@beginscope
1646
1647       \cal_apply_style:c {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_int}
1648       \cal_apply_style:c {l__cal_stroke_inline_style_ \int_use:N \l__cal_tmpa_int}
1649       \cal_apply_style:c {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}
1650
1651       \spath_clone:nn {##1} {calligraphy temp path}
1652
1653       \__spath_get:nnN {####1} {initial point} \l__cal_tmpa_tl
1654       \spath_translate:nV {calligraphy temp path} \l__cal_tmpa_tl
1655
1656       \__spath_get:nnN {####1} {length} \l__cal_tmpa_tl
1657
1658       \int_compare:nTF {\l__cal_tmpa_tl = 1}
1659       {
1660         \cal_at_least_three:n {calligraphy temp path}
1661
1662         \spath_protocol_path:n {calligraphy temp path}
1663
1664         \__spath_get:nnN {calligraphy temp path} {path} \l__cal_tmpa_tl
1665
1666         \tikz@options
1667         \dim_set:Nn \l__cal_line_width_dim {\pgflinewidth}
1668         \cal_maybe_taper:N \l__cal_tmpa_tl
1669       }
1670     {
1671
1672     \spath_weld:nn {calligraphy temp path} {####1}
1673     \spath_reverse:n {##1}
1674     \spath_reverse:n {####1}
1675     \spath_weld:nn {calligraphy temp path} {##1}
1676     \spath_weld:nn {calligraphy temp path} {####1}
1677     \spath_reverse:n {##1}
1678     \spath_reverse:n {####1}
1679
1680     \tl_clear:N \l__cal_tmpa_tl

```

```

1681     \tl_set:Nn \l__cal_tmpa_tl {fill=\pgfkeysvalueof{/tikz/pen-colour},draw=none}
1682     \tl_if_exist:cT {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_int}
1683     {
1684         \tl_put_right:Nv \l__cal_tmpa_tl {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_int}
1685     }
1686     \tl_if_exist:cT {l__cal_stroke_inline_style_ \int_use:N \l__cal_tmpa_int}
1687     {
1688         \tl_put_right:Nn \l__cal_tmpa_tl {,}
1689         \tl_put_right:Nv \l__cal_tmpa_tl {l__cal_stroke_inline_style_ \int_use:N \l__cal_tm
1690     }
1691     \tl_if_exist:cT {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}
1692     {
1693         \tl_put_right:Nn \l__cal_tmpa_tl {,}
1694         \tl_put_right:Nv \l__cal_tmpa_tl {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}
1695     }
1696     \spath_tikz_path:Vn \l__cal_tmpa_tl {calligraphy temp path}
1697
1698 }
1699 \pgfsys@endscope
1700 \group_end:
1701 }
1702 \bool_if:NT \l__cal_annotate_bool
1703 {
1704     \spath_clone:nn {##1} {calligraphy temp path}
1705     \tl_set_eq:Nc \l_tmpa_tl {l__spath_list_calligraphy pen #2}
1706     \tl_reverse:N \l_tmpa_tl
1707     \tl_set:Nx \l_tmpa_tl {\tl_head:N \l_tmpa_tl}
1708     \spath_generate_finalpoint:V \l_tmpa_tl
1709     \spath_get:VnN \l_tmpa_tl {final point} \l_tmpa_tl
1710     \spath_translate:nV {calligraphy temp path} \l_tmpa_tl
1711     \tikz@scan@one@point\pgfutil@firstofone\pgfkeysvalueof{/tikz/annotation~shift}
1712     \spath_translate:nnn {calligraphy temp path} {\pgf@x} {\pgf@y}
1713
1714     \pgfkeysgetvalue{/tikz/annotation~style}{\l_tmpa_tl}
1715     \spath_tikz_path:Vn \l_tmpa_tl {calligraphy temp path}
1716     \spath_get:nnN {calligraphy temp path} {final point} \l_tmpa_tl
1717     \exp_last_unbraced:Nv \pgfqpoint \l_tmpa_tl
1718     \begin{scope}[reset~ cm]
1719     \node[every~annotation~node/.try,tl~use:c = {l__cal_annotate_style_ \int_use:N \l__ca
1720     \end{scope}
1721 }
1722 }
1723 }

```

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

`\cal_moveto:n` When creating the path, we need to keep track of the number of components so that we can apply styles accordingly.

```

1724 \cs_new_eq:NN \cal_orig_moveto:n \pgfpathmoveto
1725 \cs_new_nopar:Npn \cal_moveto:n #1

```

```

1726 {
1727   \int_gincr:N \l__cal_path_component_int
1728   \cal_orig_moveto:n {#1}
1729 }

```

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

\cal\_apply\_style:N Interface for applying \tikzset to a token list.

```

1730 \cs_new_nopar:Npn \cal_apply_style:N #1
1731 {
1732   \tl_if_exist:NT #1 {
1733     \exp_args:NV \tikzset #1
1734   }
1735 }
1736 \cs_generate_variant:Nn \cal_apply_style:N {c}

```

(End definition for \cal\_apply\_style:N. This function is documented on page ??.)

\cal\_at\_least\_three:n A tapered path has to have at least three components. This figures out if it is necessary and sets up the splitting.

```

1737 \cs_new_nopar:Npn \cal_at_least_three:n #1
1738 {
1739   \spath_get:nnN {#1} {real length} \l__cal_tmpa_tl
1740   \tl_clear:N \l__cal_tmpb_tl
1741   \int_compare:nTF {\l__cal_tmpa_tl = 1}
1742   {
1743     \spath_get:nnN {#1} {path} \l__cal_tmpa_tl
1744     \spath_map_segment_inline:Nn \l__cal_tmpa_tl
1745     {
1746       \tl_case:NnF ##1 {
1747         \g__spath_lineto_tl {
1748           \cal_split_line_in_three:NN \l__cal_tmpb_tl ##2
1749         }
1750         \g__spath_curvetoa_tl {
1751           \cal_split_curve_in_three:NN \l__cal_tmpb_tl ##2
1752         }
1753       }
1754     }
1755     \tl_put_right:NV \l__cal_tmpb_tl ##2
1756   }
1757   \spath_put:nnV {#1} {path} \l__cal_tmpb_tl
1758 }
1759 {
1760   \int_compare:nT {\l__cal_tmpa_tl = 2}
1761   {
1762     \spath_get:nnN {#1} {path} \l__cal_tmpa_tl
1763     \spath_map_segment_inline:Nn \l__cal_tmpa_tl
1764     {
1765       \tl_case:NnF ##1 {

```

```

1767     \g__spath_lineto_tl {
1768       \cal_split_line_in_two:NN \l__cal_tmpb_tl ##2
1769     }
1770     \g__spath_curvetoa_tl {
1771       \cal_split_curve_in_two:NN \l__cal_tmpb_tl ##2
1772     }
1773   }
1774   {
1775     \tl_put_right:NV \l__cal_tmpb_tl ##2
1776   }
1777 }
1778 \spath_put:nnV {#1} {path} \l__cal_tmpb_tl
1779 }
1780 }
1781 }

```

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

`\cal_split_line_in_two:NN` Splits a line in two, adding the splits to the first token list.

```

1782 \cs_new_nopar:Npn \cal_split_line_in_two:NN #1#2
1783 {
1784   \tl_set_eq:NN \l__cal_tmpc_tl #2
1785
1786   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1787
1788   \dim_set:Nn \l__cal_tmpa_dim {\tl_head:N \l__cal_tmpc_tl}
1789   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1790
1791   \dim_set:Nn \l__cal_tmpb_dim {\tl_head:N \l__cal_tmpc_tl}
1792   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1793
1794   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1795
1796   \dim_set:Nn \l__cal_tmpc_dim {\tl_head:N \l__cal_tmpc_tl}
1797   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1798   \dim_set:Nn \l__cal_tmpd_dim {\tl_head:N \l__cal_tmpc_tl}
1799   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1800
1801   \tl_put_right:NV #1 \g__spath_lineto_tl
1802
1803   \tl_put_right:Nx #1 {
1804     {\dim_eval:n {(\l__cal_tmpa_dim + \l__cal_tmpc_dim)/2}}
1805     {\dim_eval:n {(\l__cal_tmpb_dim + \l__cal_tmpd_dim)/2}}
1806   }
1807
1808   \tl_put_right:NV #1 \g__spath_lineto_tl
1809   \tl_put_right:Nx #1 {
1810     {\dim_use:N \l__cal_tmpc_dim}
1811     {\dim_use:N \l__cal_tmpd_dim}
1812   }

```

1813 }

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

`\cal_split_line_in_three:NN` Splits a line in three, adding the splits to the first token list.

