

The `spath3` package

Andrew Stacey

stacey@math.ntnu.no

v1.2 from 2019/02/12

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} {2019/02/12} {1.2} {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_tmpe_dim
13 \dim_new:N \l__spath_tmpe_dim
14 \fp_new:N \l__spath_tmpe_fp
```

```

15 \fp_new:N \l__spath_tmpb_fp
16 \int_new:N \l__spath_tmpa_int
17 \int_new:N \g__spath_map_int

```

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

```

18 \tl_new:N \g__spath_moveto_tl
19 \tl_new:N \g__spath_lineto_tl
20 \tl_new:N \g__spath_curveto_tl
21 \tl_new:N \g__spath_curvetoa_tl
22 \tl_new:N \g__spath_curvetob_tl
23 \tl_new:N \g__spath_closepath_tl
24 \tl_set:Nn \g__spath_moveto_tl {\pgfsyssoftpath@movetotoken}
25 \tl_set:Nn \g__spath_lineto_tl {\pgfsyssoftpath@linetotoken}
26 \tl_set:Nn \g__spath_curveto_tl {\pgfsyssoftpath@curvetotoken}
27 \tl_set:Nn \g__spath_curvetoa_tl {\pgfsyssoftpath@curvetosupportatoken}
28 \tl_set:Nn \g__spath_curvetob_tl {\pgfsyssoftpath@curvetosupportbtoken}
29 \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.

```

30 \tl_new:N \g__spath_attributes
31 \tl_new:N \g__spath_moveable_attributes
32 \tl_new:N \g__spath_path_attributes
33 \tl_set:Nn \g__spath_attributes {
34   {path}
35   {reverse path}
36   {length}
37   {real length}
38   {number of components}
39   {initial point}
40   {final point}
41   {initial action}
42   {final action}
43   {min bb}
44   {max bb}
45 }
46 \tl_set:Nn \g__spath_moveable_attributes {
47   {initial point}
48   {final point}
49   {min bb}
50   {max bb}
51 }
52 \tl_set:Nn \g__spath_path_attributes {
53   {path}
54   {reverse path}
55 }

```

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`

```
56 \cs_new_nopar:Npn \spath_new:n #1
57 {
58   \prop_new:c {l__spath_#1}
59 }
```

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

`\spath_clear:n`

```
60 \cs_new_nopar:Npn \spath_clear:n #1
61 {
62   \prop_gclear:c {l__spath_#1}
63 }
```

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

`\spath_clear_new:n`

```
64 \cs_new_nopar:Npn \spath_clear_new:n #1
65 {
66   \prop_gclear_new:c {l__spath_#1}
67 }
```

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

`\spath_show:n`

```
68 \cs_new_nopar:Npn \spath_show:n #1
69 {
70   \prop_show:c {l__spath_#1}
71 }
```

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

`\spath_put:nnn`

```
72 \cs_new_nopar:Npn \spath_put:nnn #1#2#3
73 {
74   \prop_gput:cnn {l__spath_#1} {#2} {#3}
75 }
```

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

`\spath_remove:nn`

```
76 \cs_new_nopar:Npn \spath_remove:nn #1#2
77 {
78   \prop_gremove:cn {l__spath_#1} {#2}
79 }
```

(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.

```

80 \cs_new_nopar:Npn \_spath_get:nn #1#2
81 {
82   \prop_item:cn {l\_spath_#1} {#2}
83 }

```

(End definition for _spath_get:nn.)

`_spath_get:nnN`

```

84 \cs_new_nopar:Npn \_spath_get:nnN #1#2#3
85 {
86   \prop_get:cnN {l\_spath_#1} {#2} #3
87 }

```

(End definition for _spath_get:nnN.)

`\spath_if_in:nn`

```

88 \prg_new_conditional:Npnn \spath_if_in:nn #1#2 {p, T, F, TF}
89 {
90   \prop_if_in:cnTF {l\_spath_#1} {#2}
91   { \prg_return_true: }
92   { \prg_return_false: }
93 }

```

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

`_spath_get:nnN`

```

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

```

(End definition for _spath_get:nnN.)