```
1814 \cs_new_nopar:Npn \cal_split_line_in_three:NN #1#2
1815 {
1816   \tl_set_eq:NN \l__cal_tmpc_tl #2
1817
1818   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1819
1820   \dim_set:Nn \l__cal_tmppa_dim {\tl_head:N \l__cal_tmpc_tl}
1821   \tl_set:Nx \l__cal_tmppa_tl {\tl_tail:N \l__cal_tmpc_tl}
1822
1823   \dim_set:Nn \l__cal_tmppb_dim {\tl_head:N \l__cal_tmppa_tl}
1824   \tl_set:Nx \l__cal_tmppb_tl {\tl_tail:N \l__cal_tmppa_tl}
1825
1826   \tl_set:Nx \l__cal_tmppc_tl {\tl_tail:N \l__cal_tmppb_tl}
1827
1828   \dim_set:Nn \l__cal_tmppc_dim {\tl_head:N \l__cal_tmppb_tl}
1829   \tl_set:Nx \l__cal_tmppc_tl {\tl_tail:N \l__cal_tmppb_tl}
1830   \dim_set:Nn \l__cal_tmppd_dim {\tl_head:N \l__cal_tmppc_tl}
1831   \tl_set:Nx \l__cal_tmppd_tl {\tl_tail:N \l__cal_tmppc_tl}
1832
1833   \tl_put_right:NV #1 \g__spath_lineto_tl
1834
1835   \tl_put_right:Nx #1 {
1836     {\dim_eval:n {(2\l__cal_tmppa_dim + \l__cal_tmppc_dim)/3}}
1837     {\dim_eval:n {(2\l__cal_tmppb_dim + \l__cal_tmppd_dim)/3}}
1838   }
1839
1840   \tl_put_right:NV #1 \g__spath_lineto_tl
1841
1842   \tl_put_right:Nx #1 {
1843     {\dim_eval:n {(\l__cal_tmppa_dim + 2\l__cal_tmppc_dim)/3}}
1844     {\dim_eval:n {(\l__cal_tmppb_dim + 2\l__cal_tmppd_dim)/3}}
1845   }
1846
1847   \tl_put_right:NV #1 \g__spath_lineto_tl
1848   \tl_put_right:Nx #1 {
1849     {\dim_use:N \l__cal_tmppc_dim}
1850     {\dim_use:N \l__cal_tmppd_dim}
1851   }
1852 }
```

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

`\cal_split_curve_in_two:NN` Splits a curve in two, adding the splits to the first token list.

```
1853 \cs_new_nopar:Npn \cal_split_curve_in_two:NN #1#2
1854 {
```

```

1855 \spath_split_curve:nVNN {.5} #2 \l_tmpa_tl \l_tmpb_tl
1856 \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1857 \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1858 \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1859 \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
1860 \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
1861 \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
1862 \tl_put_right:NV #1 \l_tmpa_tl
1863 \tl_put_right:NV #1 \l_tmpb_tl
1864 }

```

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

`\cal_split_curve_in_three:NN` Splits a curve in three, adding the splits to the first token list.

```

1865 \cs_new_nopar:Npn \cal_split_curve_in_three:NN #1#2
1866 {
1867   \spath_split_curve:nVNN {1/3} #2 \l_tmpa_tl \l_tmpb_tl
1868
1869   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1870   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1871   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1872   \tl_put_right:NV #1 \l_tmpa_tl
1873
1874   \spath_split_curve:nVNN {.5} \l_tmpb_tl \l_tmpa_tl \l_tmpb_tl
1875   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1876   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1877   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1878   \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
1879   \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
1880   \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
1881   \tl_put_right:NV #1 \l_tmpa_tl
1882   \tl_put_right:NV #1 \l_tmpb_tl
1883 }

```

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

`\cal_maybe_taper:N` Possibly tapers the path, depending on the booleans.

```

1884 \cs_new_nopar:Npn \cal_maybe_taper:N #1
1885 {
1886   \tl_set_eq:NN \l__cal_tmpa_tl #1
1887
1888   \bool_if:NT \l__cal_taper_start_bool
1889   {
1890
1891     \dim_set:Nn \l__cal_tmpa_dim {\tl_item:Nn \l__cal_tmpa_tl {2}}
1892     \dim_set:Nn \l__cal_tmpb_dim {\tl_item:Nn \l__cal_tmpa_tl {3}}
1893     \tl_set:Nx \l__cal_tmpb_tl {\tl_item:Nn \l__cal_tmpa_tl {4}}
1894
1895     \tl_case:NnF \l__cal_tmpb_tl
1896     {

```

```

1897 \g__spath_lineto_tl
1898 {
1899
1900     \bool_set_true:N \l__cal_taperable_bool
1901     \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {5}}
1902     \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {6}}
1903     \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpa_dim + \l__cal_tmpg_dim)/3}
1904     \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmph_dim + \l__cal_tmpe_dim)/3}
1905     \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmpe_dim + 2\l__cal_tmpe_dim)/3}
1906     \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmpe_dim + 2\l__cal_tmpe_dim)/3}
1907     \prg_replicate:nn {4}
1908     {
1909         \tl_set:Nx \l__cal_tmpe_tl {\tl_tail:N \l__cal_tmpe_tl}
1910     }
1911     \tl_put_left:NV \l__cal_tmpe_tl \g__spath_moveto_tl
1912 }
1913 \g__spath_curvetoa_tl
1914 {
1915     \bool_set_true:N \l__cal_taperable_bool
1916     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpe_tl {5}}
1917     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpe_tl {6}}
1918     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpe_tl {8}}
1919     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpe_tl {9}}
1920     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpe_tl {11}}
1921     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpe_tl {12}}
1922     \prg_replicate:nn {10}
1923     {
1924         \tl_set:Nx \l__cal_tmpe_tl {\tl_tail:N \l__cal_tmpe_tl}
1925     }
1926     \tl_put_left:NV \l__cal_tmpe_tl \g__spath_moveto_tl
1927 }
1928 }
1929 {
1930     \bool_set_false:N \l__cal_taperable_bool
1931 }
1932
1933 \bool_if:NT \l__cal_taperable_bool
1934 {
1935     \__cal_taper_aux:
1936 }
1937
1938 }
1939
1940 \bool_if:NT \l__cal_taper_end_bool
1941 {
1942
1943     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpe_tl {-2}}
1944     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpe_tl {-1}}
1945     \tl_set:Nx \l__cal_tmpe_tl {\tl_item:Nn \l__cal_tmpe_tl {-3}}
1946

```

```

1947 \tl_case:NnF \l__cal_tmpb_tl
1948 {
1949   \g__spath_lineto_tl
1950   {
1951
1952     \bool_set_true:N \l__cal_taperable_bool
1953     \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {-5}}
1954     \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {-4}}
1955     \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpa_dim + \l__cal_tmpg_dim)/3}
1956     \dim_set:Nn \l__cal_tmpd_dim {(2\l__cal_tmpb_dim + \l__cal_tmph_dim)/3}
1957     \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmpa_dim + 2\l__cal_tmpg_dim)/3}
1958     \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmpe_dim + 2\l__cal_tmph_dim)/3}
1959     \tl_reverse:N \l__cal_tmpa_tl
1960     \prg_replicate:nn {3}
1961     {
1962       \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
1963     }
1964     \tl_reverse:N \l__cal_tmpa_tl
1965   }
1966   \g__spath_curveto_tl
1967   {
1968     \bool_set_true:N \l__cal_taperable_bool
1969     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-5}}
1970     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-4}}
1971     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-8}}
1972     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-7}}
1973     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-11}}
1974     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-10}}
1975     \tl_reverse:N \l__cal_tmpa_tl
1976     \prg_replicate:nn {9}
1977     {
1978       \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
1979     }
1980     \tl_reverse:N \l__cal_tmpa_tl
1981   }
1982 }
1983 {
1984   \bool_set_false:N \l__cal_taperable_bool
1985 }
1986
1987 \bool_if:NT \l__cal_taperable_bool
1988 {
1989   \__cal_taper_aux:
1990 }
1991
1992 }
1993
1994 \pgfsyssoftpath@setcurrentpath\l__cal_tmpa_tl
1995 \pgfsetstrokecolor{\pgfkeysvalueof{/tikz/pen~colour}}
1996 \pgfusepath{stroke}

```



```

1997
1998 }

```

(End definition for \cal\_maybe\_taper:N. This function is documented on page ??.)

\\_cal\_taper\_aux: Auxiliary macro to avoid unnecessary code duplication.

```

1999 \cs_new_nopar:Npn \_cal_taper_aux:
2000 {
2001   \tl_clear:N \l__cal_tmpb_tl
2002   \tl_put_right:NV \l__cal_tmpb_tl \g__spath_moveto_tl
2003
2004   \fp_set:Nn \l__cal_tmpe_dim
2005   {
2006     \l__cal_tmpe_dim - \l__cal_tmpe_dim
2007   }
2008   \fp_set:Nn \l__cal_tmpe_dim
2009   {
2010     \l__cal_tmpe_dim - \l__cal_tmpe_dim
2011   }
2012   \fp_set:Nn \l__cal_tmpe_dim
2013   {
2014     (\l__cal_tmpe_dim^2 + \l__cal_tmpe_dim^2)^.5
2015   }
2016
2017   \fp_set:Nn \l__cal_tmpe_dim { .5*\l__cal_line_width_dim * \l__cal_tmpe_dim / \l__cal_tmpe_dim }
2018   \fp_set:Nn \l__cal_tmpe_dim { .5*\l__cal_line_width_dim * \l__cal_tmpe_dim / \l__cal_tmpe_dim }
2019
2020   \fp_set:Nn \l__cal_tmpe_dim
2021   {
2022     \l__cal_tmpe_dim - \l__cal_tmpe_dim
2023   }
2024   \fp_set:Nn \l__cal_tmpe_dim
2025   {
2026     \l__cal_tmpe_dim - \l__cal_tmpe_dim
2027   }
2028   \fp_set:Nn \l__cal_tmpe_dim
2029   {
2030     (\l__cal_tmpe_dim^2 + \l__cal_tmpe_dim^2)^.5
2031   }
2032
2033   \fp_set:Nn \l__cal_tmpe_dim { .5*\l__cal_line_width_dim * \l__cal_tmpe_dim / \l__cal_tmpe_dim }
2034   \fp_set:Nn \l__cal_tmpe_dim { .5*\l__cal_line_width_dim * \l__cal_tmpe_dim / \l__cal_tmpe_dim }
2035
2036   \tl_put_right:Nx \l__cal_tmpb_tl
2037   {
2038     {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim }}
2039     {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim }}
2040   }
2041
2042   \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetoa_tl

```

```

2043
2044 \tl_put_right:Nx \l__cal_tmpb_tl
2045 {
2046   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
2047   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
2048 }
2049
2050 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetob_tl
2051
2052 \tl_put_right:Nx \l__cal_tmpb_tl
2053 {
2054   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
2055   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
2056 }
2057
2058 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curveto_tl
2059
2060 \tl_put_right:Nx \l__cal_tmpb_tl
2061 {
2062   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
2063   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
2064 }
2065
2066 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetoa_tl
2067
2068 \tl_put_right:Nx \l__cal_tmpb_tl
2069 {
2070   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim - \fp_to_dim:n{ 1.32 * \l__cal_tmpe_dim}}
2071   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim + \fp_to_dim:n {1.32* \l__cal_tmpe_dim}}
2072 }
2073
2074 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetob_tl
2075
2076 \tl_put_right:Nx \l__cal_tmpb_tl
2077 {
2078   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim - \fp_to_dim:n {1.32 * \l__cal_tmpe_dim}}
2079   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim + \fp_to_dim:n {1.32 * \l__cal_tmpe_dim}}
2080 }
2081
2082 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curveto_tl
2083
2084 \tl_put_right:Nx \l__cal_tmpb_tl
2085 {
2086   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
2087   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
2088 }
2089
2090 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetoa_tl
2091
2092 \tl_put_right:Nx \l__cal_tmpb_tl

```

```

2093 {
2094   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpc_fp + \l__cal_tmpe_dim}}
2095   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpd_fp + \l__cal_tmpf_dim}}
2096 }
2097
2098 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetob_tl
2099
2100 \tl_put_right:Nx \l__cal_tmpb_tl
2101 {
2102   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpc_dim}}
2103   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpd_dim}}
2104 }
2105
2106 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curveto_tl
2107
2108 \tl_put_right:Nx \l__cal_tmpb_tl
2109 {
2110   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpe_dim}}
2111   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpf_dim}}
2112 }
2113
2114 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetoa_tl
2115
2116 \tl_put_right:Nx \l__cal_tmpb_tl
2117 {
2118   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpe_dim + \fp_to_dim:n{ 1.32 * \l__cal_tmpe_dim}}
2119   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpf_dim - \fp_to_dim:n{ 1.32 * \l__cal_tmpe_dim}}
2120 }
2121
2122 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetob_tl
2123
2124 \tl_put_right:Nx \l__cal_tmpb_tl
2125 {
2126   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpe_dim + \fp_to_dim:n{ 1.32 * \l__cal_tmpe_dim}}
2127   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpf_dim - \fp_to_dim:n{ 1.32 * \l__cal_tmpe_dim}}
2128 }
2129
2130 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curveto_tl
2131
2132 \tl_put_right:Nx \l__cal_tmpb_tl
2133 {
2134   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpe_dim}}
2135   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpf_dim}}
2136 }
2137
2138 \pgfsyssoftpath@setcurrentpath\l__cal_tmpb_tl
2139 \pgfsetfillcolor{\pgfkeysvalueof{/tikz/pen~colour}}
2140 \pgfusepath{fill}
2141 }

```

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

Defines a copperplate pen.

```
2142 \tl_set:Nn \l__cal_tmpa_tl {\pgfsyssoftpath@movetotoken{0pt}{0pt}}
2143 \MakeSPathList{calligraphy pen copperplate}{\l__cal_tmpa_tl}
2144 \SPathListPrepare{calligraphy pen copperplate}
2145 \ExplSyntaxOff
```

### 3.4 Decorations

If a decoration library is loaded we define some decorations that use the calligraphy library, specifically the copperplate pen with its tapering.

First, a brace decoration.

```
2146 \expandafter\ifx\csname pgfdeclaredecoration\endcsname\relax
2147 \else
2148 \pgfdeclaredecoration{calligraphic brace}{brace}
2149 {
2150   \state{brace}[width+=\pgfdecoratedremainingdistance,next state=final]
2151   {
2152     \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}
2153     \pgfpathmoveto{\pgfpointorigin}
2154     \pgfpathcurveto
2155     {\pgfqqpoint{.15\pgfdecorationsegmentamplitude}{.3\pgfdecorationsegmentamplitude}}
2156     {\pgfqqpoint{.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
2157     {\pgfqqpoint{\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
2158     {
2159       \pgftransformxshift{+\pgfdecorationsegmentaspect\pgfdecoratedremainingdistance}
2160       \pgfpathlineto{\pgfqqpoint{-\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
2161       \pgfpathcurveto
2162       {\pgfqqpoint{-.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
2163       {\pgfqqpoint{-.15\pgfdecorationsegmentamplitude}{.7\pgfdecorationsegmentamplitude}}
2164       {\pgfqqpoint{0\pgfdecorationsegmentamplitude}{1\pgfdecorationsegmentamplitude}}
2165       \pgfpathmoveto{\pgfqqpoint{0\pgfdecorationsegmentamplitude}{1\pgfdecorationsegmentamplitude}}
2166       \pgfpathcurveto
2167       {\pgfqqpoint{.15\pgfdecorationsegmentamplitude}{.7\pgfdecorationsegmentamplitude}}
2168       {\pgfqqpoint{.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
2169       {\pgfqqpoint{\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
2170     }
2171   }
2172   {
2173     \pgftransformxshift{+\pgfdecoratedremainingdistance}
2174     \pgfpathlineto{\pgfqqpoint{-\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
2175     \pgfpathcurveto
2176     {\pgfqqpoint{-.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
2177     {\pgfqqpoint{-.15\pgfdecorationsegmentamplitude}{.3\pgfdecorationsegmentamplitude}}
2178     {\pgfqqpoint{0pt}{0pt}}
2179   }
2180   \tikzset{
2181     taper width=.5\pgflinewidth,
2182     taper
```

```

2182 }%
2183 \pgfsyssoftpath@getcurrentpath\cal@tmp@path
2184 \MakeSPathList{calligraphy path}{\cal@tmp@path}%
2185 \SPathListPrepare{calligraphy path}%
2186 \CalligraphyPathCreate{calligraphy path}{copperplate}%
2187 }
2188 \state{final}{}
2189 }

```

The second is a straightened parenthesis (so that when very large it doesn't bow out too far).

```

2190 \pgfdeclaredecoration{calligraphic straight parenthesis}{brace}
2191 {
2192   \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]
2193   {
2194     \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}
2195     \pgfpathmoveto{\pgfpointorigin}
2196     \pgfpathcurveto
2197     {\pgfpoint{.76604\pgfdecorationsegmentamplitude}{.64279\pgfdecorationsegmentamplitude}}
2198     {\pgfpoint{2.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
2199     {\pgfpoint{3.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
2200     {
2201       \pgftransformxshift{+\pgfdecoratedremainingdistance}
2202       \pgfpathlineto{\pgfpoint{-3.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
2203       \pgfpathcurveto
2204       {\pgfpoint{-2.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
2205       {\pgfpoint{-.76604\pgfdecorationsegmentamplitude}{.64279\pgfdecorationsegmentamplitude}}
2206       {\pgfpoint{0pt}{0pt}}
2207     }
2208     \tikzset{
2209       taper width=.5\pgflinewidth,
2210       taper
2211     }%
2212     \pgfsyssoftpath@getcurrentpath\cal@tmp@path
2213     \MakeSPathList{calligraphy path}{\cal@tmp@path}%
2214     \SPathListPrepare{calligraphy path}%
2215     \CalligraphyPathCreate{calligraphy path}{copperplate}%
2216   }
2217   \state{final}{}%
2218 }

```

The third is a curved parenthesis.

```

2219 \pgfdeclaredecoration{calligraphic curved parenthesis}{brace}
2220 {
2221   \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]
2222   {
2223     \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}
2224     \pgfpathmoveto{\pgfpointorigin}
2225     \pgf@xa=\pgfdecoratedremainingdistance\relax
2226     \advance\pgf@xa by -1.5890\pgfdecorationsegmentamplitude\relax

```

```

2227 \edef\cgrphy@xa{\the\pgf@xa}
2228 \pgfpathcurveto
2229 {\pgfqpoint{1.5890\pgfdecorationsegmentamplitude}{1.3333\pgfdecorationsegmentamplitude}}
2230 {\pgfqpoint{\cgrphy@xa}{1.3333\pgfdecorationsegmentamplitude}}
2231 {\pgfqpoint{\pgfdecoratedremainingdistance}{0pt}}
2232 \tikzset{
2233   taper width=.5\pgflinewidth,
2234   taper
2235 }%
2236 \pgfsyssoftpath@getcurrentpath\cal@tmp@path
2237 \MakeSPathList{calligraphy path}{\cal@tmp@path}%
2238 \SPathListPrepare{calligraphy path}%
2239 \CalligraphyPathCreate{calligraphy path}{copperplate}%
2240 }
2241 \state{final}{}%
2242 }

```

End the conditional for if pgfdecoration module is loaded

```

2243 \fi

```

## 4 Drawing Knots

### 4.1 Initialisation

We load the `spath3` library and the `intersections` TikZ library. Then we get going.