`\spath_if_exist:n`

```

107 \prg_new_conditional:Npnn \spath_if_exist:n #1 {p,T,F,TF}
108 {
109   \prop_if_exist:cTF {l\_spath_#1}
110   {
111     \prg_return_true:
112   }
113   {
114     \prg_return_false:
115   }
116 }

```

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

`\spath_clone:nn` Clones an spath.

```
117 \cs_new_nopar:Npn \spath_clone:nn #1 #2
118 {
119   \spath_clear_new:n {#2}
120   \tl_map_inline:Nn \g__spath_attributes
121   {
122     \spath_if_in:nnT {#1} {##1}
123     {
124       \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
125       \spath_put:nnV {#2} {##1} \l__spath_tmpa_tl
126     }
127   }
128 }
```

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

`\spath_get_current_path:n`

```
129 \cs_new_protected_nopar:Npn \spath_get_current_path:n #1
130 {
131   \pgfsyssoftpath@getcurrentpath\l__spath_tmpa_tl
132   \spath_clear_new:n {#1}
133   \spath_put:nnV {#1} {path} \l__spath_tmpa_tl
134 }
```

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

`\spath_set_current_path:n`

```
135 \cs_new_protected_nopar:Npn \spath_set_current_path:n #1
136 {
137   \spath_get:nnN {#1} {min bb} \l__spath_tmpa_tl
138   \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl
139
140   \spath_get:nnN {#1} {max bb} \l__spath_tmpa_tl
141   \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl
142
143   \spath_get:nnN {#1} {path} \l__spath_tmpa_tl
144   \pgfsyssoftpath@setcurrentpath\l__spath_tmpa_tl
145   \pgfsyssoftpath@flushcurrentpath
146 }
```

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

`\spath_use_path:nn`

```
147 \cs_new_protected_nopar:Npn \spath_use_path:nn #1#2
148 {
149   \spath_set_current_path:n {#1}
150   \pgfusepath{#2}
151 }
```

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

`\spath_protocol_path:n`

```
152 \cs_new_protected_nopar:Npn \spath_protocol_path:n #1
153 {
154   \spath_get:nnN {#1} {min bb} \l__spath_tmpa_tl
155   \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl
156
157   \spath_get:nnN {#1} {max bb} \l__spath_tmpa_tl
158   \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl
159 }
```

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

`\spath_tikz_path:nn`

```
160 \cs_new_protected_nopar:Npn \spath_tikz_path:nn #1#2
161 {
162   \path[#1] \pgfextra{
163     \spath_get:nnN {#2} {path} \l__spath_tmpa_tl
164     \pgfsyssoftpath@setcurrentpath \l__spath_tmpa_tl
165   };
166 }
167 \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.

```
168 \cs_new_nopar:Npn \spath_get:nn #1#2
169 {
170   \spath_if_in:nnF {#1} {#2}
171   {
172     \cs_if_exist_use:cT {spath_generate_#2:n} {{{#1}}}
173   }
174   \__spath_get:nn {#1} {#2}
175 }
```

(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.

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

(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.

```

185 \cs_new_nopar:Npn \spath_generate_length:n #1
186 {
187   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
188   \spath_put:nx {#1} {length} {\int_eval:n {\tl_count:N \l__spath_tmpa_tl /3 }}
189 }

```

(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).

```

190 \cs_new_nopar:Npn \spath_generate_reallength:n #1
191 {
192   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
193   \int_set:Nn \l__spath_tmpa_int {0}
194   \tl_map_inline:Nn \l__spath_tmpa_tl {
195     \tl_if_eq:nnT {##1} {\pgfsyssoftpath@linetotoken}
196     {
197       \int_incr:N \l__spath_tmpa_int
198     }
199     \tl_if_eq:nnT {##1} {\pgfsyssoftpath@curvetotoken}
200     {
201       \int_incr:N \l__spath_tmpa_int
202     }
203   }
204   \spath_put:nx {#1} {real length} {\int_use:N \l__spath_tmpa_int}
205 }

```

(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.

```

206 \cs_new_nopar:Npn \spath_generate_numberofcomponents:n #1
207 {
208   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
209   \int_set:Nn \l__spath_tmpa_int {0}
210   \tl_map_inline:Nn \l__spath_tmpa_tl {
211     \tl_if_eq:nnT {##1} {\pgfsyssoftpath@movetotoken}
212     {
213       \int_incr:N \l__spath_tmpa_int
214     }
215   }
216   \spath_put:nx {#1} {number of components} {\int_use:N \l__spath_tmpa_int}
217 }

```

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

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

```

218 \cs_new_nopar:Npn \spath_generate_initialpoint:n #1
219 {
220   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
221   \tl_clear:N \l__spath_tmpb_tl
222   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
223   \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}

```

```

224 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
225 \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
226 \spath_put:nnV {#1} {initial point} \l__spath_tmpb_tl
227 }

```

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

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