```

2244 \RequirePackage{spath3}
2245 \usetikzlibrary{intersections}
2246
2247 \ExplSyntaxOn
2248
2249 \tl_new:N \l__knot_tmpa_tl
2250 \tl_new:N \l__knot_tmpb_tl
2251 \tl_new:N \l__knot_tmpc_tl
2252 \tl_new:N \l__knot_tmpd_tl
2253 \tl_new:N \l__knot_tmpe_tl
2254 \tl_new:N \l__knot_tmpf_tl
2255 \tl_new:N \l__knot_tmpg_tl
2256 \tl_new:N \l__knot_redraws_tl
2257 \tl_new:N \l__knot_clip_width_tl
2258 \tl_new:N \l__knot_name_tl
2259 \tl_new:N \l__knot_node_tl
2260
2261 \int_new:N \l__knot_tmpa_int
2262 \int_new:N \l__knot_strands_int
2263 \int_new:N \l__knot_intersections_int
2264 \int_new:N \l__knot_filaments_int
2265 \int_new:N \l__knot_component_start_int
2266
2267 \dim_new:N \l__knot_tmpa_dim

```

```

2268 \dim_new:N \l__knot_tmpb_dim
2269 \dim_new:N \l__knot_tmpc_dim
2270 \dim_new:N \l__knot_tolerance_dim
2271 \dim_new:N \l__knot_clip_radius_dim
2272
2273 \bool_new:N \l__knot_draft_bool
2274 \bool_new:N \l__knot_ignore_ends_bool
2275 \bool_new:N \l__knot_self_intersections_bool
2276 \bool_new:N \l__knot_splits_bool
2277 \bool_new:N \l__knot_super_draft_bool
2278
2279 \bool_new:N \l__knot_prepend_prev_bool
2280 \bool_new:N \l__knot_append_next_bool
2281 \bool_new:N \l__knot_skip_bool
2282
2283 \bool_set_true:N \l__knot_ignore_ends_bool
Configuration is via TikZ keys and styles.
2284 \tikzset{
2285   knot/.code={
2286     \tl_if_eq:nnTF {#1} {none}
2287     {
2288       \tikz@addmode{\tikz@mode@doublefalse}
2289     }
2290     {
2291       \tikz@addmode{\tikz@mode@doubletrue}
2292       \tl_if_eq:nnTF {\pgfkeysnovalue} {#1}
2293       {
2294         \tikz@addoption{\pgfsetinnerstrokecolor{.}}
2295       }
2296       {
2297         \pgfsetinnerstrokecolor{#1}
2298       }
2299       \tikz@addoption{
2300         \pgfsetstrokecolor{knotbg}
2301       }
2302       \tl_set:Nn \tikz@double@setup{
2303         \pgfsetinnerlinewidth{\pgflinewidth}
2304         \pgfsetlinewidth{\dim_eval:n {\tl_use:N \l__knot_gap_tl \pgflinewidth}}
2305       }
2306     }
2307   },
2308   knot~ gap/.store~ in=\l__knot_gap_tl,
2309   knot~ gap=3,
2310   knot~ diagram/.is-family,
2311   knot~ diagram/.unknown/.code={
2312     \tl_set_eq:NN \l__knot_tmpa_tl \pgfkeyscurrentname
2313     \pgfkeysalso{
2314       /tikz/\l__knot_tmpa_tl=#1
2315     }

```

```

2316 },
2317 background~ colour/.code={%
2318   \colorlet{knotbg}{#1}%
2319 },
2320 background~ color/.code={%
2321   \colorlet{knotbg}{#1}%
2322 },
2323 background~ colour=white,
2324 knot~ diagram,
2325 name/.store~in=\l__knot_name_tl,
2326 name={knot},
2327 every~ strand/.style={draw},
2328 ignore~ endpoint~ intersections/.code={
2329   \tl_if_eq:nnTF {#1} {true}
2330   {
2331     \bool_set_true:N \l__knot_ignore_ends_bool
2332   }
2333   {
2334     \bool_set_false:N \l__knot_ignore_ends_bool
2335   }
2336 },
2337 ignore~ endpoint~ intersections/.default=true,
2338 consider~ self~ intersections/.is~choice,
2339 consider~ self~ intersections/true/.code={
2340   \bool_set_true:N \l__knot_self_intersections_bool
2341   \bool_set_true:N \l__knot_splits_bool
2342 },
2343 consider~ self~ intersections/false/.code={
2344   \bool_set_false:N \l__knot_self_intersections_bool
2345   \bool_set_false:N \l__knot_splits_bool
2346 },
2347 consider~ self~ intersections/no~ splits/.code={
2348   \bool_set_true:N \l__knot_self_intersections_bool
2349   \bool_set_false:N \l__knot_splits_bool
2350 },
2351 consider~ self~ intersections/.default={true},
2352 clip~ radius/.code={
2353   \dim_set:Nn \l__knot_clip_radius_dim {#1}
2354 },
2355 clip~ radius=10pt,
2356 end~ tolerance/.code={
2357   \dim_set:Nn \l__knot_tolerance_dim {#1}
2358 },
2359 end~ tolerance=14pt,
2360 clip~ width/.code={
2361   \tl_set:Nn \l__knot_clip_width_tl {#1}
2362 },
2363 clip~ width=3,
2364 flip~ crossing/.code={%
2365   \tl_clear_new:c {l__knot_crossing_#1}

```



```

2366   \tl_set:cn {l__knot_crossing_#1} {x}
2367 },
2368 draft~ mode/.is~ choice,
2369 draft~ mode/off/.code={%
2370   \bool_set_false:N \l__knot_draft_bool
2371   \bool_set_false:N \l__knot_super_draft_bool
2372 },
2373 draft~ mode/crossings/.code={%
2374   \bool_set_true:N \l__knot_draft_bool
2375   \bool_set_false:N \l__knot_super_draft_bool
2376 },
2377 draft~ mode/strands/.code={%
2378   \bool_set_true:N \l__knot_draft_bool
2379   \bool_set_true:N \l__knot_super_draft_bool
2380 },
2381 draft/.is~ family,
2382 draft,
2383 crossing~ label/.style={
2384   overlay,
2385   fill=white,
2386   fill~ opacity=.5,
2387   text~ opacity=1,
2388   text=blue,
2389   pin~ edge={blue,<-}
2390 },
2391 strand~ label/.style={
2392   overlay,
2393   circle,
2394   draw=purple,
2395   fill=white,
2396   fill~ opacity=.5,
2397   text~ opacity=1,
2398   text=purple,
2399   inner~ sep=0pt
2400 },
2401 }

```

Wrapper around `\tikzset` for applying keys from a token list, checking for if the given token list exists.

```

2402 \cs_new_nopar:Npn \knot_apply_style:N #1
2403 {
2404   \tl_if_exist:NT #1 {
2405     \exp_args:NV \tikzset #1
2406   }
2407 }
2408 \cs_generate_variant:Nn \knot_apply_style:N {c}

```

`\flipcrossings` The user can specify a comma separated list of crossings to flip.

```

2409 \NewDocumentCommand \flipcrossings {m}
2410 {

```

```

2411 \tikzset{knot~ diagram/flip~ crossing/.list={#1}}%
2412 }

```

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

`\strand` This is how the user specifies a strand of the knot.

```

2413 \NewDocumentCommand \strand { 0{} }
2414 {
2415   \int_incr:N \l__knot_strands_int
2416   \tl_clear_new:c {l__knot_options_strand \int_use:N \l__knot_strands_int}
2417   \tl_set:cn {l__knot_options_strand \int_use:N \l__knot_strands_int} {#1}
2418   \path[#1,save~ spath=knot strand \int_use:N \l__knot_strands_int]
2419 }

```

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

`knot` This is the wrapper environment that calls the knot generation code.

```

2420 \NewDocumentEnvironment{knot} { 0{} }
2421 {
2422   \knot_initialise:n {#1}
2423 }
2424 {
2425   \knot_render:
2426 }

```

(End definition for `knot`. This function is documented on page ??.)

`\knot_initialise:n` Set up some stuff before loading in the strands.

```

2427 \cs_new_protected_nopar:Npn \knot_initialise:n #1
2428 {
2429   \tikzset{knot~ diagram/.cd,every~ knot~ diagram/.try,#1}
2430   \int_zero:N \l__knot_strands_int
2431   \tl_clear:N \l__knot_redraws_tl
2432 }

```

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

`\knot_render:` This is the code that starts the work of rendering the knot.

```

2433 \cs_new_protected_nopar:Npn \knot_render:
2434 {

```

Start a scope and reset the transformation (since all transformations have already been taken into account when defining the strands).

```

2435   \pgfscope
2436   \pgftransformreset

```

Loop through the strands drawing each one for the first time.

```

2437   \int_step_function:nnnN {1} {1} {\l__knot_strands_int} \knot_draw_strand:n

```

In super draft mode we don't do anything else.

```

2438   \bool_if:NF \l__knot_super_draft_bool
2439   {

```

In draft mode we draw labels at the ends of the strands; this also handles splitting curves to avoid self-intersections of Bezier curves if that's requested.

```
2440 \int_step_function:nnnN {1} {1} {\l__knot_strands_int} \knot_draw_labels:n
```

If we're considering self intersections we need to split the strands into filaments.

```
2441 \bool_if:NTF \l__knot_self_intersections_bool
2442 {
2443   \knot_split_strands:
2444   \int_set_eq:NN \l__knot_tmpa_int \l__knot_filaments_int
2445   \tl_set:Nn \l__knot_prefix_tl {filament}
2446 }
2447 {
2448   \int_set_eq:NN \l__knot_tmpa_int \l__knot_strands_int
2449   \tl_set:Nn \l__knot_prefix_tl {strand}
2450 }
```

Initialise the intersection count.

```
2451 \int_gzero:N \l__knot_intersections_int
```

If in draft mode we label the intersections, otherwise we just stick a coordinate at each one.

```
2452 \bool_if:NTF \l__knot_draft_bool
2453 {
2454   \tl_set:Nn \l__knot_node_tl
2455   {\node[coordinate,pin={\knot~ diagram/draft/crossing~ label}]{\int_use:N \l__knot_interse
2456 }
2457 {
2458   \tl_set:Nn \l__knot_node_tl {\node[coordinate]}
2459 }
```

This double loop steps through the pieces (strands or filaments) and computes the intersections and does stuff with those.

```
2460 \int_step_variable:nnnNn {1} {1} {\l__knot_tmpa_int - 1} \l__knot_tmpa_tl
2461 {
2462   \int_step_variable:nnnNn {\tl_use:N \l__knot_tmpa_tl + 1} {1} {\l__knot_tmpa_int} \
2463   {
2464     \knot_intersections:VV \l__knot_tmpa_tl \l__knot_tmpb_tl
2465   }
2466 }
```

If any redraws were requested, do them here.

```
2467 \tl_use:N \l__knot_redraws_tl
2468 }
```

Close the scope

```
2469 \endpgfscope
2470 }
```

*(End definition for \knot\_render:. This function is documented on page ??.)*

`\knot_draw_strand:n` This renders a strand using the options originally specified.

```

2471 \cs_new_protected_nopar:Npn \knot_draw_strand:n #1
2472 {
2473   \pgfscope
2474   \group_begin:
2475   \tl_set:Nn \l_tmpa_tl {knot~ diagram/every~ strand/.try,}
2476   \tl_put_right:Nv \l_tmpa_tl {l__knot_options_strand #1}
2477   \tl_put_right:Nn \l_tmpa_tl {,knot~ diagram/only~ when~ rendering/.try,only~ when~ rendering}
2478   \spath_bake_round:n {knot strand #1}
2479   \spath_tikz_path:Vn \l_tmpa_tl {knot strand #1}
2480   \group_end:
2481   \endpgfscope
2482 }
2483 \cs_generate_variant:Nn \tl_put_right:Nn {Nv}

```

*(End definition for \knot\_draw\_strand:n. This function is documented on page ??.)*

`\knot_draw_labels:n` Draw a label at each end of each strand, if in draft mode. Also, if requested, split potentially self intersecting Bezier curves.

```

2484 \cs_new_protected_nopar:Npn \knot_draw_labels:n #1
2485 {
2486   \bool_if:NT \l__knot_draft_bool
2487   {
2488     \spath_get:nnN {knot strand #1} {final point} \l__knot_tmpb_tl
2489     \dim_set:Nn \l__knot_tmpa_dim {\tl_item:Nn \l__knot_tmpb_tl {1}}
2490     \dim_set:Nn \l__knot_tmpb_dim {\tl_item:Nn \l__knot_tmpb_tl {2}}
2491     \node[knot~ diagram/draft/strand~label] at (\l__knot_tmpa_dim,\l__knot_tmpb_dim) {#1};
2492     \spath_get:nnN {knot strand #1} {initial point} \l__knot_tmpb_tl
2493     \dim_set:Nn \l__knot_tmpa_dim {\tl_item:Nn \l__knot_tmpb_tl {1}}
2494     \dim_set:Nn \l__knot_tmpb_dim {\tl_item:Nn \l__knot_tmpb_tl {2}}
2495     \node[knot~ diagram/draft/strand~label] at (\l__knot_tmpa_dim,\l__knot_tmpb_dim) {#1};
2496   }
2497   \bool_if:nT {
2498     \l__knot_self_intersections_bool
2499     &&
2500     \l__knot_splits_bool
2501   }
2502   {
2503     \tl_clear:N \l__knot_tmpa_tl
2504     \spath_map_segment_function:nN {knot strand #1} \knot_split_self_intersects:NN
2505     \spath_put:nnV {knot strand #1} {path} \l__knot_tmpa_tl
2506   }
2507 }

```

*(End definition for \knot\_draw\_labels:n. This function is documented on page ??.)*

`\knot_split_self_intersects:NN` This is the macro that does the split. Figuring out whether a Bezier cubic self intersects is apparently a difficult problem so we don't bother. We compute a point such that if there is an intersection then it lies on either side of the point. I don't recall where the formula came from!

```

2508 \cs_new_protected_nopar:Npn \knot_split_self_intersects:NN #1#2
2509 {
2510   \tl_case:NnF #1
2511   {
2512     \g__spath_curvetoa_tl
2513     {
2514       \fp_set:Nn \l_tmpa_fp
2515       {
2516         (\tl_item:Nn #2 {3} - 3 * \tl_item:Nn #2 {6} + 3 * \tl_item:Nn #2 {9} - \tl_item:Nn #2 {12})
2517         *
2518         (3 * \tl_item:Nn #2 {8} - 3 * \tl_item:Nn #2 {11})
2519         -
2520         (\tl_item:Nn #2 {2} - 3 * \tl_item:Nn #2 {5} + 3 * \tl_item:Nn #2 {8} - \tl_item:Nn #2 {11})
2521         *
2522         (3 * \tl_item:Nn #2 {9} - 3 * \tl_item:Nn #2 {12})
2523       }
2524       \fp_set:Nn \l_tmpb_fp
2525       {
2526         (\tl_item:Nn #2 {2} - 3 * \tl_item:Nn #2 {5} + 3 * \tl_item:Nn #2 {8} - \tl_item:Nn #2 {11})
2527         *
2528         (3 * \tl_item:Nn #2 {6} - 6 * \tl_item:Nn #2 {9} + 3 * \tl_item:Nn #2 {12})
2529         -
2530         (\tl_item:Nn #2 {3} - 3 * \tl_item:Nn #2 {6} + 3 * \tl_item:Nn #2 {9} - \tl_item:Nn #2 {12})
2531         *
2532         (3 * \tl_item:Nn #2 {5} - 6 * \tl_item:Nn #2 {8} + 3 * \tl_item:Nn #2 {11})
2533       }
2534       \fp_compare:nTF
2535       {
2536         \l_tmpb_fp != 0
2537       }
2538       {
2539         \fp_set:Nn \l_tmpa_fp {.5 * \l_tmpa_fp / \l_tmpb_fp}
2540         \fp_compare:nTF
2541         {
2542           0 < \l_tmpa_fp && \l_tmpa_fp < 1
2543         }
2544         {
2545           \spath_split_curve:VVNN \l_tmpa_fp #2 \l_tmpa_tl \l_tmpb_tl
2546           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2547           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2548           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2549           \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
2550           \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
2551           \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
2552           \tl_put_right:NV \l__knot_tmpa_tl \l_tmpa_tl
2553           \tl_put_right:NV \l__knot_tmpa_tl \l_tmpb_tl
2554         }
2555       }
2556       \tl_set_eq:NN \l_tmpa_tl #2
2557       \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}

```

```

2558         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2559         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2560         \tl_put_right:NV \l__knot_tmpa_tl \l_tmpa_tl
2561     }
2562 }
2563 {
2564     \tl_set_eq:NN \l_tmpa_tl #2
2565     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2566     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2567     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2568     \tl_put_right:NV \l__knot_tmpa_tl \l_tmpa_tl
2569 }
2570 }
2571 \g__spath_lineto_tl
2572 {
2573     \tl_set_eq:NN \l_tmpa_tl #2
2574     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2575     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2576     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2577     \tl_put_right:NV \l__knot_tmpa_tl \l_tmpa_tl
2578 }
2579 }
2580 {
2581     \tl_put_right:NV \l__knot_tmpa_tl #2
2582 }
2583 }

```

*(End definition for \knot\_split\_self\_intersects:NN. This function is documented on page ??.)*

`\knot_intersections:nn` This computes the intersections of two pieces and steps through them.