```

228 \cs_new_nopar:Npn \spath_generate_finalpoint:n #1
229 {
230   \tl_clear:N \l__spath_tmpb_tl
231   \spath_if_in:nnTF {#1} {reverse path}
232   {
233     \__spath_get:nnN {#1} {reverse path} \l__spath_tmpa_tl
234     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
235     \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
236     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
237     \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
238   }
239   {
240     \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
241     \tl_reverse:N \l__spath_tmpa_tl
242     \tl_put_left:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
243     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
244     \tl_put_left:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
245   }
246   \spath_put:nnV {#1} {final point} \l__spath_tmpb_tl
247 }
248 \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.

```

249 \cs_new_nopar:Npn \spath_generate_reversepath:n #1
250 {
251   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
252   \tl_clear:N \l__spath_tmpb_tl
253   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
254   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
255   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
256   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
257   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
258   \tl_put_left:Nx \l__spath_tmpb_tl
259   {
260     {\dim_use:N \l__spath_tmpa_dim}
261     {\dim_use:N \l__spath_tmpb_dim}
262   }
263   \bool_until_do:nn {
264     \tl_if_empty_p:N \l__spath_tmpa_tl
265   }
266   {
267     \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
268     \tl_set:Nn \l__spath_tmpd_tl {}
269     \tl_case:NnF \l__spath_tmpc_tl
270     {

```



```

271     \g__spath_moveto_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_moveto_tl }
272     \g__spath_lineto_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_lineto_tl }
273     \g__spath_curveto_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curvetoa_tl }
274     \g__spath_curvetoa_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curveto_tl }
275     \g__spath_curvetob_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curvetob_tl }
276   }
277   {
278     \tl_show:N \l__spath_tmpc_tl
279   }
280   \tl_put_left:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
281   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
282   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
283   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
284   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
285   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
286   \tl_put_left:Nx \l__spath_tmpb_tl
287   {
288     {\dim_use:N \l__spath_tmpa_dim}
289     {\dim_use:N \l__spath_tmpb_dim}
290   }
291 }
292 \tl_put_left:NV \l__spath_tmpb_tl \g__spath_moveto_tl
293 \spath_put:nnV {#1} {reverse path} \l__spath_tmpb_tl
294 }

```

(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).

```

295 \cs_new_nopar:Npn \spath_generate_initialaction:n #1
296 {
297   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
298   \tl_clear:N \l__spath_tmpb_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_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
302   \tl_if_empty:NF \l__spath_tmpa_tl {
303     \tl_set:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
304   }
305   \spath_put:nnV {#1} {initial action} \l__spath_tmpb_tl
306 }

```

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

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

```

307 \cs_new_nopar:Npn \spath_generate_finalaction:n #1
308 {
309   \tl_clear:N \l__spath_tmpb_tl
310   \spath_if_in:nnTF {#1} {reverse path}
311   {
312     \__spath_get:nnN {#1} {reverse path} \l__spath_tmpa_tl
313     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
314   }
315   {
316     \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
317     \tl_reverse:N \l__spath_tmpa_tl

```

```

318 }
319 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
320 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
321 \tl_if_empty:NF \l__spath_tmpa_tl {
322   \tl_set:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
323 }
324 \tl_if_eq:NNT \l__spath_tmpa_tl \g__spath_curvetoa_tl
325 {
326   \tl_set_eq:NN \l__spath_tmpa_tl \g__spath_curveto_tl
327 }
328 \spath_put:nnV {#1} {final action} \l__spath_tmpb_tl
329 }

```

(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.

```

330 \cs_new_nopar:Npn \spath_generate_minbb:n #1
331 {
332   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
333   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
334   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
335   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
336   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
337   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
338   \bool_until_do:nn {
339     \tl_if_empty_p:N \l__spath_tmpa_tl
340   }
341   {
342     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
343     \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tm
344     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
345     \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tm
346     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
347   }
348   \tl_clear:N \l__spath_tmpb_tl
349   \tl_put_right:Nx \l__spath_tmpb_tl
350   {
351     {\dim_use:N \l__spath_tmpa_dim}
352     {\dim_use:N \l__spath_tmpb_dim}
353   }
354   \spath_put:nnV {#1} {min bb} \l__spath_tmpb_tl
355 }

```

(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.

```

356 \cs_new_nopar:Npn \spath_generate_maxbb:n #1
357 {
358   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
359   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
360   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
361   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
362   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
363   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
364   \bool_until_do:nn {

```

```

365 \tl_if_empty_p:N \l__spath_tmpa_tl
366 }
367 {
368 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
369 \dim_set:Nn \l__spath_tmpa_dim {\dim_max:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tm
370 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
371 \dim_set:Nn \l__spath_tmpb_dim {\dim_max:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tm
372 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
373 }
374 \tl_clear:N \l__spath_tmpb_tl
375 \tl_put_right:Nx \l__spath_tmpb_tl
376 {
377 {\dim_use:N \l__spath_tmpa_dim}
378 {\dim_use:N \l__spath_tmpb_dim}
379 }
380 \spath_put:nnV {#1} {max bb} \l__spath_tmpb_tl
381 }

```

(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

```

382 \tl_new:N \l__spath_rp_tl
383 \int_new:N \l__spath_l_int
384 \int_new:N \l__spath_rl_int
385 \int_new:N \l__spath_nc_int
386 \tl_new:N \l__spath_ip_tl
387 \tl_new:N \l__spath_fp_tl
388 \tl_new:N \l__spath_ia_tl
389 \tl_new:N \l__spath_fa_tl
390 \dim_new:N \l__spath_minx_dim

```

```

391 \dim_new:N \l__spath_miny_dim
392 \dim_new:N \l__spath_maxx_dim
393 \dim_new:N \l__spath_maxy_dim
394
395 \cs_new_nopar:Npn \spath_generate_all:n #1
396 {
397   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
398
399   \tl_clear:N \l__spath_rp_tl
400   \int_set:Nn \l__spath_l_int {1}
401   \int_zero:N \l__spath_rl_int
402   \int_set:Nn \l__spath_nc_int {1}
403   \tl_clear:N \l__spath_ip_tl
404   \tl_clear:N \l__spath_fp_tl
405   \tl_clear:N \l__spath_ia_tl
406   \tl_clear:N \l__spath_fa_tl
407   \dim_zero:N \l__spath_minx_dim
408   \dim_zero:N \l__spath_miny_dim
409   \dim_zero:N \l__spath_maxx_dim
410   \dim_zero:N \l__spath_maxy_dim
411
412   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
413   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
414   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
415   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
416   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
417
418   \tl_clear:N \l__spath_ip_tl
419   \tl_put_right:Nx \l__spath_ip_tl
420   {
421     {\dim_use:N \l__spath_tmpa_dim}
422     {\dim_use:N \l__spath_tmpb_dim}
423   }
424
425   \dim_set_eq:NN \l__spath_minx_dim \l__spath_tmpa_dim
426   \dim_set_eq:NN \l__spath_miny_dim \l__spath_tmpb_dim
427   \dim_set_eq:NN \l__spath_maxx_dim \l__spath_tmpa_dim
428   \dim_set_eq:NN \l__spath_maxy_dim \l__spath_tmpb_dim
429
430   \tl_put_left:Nx \l__spath_rp_tl
431   {
432     {\dim_use:N \l__spath_tmpa_dim}
433     {\dim_use:N \l__spath_tmpb_dim}
434   }
435
436   \tl_set:Nx \l__spath_ia_tl {\tl_head:N \l__spath_tmpa_tl}
437
438   \bool_until_do:nn {
439     \tl_if_empty_p:N \l__spath_tmpa_tl
440   }
441   {
442     \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
443     \tl_set:Nn \l__spath_tmpd_tl {}
444     \tl_set_eq:NN \l__spath_fa_tl \l__spath_tmpc_tl