```

2584 \cs_new_protected_nopar:Npn \knot_intersections:nn #1#2
2585 {
2586     \group_begin:
2587     \tl_set_eq:NN \l__knot_tmpa_tl \l__knot_prefix_tl
2588     \tl_put_right:Nn \l__knot_tmpa_tl {#1}
2589     \tl_set_eq:NN \l__knot_tmpb_tl \l__knot_prefix_tl
2590     \tl_put_right:Nn \l__knot_tmpb_tl {#2}
2591     \spath_get:nnN {knot \tl_use:N \l__knot_tmpa_tl} {path} \l__knot_tmpc_tl
2592     \spath_get:nnN {knot \tl_use:N \l__knot_tmpb_tl} {path} \l__knot_tmpd_tl
2593
2594     \pgfintersectionofpaths{\pgfsetpath\l__knot_tmpc_tl}{\pgfsetpath\l__knot_tmpd_tl}
2595
2596     \int_compare:nT {\pgfintersectionsolutions > 0}
2597     {
2598         \int_step_function:nnnN {1} {1} {\pgfintersectionsolutions} \knot_do_intersection:n
2599     }
2600     \group_end:
2601 }

```

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

`\knot_do_intersection:n` This handles a specific intersection.

```
2602 \cs_new_protected_nopar:Npn \knot_do_intersection:n #1
2603 {
```

Get the intersection coordinates.

```
2604 \pgfpointintersectionsolution{#1}
2605 \dim_set:Nn \l__knot_tmpa_dim {\pgf@x}
2606 \dim_set:Nn \l__knot_tmpb_dim {\pgf@y}
```

If we're dealing with filaments, we can get false positives from the end points.

```
2607 \bool_set_false:N \l__knot_skip_bool
2608 \bool_if:NT \l__knot_self_intersections_bool
2609 {
```

If one filament preceded the other, test for the intersection being at the relevant end point.

```
2610 \tl_set:Nn \l_tmpa_tl {knot previous}
2611 \tl_put_right:NV \l_tmpa_tl \l__knot_tmpa_tl
2612 \tl_set:Nv \l_tmpa_tl \l_tmpa_tl
2613 \tl_if_eq:NNT \l_tmpa_tl \l__knot_tmpb_tl
2614 {
2615   \knot_test_endpoint:VnT \l__knot_tmpb_tl {final point}
2616   {
2617     \bool_set_true:N \l__knot_skip_bool
2618   }
2619 }
2620
2621 \tl_set:Nn \l_tmpa_tl {knot previous}
2622 \tl_put_right:NV \l_tmpa_tl \l__knot_tmpb_tl
2623 \tl_set:Nv \l_tmpa_tl \l_tmpa_tl
2624 \tl_if_eq:NNT \l_tmpa_tl \l__knot_tmpa_tl
2625 {
2626   \knot_test_endpoint:VnT \l__knot_tmpa_tl {final point}
2627   {
2628     \bool_set_true:N \l__knot_skip_bool
2629   }
2630 }
2631 }
```

The can also say that end points of filaments (or strands) should simply be ignored anyway.

```
2632 \bool_if:NT \l__knot_ignore_ends_bool
2633 {
2634   \knot_test_endpoint:VnT \l__knot_tmpa_tl {initial point}
2635   {
2636     \bool_set_true:N \l__knot_skip_bool
2637   }
2638   \knot_test_endpoint:VnT \l__knot_tmpa_tl {final point}
2639   {
2640     \bool_set_true:N \l__knot_skip_bool
2641   }

```

```

2642 \knot_test_endpoint:VnT \l__knot_tmpb_tl {initial point}
2643 {
2644   \bool_set_true:N \l__knot_skip_bool
2645 }
2646 \knot_test_endpoint:VnT \l__knot_tmpb_tl {final point}
2647 {
2648   \bool_set_true:N \l__knot_skip_bool
2649 }
2650 }

```

Assuming that we passed all the above tests, we render the crossing.

```

2651 \bool_if:NF \l__knot_skip_bool
2652 {
2653
2654   \int_gincr:N \l__knot_intersections_int

```

This is the flip test. We only render one of the paths. The “flip” swaps which one we render.

```

2655 \bool_if:nTF
2656 {
2657   \tl_if_exist_p:c {l__knot_crossing_ \int_use:N
2658     \l__knot_intersections_int}
2659   &&
2660   ! \tl_if_empty_p:c {l__knot_crossing_ \int_use:N
2661     \l__knot_intersections_int}
2662 }
2663 {
2664   \tl_set_eq:NN \l__knot_tmpg_tl \l__knot_tmpb_tl
2665 }
2666 {
2667   \tl_set_eq:NN \l__knot_tmpg_tl \l__knot_tmpa_tl
2668 }

```

Now we know which one we’re rendering, we test to see if we should also render its predecessor or successor to ensure that we render a path through the entire crossing region.

```

2669 \bool_if:NT \l__knot_self_intersections_bool
2670 {
2671   \knot_test_endpoint:VnT \l__knot_tmpg_tl {initial point}
2672   {
2673     \bool_set_true:N \l__knot_prepend_prev_bool
2674   }
2675   {
2676     \bool_set_false:N \l__knot_prepend_prev_bool
2677   }
2678
2679   \knot_test_endpoint:VnT \l__knot_tmpg_tl {final point}
2680   {
2681     \bool_set_true:N \l__knot_append_next_bool
2682   }
2683   {

```



```

2684     \bool_set_false:N \l__knot_append_next_bool
2685   }

```

If either of those tests succeeded, do the appending or prepending.

```

2686   \bool_if:nT
2687   {
2688     \l__knot_prepend_prev_bool || \l__knot_append_next_bool
2689   }
2690   {
2691     \spath_clone:nn {knot \tl_use:N \l__knot_tmpg_tl}
2692     {knot \tl_use:N \l__knot_prefix_tl -1}
2693
2694     \tl_set_eq:cc {l__knot_options_ \tl_use:N \l__knot_prefix_tl -1} {l__knot_options_ \tl_use:N \l__knot_prefix_tl -1}
2695
2696     \bool_if:nT
2697     {
2698       \l__knot_prepend_prev_bool
2699       &&
2700       \tl_if_exist_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
2701       &&
2702       !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
2703     }
2704     {
2705       \spath_prepend_no_move:nn {knot \tl_use:N \l__knot_prefix_tl -1} {knot \tl_use:c {knot \tl_use:N \l__knot_prefix_tl -1}}

```

If we split potentially self intersecting curves, we test to see if we should prepend yet another segment.

```

2706     \bool_if:nT
2707     {
2708       \l__knot_splits_bool
2709       &&
2710       \tl_if_exist_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
2711       &&
2712       !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
2713     }
2714     {
2715       \knot_test_endpoint:vnT {knot previous \tl_use:N \l__knot_tmpg_tl} {initial point}
2716       {
2717         \spath_get:nnN {knot \tl_use:N \l__knot_prefix_tl -1} {path} \l_tmpa_tl
2718         \spath_prepend_no_move:nn {knot \tl_use:N \l__knot_prefix_tl -1} {knot \tl_use:N \l__knot_prefix_tl -1}
2719         \spath_get:nnN {knot \tl_use:N \l__knot_prefix_tl -1} {path} \l_tmpa_tl
2720       }
2721     }
2722   }
2723 }

```

Now the same for appending.

```

2724   \bool_if:nT
2725   {
2726     \l__knot_append_next_bool
2727     &&

```

```

2728     \tl_if_exist_p:c {knot next \tl_use:N \l__knot_tmpg_tl}
2729     &&
2730     !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
2731   }
2732   {
2733     \spath_append_no_move:nn {knot \tl_use:N \l__knot_prefix_tl -1} {knot \tl_use:c {kn
2734     \bool_if:nT
2735     {
2736       \l__knot_splits_bool
2737       &&
2738       \tl_if_exist_p:c {knot previous \tl_use:N
2739         \l__knot_tmpg_tl}
2740       &&
2741       !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
2742     }
2743     {
2744       \knot_test_endpoint:vnT {knot previous \tl_use:N \l__knot_tmpg_tl} {final point}
2745       {
2746         \spath_append_no_move:nn {knot \tl_use:N \l__knot_prefix_tl -1} {knot \tl_use:c
2747       }
2748     }
2749   }
2750 }
2751
2752 \tl_set:Nn \l__knot_tmpg_tl {\tl_use:N \l__knot_prefix_tl -1}
2753 }
2754 }

```

Now we render the crossing.

```

2755   \pgfscope
2756   \group_begin:
2757   \tikzset{knot~ diagram/every~ intersection/.try, every~ intersection/.try, knot~ diagram/
2758   \knot_draw_crossing:nVV {\tl_use:N \l__knot_tmpg_tl} \l__knot_tmpa_dim \l__knot_tmpb_dim
2759   \group_end:
2760   \endpgfscope

```

And stick a coordinate possibly with a label at the crossing.

```

2761     \tl_use:N \l__knot_node_tl (\l__knot_name_tl \c_space_tl \int_use:N \l__knot_intersections
2762   }
2763 }
2764
2765 \cs_generate_variant:Nn \knot_intersections:nn {VV}

```

*(End definition for \knot\_do\_intersection:n. This function is documented on page ??.)*

`\knot_test_endpoint:N` Test whether the point is near the intersection point.