```

```

445 \int_incr:N \l__spath_l_int
446
447 \tl_case:NnF \l__spath_tmpc_tl
448 {
449   \g__spath_moveto_tl {
450     \tl_set_eq:NN \l__spath_tmpd_tl \g__spath_moveto_tl
451     \int_incr:N \l__spath_nc_int
452   }
453   \g__spath_lineto_tl {
454     \tl_set_eq:NN \l__spath_tmpd_tl \g__spath_lineto_tl
455     \int_incr:N \l__spath_rl_int
456   }
457   \g__spath_curveto_tl {
458     \tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curvetoa_tl
459     \int_incr:N \l__spath_rl_int
460   }
461   \g__spath_curvetoa_tl {
462     \tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curveto_tl
463   }
464   \g__spath_curvetob_tl {
465     \tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curvetob_tl
466   }
467 }
468 {
469   \tl_show:N \l__spath_tmpc_tl
470 }
471 \tl_put_left:NV \l__spath_rp_tl \l__spath_tmpd_tl
472 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
473
474 \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
475 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
476 \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
477 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
478
479 \dim_set:Nn \l__spath_minx_dim {\dim_min:nn { \l__spath_minx_dim} {\l__spath_tmpa_dim}}
480 \dim_set:Nn \l__spath_miny_dim {\dim_min:nn { \l__spath_miny_dim} {\l__spath_tmpb_dim}}
481 \dim_set:Nn \l__spath_maxx_dim {\dim_max:nn { \l__spath_maxx_dim} {\l__spath_tmpa_dim}}
482 \dim_set:Nn \l__spath_maxy_dim {\dim_max:nn { \l__spath_maxy_dim} {\l__spath_tmpb_dim}}
483
484 \tl_put_left:Nx \l__spath_rp_tl
485 {
486   {\dim_use:N \l__spath_tmpa_dim}
487   {\dim_use:N \l__spath_tmpb_dim}
488 }
489
490 \tl_set:Nx \l__spath_fp_tl
491 {
492   {\dim_use:N \l__spath_tmpa_dim}
493   {\dim_use:N \l__spath_tmpb_dim}
494 }
495
496 }
497
498 \tl_put_left:NV \l__spath_rp_tl \g__spath_moveto_tl

```

```

499
500 \spath_put:nnV {#1} {reverse path} \l__spath_rp_tl
501 \spath_put:nnV {#1} {length} \l__spath_l_int
502 \spath_put:nnV {#1} {real length} \l__spath_rl_int
503 \spath_put:nnV {#1} {number of components} \l__spath_nc_int
504 \spath_put:nnV {#1} {initial point} \l__spath_ip_tl
505 \spath_put:nnV {#1} {final point} \l__spath_fp_tl
506 \spath_put:nnV {#1} {initial action} \l__spath_ia_tl
507 \spath_put:nnV {#1} {final action} \l__spath_fa_tl
508
509 \tl_clear:N \l__spath_tmpb_tl
510 \tl_put_right:Nx \l__spath_tmpb_tl
511 {
512   {\dim_use:N \l__spath_minx_dim}
513   {\dim_use:N \l__spath_miny_dim}
514 }
515 \spath_put:nnV {#1} {min bb} \l__spath_tmpb_tl
516
517 \tl_clear:N \l__spath_tmpb_tl
518 \tl_put_right:Nx \l__spath_tmpb_tl
519 {
520   {\dim_use:N \l__spath_maxx_dim}
521   {\dim_use:N \l__spath_maxy_dim}
522 }
523 \spath_put:nnV {#1} {max bb} \l__spath_tmpb_tl
524
525 }

```

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

2.4 Path Manipulation

`\spath_translate:nnm` Translates a path.

```

526 \cs_new_nopar:Npn \spath_translate:nnm #1#2#3
527 {
528   \tl_map_inline:Nn \g__spath_moveable_attributes
529   {
530     \spath_if_in:nnT {#1} {##1}
531     {
532       \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
533
534       \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl + #2}
535       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
536       \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl + #3}
537       \tl_clear:N \l__spath_tmpb_tl
538       \tl_put_right:Nx \l__spath_tmpb_tl
539       {
540         {\dim_use:N \l__spath_tmpa_dim}
541         {\dim_use:N \l__spath_tmpb_dim}
542       }
543       \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
544     }
545   }
546   \tl_map_inline:Nn \g__spath_path_attributes

```

```

547 {
548   \spath_if_in:nnT {#1} {##1}
549   {
550     \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
551     \tl_clear:N \l__spath_tmpb_tl
552     \bool_until_do:nn {
553       \tl_if_empty_p:N \l__spath_tmpa_tl
554     }
555     {
556       \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
557       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
558
559       \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl + #2}
560       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
561
562       \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl + #3}
563       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
564
565       \tl_put_right:Nx \l__spath_tmpb_tl
566       {
567         {\dim_use:N \l__spath_tmpa_dim}
568         {\dim_use:N \l__spath_tmpb_dim}
569       }
570     }
571     \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
572   }
573 }
574 }
575
576 \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.

```

577 \cs_new_nopar:Npn \spath_translate:nn #1#2
578 {
579   \spath_translate:nnn {#1} #2
580 }
581
582 \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.

```

583 \cs_new_nopar:Npn \spath_scale:nnn #1#2#3
584 {
585   \tl_map_inline:Nn \g__spath_moveable_attributes
586   {
587     \spath_if_in:nnT {#1} {##1}
588     {
589       \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
590
591       \fp_set:Nn \l__spath_tmpa_fp {\tl_head:N \l__spath_tmpa_tl * #2}
592       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
593       \fp_set:Nn \l__spath_tmpb_fp {\tl_head:N \l__spath_tmpa_tl * #3}
594       \tl_clear:N \l__spath_tmpb_tl

```

```

595     \tl_put_right:Nx \l__spath_tmpb_tl
596     {
597       {\fp_to_dim:N \l__spath_tmpa_fp}
598       {\fp_to_dim:N \l__spath_tmpb_fp}
599     }
600     \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
601   }
602 }
603 \tl_map_inline:Nn \g__spath_path_attributes
604 {
605   \spath_if_in:nnT {#1} {##1}
606   {
607     \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
608     \tl_clear:N \l__spath_tmpb_tl
609     \bool_until_do:nn {
610       \tl_if_empty_p:N \l__spath_tmpa_tl
611     }
612     {
613       \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
614       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
615
616       \fp_set:Nn \l__spath_tmpa_fp {\tl_head:N \l__spath_tmpa_tl * #2}
617       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
618
619       \fp_set:Nn \l__spath_tmpb_fp {\tl_head:N \l__spath_tmpa_tl * #3}
620       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
621
622       \tl_put_right:Nx \l__spath_tmpb_tl
623       {
624         {\fp_to_dim:N \l__spath_tmpa_fp}
625         {\fp_to_dim:N \l__spath_tmpb_fp}
626       }
627     }
628     \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
629   }
630 }
631 }
632 \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.

```

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

```

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

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

```

639 \cs_new_nopar:Npn \spath_transform:nnnnnn #1#2#3#4#5#6#7
640 {
641   \tl_map_inline:Nn \g__spath_moveable_attributes

```



```

642 {
643   \spath_if_in:nnT {#1} {##1}
644   {
645     \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
646     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpa_tl}
647     \tl_set:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpb_tl}
648     \tl_set:Nx \l__spath_tmpa_tl {\tl_head:N \l__spath_tmpa_tl}
649     \fp_set:Nn \l__spath_tmpa_fp {\l__spath_tmpa_tl * #2 + \l__spath_tmpb_tl * #3 + #6}
650     \fp_set:Nn \l__spath_tmpb_fp {\l__spath_tmpa_tl * #4 + \l__spath_tmpb_tl * #5 + #7}
651     \tl_clear:N \l__spath_tmpb_tl
652     \tl_put_right:Nx \l__spath_tmpb_tl
653     {
654       {\fp_to_dim:N \l__spath_tmpa_fp}
655       {\fp_to_dim:N \l__spath_tmpb_fp}
656     }
657     \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
658   }
659 }
660 \tl_map_inline:Nn \g__spath_path_attributes
661 {
662   \spath_if_in:nnT {#1} {##1}
663   {
664     \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
665     \tl_clear:N \l__spath_tmpb_tl
666     \bool_until_do:nn {
667       \tl_if_empty_p:N \l__spath_tmpa_tl
668     }
669     {
670       \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
671       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
672       \tl_set:Nx \l_tmpa_tl {\tl_head:N \l__spath_tmpa_tl}
673       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
674       \tl_set:Nx \l_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
675       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
676
677       \fp_set:Nn \l__spath_tmpa_fp {\l_tmpa_tl * #2 + \l_tmpb_tl * #3 + #6}
678       \fp_set:Nn \l__spath_tmpb_fp {\l_tmpa_tl * #4 + \l_tmpb_tl * #5 + #7}
679       \tl_put_right:Nx \l__spath_tmpb_tl
680       {
681         {\fp_to_dim:N \l__spath_tmpa_fp}
682         {\fp_to_dim:N \l__spath_tmpb_fp}
683       }
684     }
685     \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
686   }
687 }
688 }
689
690 \cs_generate_variant:Nn \spath_transform:nnnnnn {nxxxxxx}

```

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

```

691 \cs_new_nopar:Npn \spath_transform:nn #1#2
692 {

```

```

693 \spath_transform:nnnnnnn {#1} #2
694 }
695
696 \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.

```

697 \cs_new_nopar:Npn \spath_reverse:n #1
698 {
699   \spath_if_in:nnF {#1} {reverse path} {
700     \use:c {spath_generate_reverse_path:n} {#1}
701   }
702   \spath_swap:nnn {#1} {path} {reverse path}
703   \spath_swap:nnn {#1} {initial point} {final point}
704   \spath_swap:nnn {#1} {initial action} {final action}
705 }