```

2766 \prg_new_conditional:Npnn \knot_test_endpoint:N #1 {p,T,F,TF}
2767 {
2768   \dim_compare:nTF
2769   {
2770     \dim_abs:n {\pgf@x - \tl_item:Nn #1 {1}}

```

```

2771 +
2772 \dim_abs:n {\pgf@y - \tl_item:Nn #1 {2}}
2773 <
2774 \l_knot_tolerance_dim
2775 }
2776 {
2777 \prg_return_true:
2778 }
2779 {
2780 \prg_return_false:
2781 }
2782 }

```

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

`\knot_test_endpoint:nn` Wrapper around the above.

```

2783 \prg_new_protected_conditional:Npnn \knot_test_endpoint:nn #1#2 {T,F,TF}
2784 {
2785 \spath_get:nnN {knot #1} {#2} \l_knot_tmpd_tl
2786 \knot_test_endpoint:NTF \l_knot_tmpd_tl
2787 {
2788 \prg_return_true:
2789 }
2790 {
2791 \prg_return_false:
2792 }
2793 }
2794
2795 \cs_generate_variant:Nn \knot_test_endpoint:nnT {VnT,vnT}
2796 \cs_generate_variant:Nn \knot_test_endpoint:nnF {VnF,vnF}
2797 \cs_generate_variant:Nn \knot_test_endpoint:nnTF {VnTF,vnTF}

```

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

`\knot_draw_crossing:nnn` This is the code that actually renders a crossing.

```

2798 \cs_new_nopar:Npn \knot_draw_crossing:nnn #1#2#3
2799 {
2800 \group_begin:
2801 \pgfscope
2802 \clip (#2, #3) circle[radius=\l_knot_clip_radius_dim];
2803
2804 \tl_set:Nn \l_tmpa_tl {knot~ diagram/every~ strand/.try,}
2805 \tl_if_exist:cT {l_knot_options_ #1}
2806 {
2807 \tl_put_right:Nv \l_tmpa_tl {l_knot_options_ #1}
2808 }
2809 \tl_put_right:Nn \l_tmpa_tl {,knotbg,line~ width= \tl_use:N \l_knot_clip_width_tl * \pgflinewidth}
2810 \spath_tikz_path:Vn \l_tmpa_tl {knot #1}
2811
2812 \endpgfscope

```

```

2813
2814 \pgfscope
2815 \clip (#2, #3) circle[radius=1.1\l__knot_clip_radius_dim];
2816
2817 \tl_set:Nn \l_tmpa_tl {knot~ diagram/every~ strand/.try,}
2818 \tl_if_exist:cT {l__knot_options_ #1}
2819 {
2820 \tl_put_right:Nv \l_tmpa_tl {l__knot_options_ #1}
2821 }
2822 \tl_put_right:Nn \l_tmpa_tl {,knot~ diagram/only~ when~ rendering/.try,only~ when~ rendering}
2823 \spath_tikz_path:Vn \l_tmpa_tl {knot #1}
2824
2825 \endpgfscope
2826 \group_end:
2827 }
2828
2829 \cs_generate_variant:Nn \knot_draw_crossing:nnn {nVV}

```

*(End definition for \knot\_draw\_crossing:nnn. This function is documented on page ??.)*

`\knot_split_strands:` This, and the following macros, are for splitting strands into filaments.

```

2830 \cs_new_protected_nopar:Npn \knot_split_strands:
2831 {
2832 \int_gzero:N \l__knot_filaments_int
2833 \int_step_function:nnnN {1} {1} {\l__knot_strands_int} \knot_split_strand:n
2834 \int_step_function:nnnN {1} {1} {\l__knot_filaments_int} \knot_compute_nexts:n
2835 }

```

*(End definition for \knot\_split\_strands:. This function is documented on page ??.)*

`\knot_compute_nexts:n` Each filament needs to know its predecessor and successor. We work out the predecessors as we go along, this fills in the successors.

```

2836 \cs_new_protected_nopar:Npn \knot_compute_nexts:n #1
2837 {
2838 \tl_clear_new:c {knot next \tl_use:c {knot previous filament #1}}
2839 \tl_set:cn {knot next \tl_use:c {knot previous filament #1}} {filament #1}
2840 }

```

*(End definition for \knot\_compute\_nexts:n. This function is documented on page ??.)*

`\knot_split_strand:n` Sets up the split for a single strand.

```

2841 \cs_new_protected_nopar:Npn \knot_split_strand:n #1
2842 {
2843 \int_set_eq:NN \l__knot_component_start_int \l__knot_filaments_int
2844 \int_incr:N \l__knot_component_start_int
2845 \tl_set_eq:Nc \l__knot_tmpa_tl {l__knot_options_strand #1}
2846 \spath_map_segment_function:nN {knot strand #1} \knot_save_filament:NN
2847 }

```

*(End definition for \knot\_split\_strand:n. This function is documented on page ??.)*

`\knot_save_filament:NN` Saves a filament as a new `spath` object.

```

2848 \cs_new_protected_nopar:Npn \knot_save_filament:NN #1#2
2849 {
2850   \tl_case:NnF #1
2851   {
2852     \g__spath_moveto_tl
2853     {
2854       \int_compare:nT {\l__knot_component_start_int < \l__knot_filaments_int}
2855       {
2856         \int_set_eq:NN \l__knot_component_start_int \l__knot_filaments_int
2857       }
2858     }
2859     \g__spath_lineto_tl
2860     {
2861       \int_gincr:N \l__knot_filaments_int
2862       \spath_clear_new:n {knot filament \int_use:N \l__knot_filaments_int}
2863       \spath_put:nnV {knot filament \int_use:N \l__knot_filaments_int} {path} #2
2864       \tl_set_eq:cN {l__knot_options_filament \int_use:N \l__knot_filaments_int} \l__knot_tmpa_tl #2
2865
2866       \tl_clear_new:c {knot previous filament \int_use:N \l__knot_filaments_int}
2867       \int_compare:nF {\l__knot_component_start_int == \l__knot_filaments_int}
2868       {
2869         \tl_set:cx {knot previous filament \int_use:N \l__knot_filaments_int} {filament \int_use:N \l__knot_filaments_int}
2870       }
2871     }
2872     \g__spath_curvetoa_tl
2873     {
2874       \int_gincr:N \l__knot_filaments_int
2875       \spath_clear_new:n {knot filament \int_use:N \l__knot_filaments_int}
2876       \spath_put:nnV {knot filament \int_use:N \l__knot_filaments_int} {path} #2
2877       \tl_set_eq:cN {l__knot_options_filament \int_use:N \l__knot_filaments_int} \l__knot_tmpa_tl #2
2878
2879       \tl_clear_new:c {knot previous filament \int_use:N \l__knot_filaments_int}
2880       \int_compare:nF {\l__knot_component_start_int == \l__knot_filaments_int}
2881       {
2882         \tl_set:cx {knot previous filament \int_use:N \l__knot_filaments_int} {filament \int_use:N \l__knot_filaments_int}
2883       }
2884     }
2885     \g__spath_close_tl
2886     {
2887       \int_gincr:N \l__knot_filaments_int
2888       \spath_clear_new:n {knot filament \int_use:N \l__knot_filaments_int}
2889       \tl_set_eq:NN \l__knot_tmpa_tl #2
2890       \tl_set:Nx \l__knot_tmpa_tl {\tl_item:Nn #2 {1}\tl_item:Nn #2 {2}\tl_item:Nn #2 {3}}
2891       \tl_put_right:NV \l__knot_tmpa_tl \g__spath_lineto_tl
2892       \tl_set:Nx \l__knot_tmpa_tl {\tl_item:Nn #2 {5}\tl_item:Nn #2 {6}}
2893       \spath_put:nnV {knot filament \int_use:N \l__knot_filaments_int} {path} \l__knot_tmpa_tl
2894       \tl_set_eq:cN {l__knot_options_filament \int_use:N \l__knot_filaments_int} \l__knot_tmpa_tl #2
2895       \tl_clear_new:c {knot previous filament \int_use:N \l__knot_filaments_int}

```

```

2896     \int_compare:nF {\l__knot_component_start_int == \l__knot_filaments_int}
2897     {
2898         \tl_set:cx {knot previous filament \int_use:N \l__knot_filaments_int} {filament \int_e
2899     }
2900     \tl_set:cx {knot previous filament \int_use:N \l__knot_component_start_int} {filament \
2901 }
2902 }
2903 {
2904 }
2905 }

```

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

`\redraw` The user can redraw segments of the strands at specific locations.

```

2906 \NewDocumentCommand \redraw { m m }
2907 {
2908     \tikz@scan@one@point\pgfutil@firstofone #2 \relax
2909     \tl_put_right:Nn \l__knot_redraws_tl {\knot_draw_crossing:nnn}
2910     \tl_put_right:Nx \l__knot_redraws_tl {
2911         {strand #1} {\dim_use:N \pgf@x} {\dim_use:N \pgf@y}
2912     }
2913 }

```

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

```

2914 \ExplSyntaxOff

```

`\pgf@sh@@knotanchor` Add the extra anchors for the knot crossing nodes.