```

(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.

```

706 \cs_new_nopar:Npn \spath_swap:nnn #1#2#3
707 {
708   \__spath_get:nnNF {#1} {#2} \l__spath_tmpa_tl {\tl_clear:N \l__spath_tmpa_tl}
709   \__spath_get:nnNF {#1} {#3} \l__spath_tmpb_tl {\tl_clear:N \l__spath_tmpb_tl}
710   \tl_if_empty:NTF \l__spath_tmpb_tl
711   {\spath_remove:nn {#1} {#2}}
712   {\spath_put:nnV {#1} {#2} \l__spath_tmpb_tl}
713   \tl_if_empty:NTF \l__spath_tmpa_tl
714   {\spath_remove:nn {#1} {#3}}
715   {\spath_put:nnV {#1} {#3} \l__spath_tmpa_tl}
716 }

```

(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.

```

717 \cs_new_nopar:Npn \spath_weld:nn #1#2
718 {
719   \spath_clone:nn {#2} {tmp_path}
720   \spath_get:nnN {#1} {final point} \l__spath_tmpa_tl
721
722   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
723   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
724   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
725
726   \spath_get:nnN {#2} {initial point} \l__spath_tmpa_tl
727
728   \dim_sub:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
729   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
730   \dim_sub:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
731

```

```

732 \spath_translate:nxx {tmp_path} {\dim_use:N \l__spath_tmpa_dim} {\dim_use:N \l__spath_tmpb_dim}
733
734 \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
735 \__spath_get:nnN {tmp_path} {path} \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_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
739 \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
740
741 \spath_put:nnV {#1} {path} \l__spath_tmpa_tl
742
743 \__spath_get:nnNTF {tmp_path} {final point} \l__spath_tmpa_tl
744 {
745   \spath_put:nnV {#1} {final point} \l__spath_tmpa_tl
746 }
747 {
748   \spath_remove:nn {#1} {final point}
749 }
750
751 \__spath_get:nnNTF {tmp_path} {final action} \l__spath_tmpa_tl
752 {
753   \spath_put:nnV {#1} {final action} \l__spath_tmpa_tl
754 }
755 {
756   \spath_remove:nn {#1} {final action}
757 }
758
759 \__spath_get:nnNT {tmp_path} {min bb} \l__spath_tmpa_tl
760 {
761   \__spath_get:nnNT {#1} {min bb} \l__spath_tmpb_tl
762   {
763     \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
764     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
765     \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
766
767     \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\l__spath_tmpa_dim} {\tl_head:N
768     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
769     \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\l__spath_tmpb_dim} {\tl_head:N \l__spath
770
771     \tl_clear:N \l__spath_tmpb_tl
772     \tl_put_right:Nx \l__spath_tmpb_tl
773     {
774       {\dim_use:N \l__spath_tmpa_dim}
775       {\dim_use:N \l__spath_tmpb_dim}
776     }
777     \spath_put:nnV {#1} {min bb} \l__spath_tmpb_tl
778   }
779 }
780
781 \__spath_get:nnNT {tmp_path} {max bb} \l__spath_tmpa_tl
782 {
783   \__spath_get:nnNT {#1} {max bb} \l__spath_tmpb_tl
784   {
785     \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}

```

```

786 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
787 \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
788
789 \dim_set:Nn \l__spath_tmpa_dim {\dim_max:nn {\l__spath_tmpa_dim} {\tl_head:N
790 \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
791 \dim_set:Nn \l__spath_tmpb_dim {\dim_max:nn {\l__spath_tmpb_dim} {\tl_head:N \l__spath
792
793 \tl_clear:N \l__spath_tmpb_tl
794 \tl_put_right:Nx \l__spath_tmpb_tl
795 {
796   {\dim_use:N \l__spath_tmpa_dim}
797   {\dim_use:N \l__spath_tmpb_dim}
798 }
799 \spath_put:nnV {#1} {max bb} \l__spath_tmpb_tl
800 }
801 }
802
803 \__spath_get:nnNT {tmp_path} {reverse path} \l__spath_tmpa_tl
804 {
805   \__spath_get:nnNT {#1} {reverse path} \l__spath_tmpb_tl
806   {
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_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
810     \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
811
812     \spath_put:nnV {#1} {reverse path} \l__spath_tmpa_tl
813   }
814 }
815
816 \__spath_get:nnNT {tmp_path} {length} \l__spath_tmpa_tl
817 {
818   \__spath_get:nnNT {#1} {length} \l__spath_tmpb_tl
819   {
820     \int_set:Nn \l__spath_tmpa_int {\l__spath_tmpa_tl + \l__spath_tmpb_tl - 1}
821     \spath_put:nnV {#1} {length} \l__spath_tmpa_int
822   }
823 }
824
825 \__spath_get:nnNT {tmp_path} {real length} \l__spath_tmpa_tl
826 {
827   \__spath_get:nnNT {#1} {real length} \l__spath_tmpb_tl
828   {
829     \int_set:Nn \l__spath_tmpa_int {\l__spath_tmpa_tl + \l__spath_tmpb_tl}
830     \spath_put:nnV {#1} {real length} \l__spath_tmpa_int
831   }
832 }
833
834 \__spath_get:nnNT {tmp_path} {number of components} \l__spath_tmpa_tl
835 {
836   \__spath_get:nnNT {#1} {number of components} \l__spath_tmpb_tl
837   {
838     \int_set:Nn \l__spath_tmpa_int {\l__spath_tmpa_tl + \l__spath_tmpb_tl - 1}
839     \spath_put:nnV {#1} {number of components} \l__spath_tmpa_int