```

2915 \def\pgf@sh@@knotanchor#1#2{%
2916     \anchor{#2 north west}{%
2917         \csname pgf@anchor@knot #1@north west\endcsname%
2918         \pgf@x=#2\pgf@x%
2919         \pgf@y=#2\pgf@y%
2920     }%
2921     \anchor{#2 north east}{%
2922         \csname pgf@anchor@knot #1@north east\endcsname%
2923         \pgf@x=#2\pgf@x%
2924         \pgf@y=#2\pgf@y%
2925     }%
2926     \anchor{#2 south west}{%
2927         \csname pgf@anchor@knot #1@south west\endcsname%
2928         \pgf@x=#2\pgf@x%
2929         \pgf@y=#2\pgf@y%
2930     }%
2931     \anchor{#2 south east}{%
2932         \csname pgf@anchor@knot #1@south east\endcsname%
2933         \pgf@x=#2\pgf@x%
2934         \pgf@y=#2\pgf@y%
2935     }%
2936     \anchor{#2 north}{%
2937         \csname pgf@anchor@knot #1@north\endcsname%

```

```

2938   \pgf@x=#2\pgf@x%
2939   \pgf@y=#2\pgf@y%
2940 }%
2941 \anchor{#2 east}{%
2942   \csname pgf@anchor@knot #1@east\endcsname%
2943   \pgf@x=#2\pgf@x%
2944   \pgf@y=#2\pgf@y%
2945 }%
2946 \anchor{#2 west}{%
2947   \csname pgf@anchor@knot #1@west\endcsname%
2948   \pgf@x=#2\pgf@x%
2949   \pgf@y=#2\pgf@y%
2950 }%
2951 \anchor{#2 south}{%
2952   \csname pgf@anchor@knot #1@south\endcsname%
2953   \pgf@x=#2\pgf@x%
2954   \pgf@y=#2\pgf@y%
2955 }%
2956 }

```

(End definition for \pgf@sh@@knotanchor. This function is documented on page ??.)

#### knot crossing

```

2957 \pgfdeclareshape{knot crossing}
2958 {
2959   \inheritsavedanchors[from=circle] % this is nearly a circle
2960   \inheritanchorborder[from=circle]
2961   \inheritanchor[from=circle]{north}
2962   \inheritanchor[from=circle]{north west}
2963   \inheritanchor[from=circle]{north east}
2964   \inheritanchor[from=circle]{center}
2965   \inheritanchor[from=circle]{west}
2966   \inheritanchor[from=circle]{east}
2967   \inheritanchor[from=circle]{mid}
2968   \inheritanchor[from=circle]{mid west}
2969   \inheritanchor[from=circle]{mid east}
2970   \inheritanchor[from=circle]{base}
2971   \inheritanchor[from=circle]{base west}
2972   \inheritanchor[from=circle]{base east}
2973   \inheritanchor[from=circle]{south}
2974   \inheritanchor[from=circle]{south west}
2975   \inheritanchor[from=circle]{south east}
2976   \inheritanchorborder[from=circle]
2977   \pgf@sh@@knotanchor{crossing}{2}
2978   \pgf@sh@@knotanchor{crossing}{3}
2979   \pgf@sh@@knotanchor{crossing}{4}
2980   \pgf@sh@@knotanchor{crossing}{8}
2981   \pgf@sh@@knotanchor{crossing}{16}
2982   \pgf@sh@@knotanchor{crossing}{32}
2983   \backgroundpath{

```

```

2984 \pgfutil@tempdima=\radius%
2985 \pgfmathsetlength{\pgf@xb}{\pgfkeysvalueof{/pgf/outer xsep}}%
2986 \pgfmathsetlength{\pgf@yb}{\pgfkeysvalueof{/pgf/outer ysep}}%
2987 \ifdim\pgf@xb<\pgf@yb%
2988   \advance\pgfutil@tempdima by-\pgf@yb%
2989 \else%
2990   \advance\pgfutil@tempdima by-\pgf@xb%
2991 \fi%
2992 }
2993 }

```

(End definition for knot crossing. This function is documented on page ??.)

knot over cross

```

2994 \pgfdeclareshape{knot over cross}
2995 {
2996   \inheritsavedanchors[from=rectangle] % this is nearly a circle
2997   \inheritanchorborder[from=rectangle]
2998   \inheritanchor[from=rectangle]{north}
2999   \inheritanchor[from=rectangle]{north west}
3000   \inheritanchor[from=rectangle]{north east}
3001   \inheritanchor[from=rectangle]{center}
3002   \inheritanchor[from=rectangle]{west}
3003   \inheritanchor[from=rectangle]{east}
3004   \inheritanchor[from=rectangle]{mid}
3005   \inheritanchor[from=rectangle]{mid west}
3006   \inheritanchor[from=rectangle]{mid east}
3007   \inheritanchor[from=rectangle]{base}
3008   \inheritanchor[from=rectangle]{base west}
3009   \inheritanchor[from=rectangle]{base east}
3010   \inheritanchor[from=rectangle]{south}
3011   \inheritanchor[from=rectangle]{south west}
3012   \inheritanchor[from=rectangle]{south east}
3013   \inheritanchorborder[from=rectangle]
3014   \backgroundpath{
3015     \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
3016     \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
3017     \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
3018     \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
3019   }
3020   \foregroundpath{
3021     % store lower right in xa/ya and upper right in xb/yb
3022     \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
3023     \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
3024     \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
3025     \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
3026   }
3027 }

```

(End definition for knot over cross. This function is documented on page ??.)



knot under cross

```
3028 \pgfdeclareshape{knot under cross}
3029 {
3030   \inheritsavedanchors[from=rectangle] % this is nearly a circle
3031   \inheritanchorborder[from=rectangle]
3032   \inheritanchor[from=rectangle]{north}
3033   \inheritanchor[from=rectangle]{north west}
3034   \inheritanchor[from=rectangle]{north east}
3035   \inheritanchor[from=rectangle]{center}
3036   \inheritanchor[from=rectangle]{west}
3037   \inheritanchor[from=rectangle]{east}
3038   \inheritanchor[from=rectangle]{mid}
3039   \inheritanchor[from=rectangle]{mid west}
3040   \inheritanchor[from=rectangle]{mid east}
3041   \inheritanchor[from=rectangle]{base}
3042   \inheritanchor[from=rectangle]{base west}
3043   \inheritanchor[from=rectangle]{base east}
3044   \inheritanchor[from=rectangle]{south}
3045   \inheritanchor[from=rectangle]{south west}
3046   \inheritanchor[from=rectangle]{south east}
3047   \inheritanchorborder[from=rectangle]
3048   \backgroundpath{
3049     \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
3050     \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
3051     \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
3052     \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
3053   }
3054   \foregroundpath{
3055     % store lower right in xa/ya and upper right in xb/yb
3056     \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
3057     \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
3058     \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
3059     \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
3060   }
3061 }
```

*(End definition for knot under cross. This function is documented on page ??.)*

knot vert

```
3062 \pgfdeclareshape{knot vert}
3063 {
3064   \inheritsavedanchors[from=rectangle] % this is nearly a circle
3065   \inheritanchorborder[from=rectangle]
3066   \inheritanchor[from=rectangle]{north}
3067   \inheritanchor[from=rectangle]{north west}
3068   \inheritanchor[from=rectangle]{north east}
3069   \inheritanchor[from=rectangle]{center}
3070   \inheritanchor[from=rectangle]{west}
3071   \inheritanchor[from=rectangle]{east}
3072   \inheritanchor[from=rectangle]{mid}
```

```

3073 \inheritanchor[from=rectangle]{mid west}
3074 \inheritanchor[from=rectangle]{mid east}
3075 \inheritanchor[from=rectangle]{base}
3076 \inheritanchor[from=rectangle]{base west}
3077 \inheritanchor[from=rectangle]{base east}
3078 \inheritanchor[from=rectangle]{south}
3079 \inheritanchor[from=rectangle]{south west}
3080 \inheritanchor[from=rectangle]{south east}
3081 \inheritanchorborder[from=rectangle]
3082 \backgroundpath{
3083 % store lower right in xa/ya and upper right in xb/yb
3084 \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
3085 \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
3086 \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
3087 \pgfpathlineto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
3088 \pgfpathmoveto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
3089 \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
3090 }
3091 }

```

*(End definition for knot vert. This function is documented on page ??.)*

**knot horiz**

```

3092 \pgfdeclareshape{knot horiz}
3093 {
3094 \inheritssavedanchors[from=rectangle] % this is nearly a circle
3095 \inheritanchorborder[from=rectangle]
3096 \inheritanchor[from=rectangle]{north}
3097 \inheritanchor[from=rectangle]{north west}
3098 \inheritanchor[from=rectangle]{north east}
3099 \inheritanchor[from=rectangle]{center}
3100 \inheritanchor[from=rectangle]{west}
3101 \inheritanchor[from=rectangle]{east}
3102 \inheritanchor[from=rectangle]{mid}
3103 \inheritanchor[from=rectangle]{mid west}
3104 \inheritanchor[from=rectangle]{mid east}
3105 \inheritanchor[from=rectangle]{base}
3106 \inheritanchor[from=rectangle]{base west}
3107 \inheritanchor[from=rectangle]{base east}
3108 \inheritanchor[from=rectangle]{south}
3109 \inheritanchor[from=rectangle]{south west}
3110 \inheritanchor[from=rectangle]{south east}
3111 \inheritanchorborder[from=rectangle]
3112 \foregroundpath{
3113 % store lower right in xa/ya and upper right in xb/yb
3114 \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
3115 \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
3116 \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
3117 \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
3118 \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}

```

```
3119 \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
3120 }
3121 }
```

*(End definition for knot horiz. This function is documented on page ??.)*