```

```

840     }
841   }
842
843 }

```

(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.

```

844 \cs_new_nopar:Npn \spath_prepend_no_move:nn #1#2
845 {
846   \spath_if_exist:nT {#2}
847   {
848     \__spath_get:nnN {#2} {path} \l__spath_tmpa_tl
849     \__spath_get:nnN {#1} {path} \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_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
853     \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
854     \spath_put:nnV {#1} {path} \l__spath_tmpa_tl
855
856     \spath_if_in:nnTF {#2} {initial point}
857     {
858       \__spath_get:nnN {#2} {initial point} \l__spath_tmpa_tl
859       \spath_put:nnV {#1} {initial point} \l__spath_tmpa_tl
860     }
861     {
862       \spath_remove:nn {#1} {initial point}
863     }
864
865     \spath_if_in:nnTF {#2} {initial action}
866     {
867       \__spath_get:nnN {#2} {initial action} \l__spath_tmpa_tl
868       \spath_put:nnV {#1} {initial action} \l__spath_tmpa_tl
869     }
870     {
871       \spath_remove:nn {#1} {initial action}
872     }
873
874     \bool_if:nTF
875     {
876       \spath_if_in_p:nn {#1} {length}
877       &&
878       \spath_if_in_p:nn {#2} {length}
879     }
880     {
881       \__spath_get:nnN {#1} {length} \l__spath_tmpa_tl
882       \__spath_get:nnN {#2} {length} \l__spath_tmpb_tl
883       \spath_put:nnx {#1} {length} {\int_eval:n {\l__spath_tmpa_tl +
884         \l__spath_tmpb_tl - 1}}
885     }
886     {
887       \spath_remove:nn {#1} {length}
888     }
889     \bool_if:nTF

```

```

890 {
891   \spath_if_in_p:nn {#1} {real length}
892   &&
893   \spath_if_in_p:nn {#2} {real length}
894 }
895 {
896   \__spath_get:nnN {#1} {real length} \l__spath_tmpa_tl
897   \__spath_get:nnN {#2} {real length} \l__spath_tmpb_tl
898   \spath_put:nx {#1} {real length} {\int_eval:n {\l__spath_tmpa_tl +
899     \l__spath_tmpb_tl }}
900 }
901 {
902   \spath_remove:nn {#1} {real length}
903 }
904 \bool_if:nTF
905 {
906   \spath_if_in_p:nn {#1} {number of components}
907   &&
908   \spath_if_in_p:nn {#2} {number of components}
909 }
910 {
911   \__spath_get:nnN {#1} {number of components} \l__spath_tmpa_tl
912   \__spath_get:nnN {#2} {number of components} \l__spath_tmpb_tl
913   \spath_put:nx {#1} {number of components} {\int_eval:n {\l__spath_tmpa_tl +
914     \l__spath_tmpb_tl - 1}}
915 }
916 {
917   \spath_remove:nn {#1} {number of components}
918 }
919 \bool_if:nTF
920 {
921   \spath_if_in_p:nn {#1} {min bb}
922   &&
923   \spath_if_in_p:nn {#2} {min bb}
924 }
925 {
926   \__spath_get:nnN {#1} {min bb} \l__spath_tmpa_tl
927   \__spath_get:nnN {#2} {min bb} \l__spath_tmpb_tl
928   \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_item:Nn
929     \l__spath_tmpa_tl {1}} {\tl_item:Nn
930     \l__spath_tmpb_tl {1}}}
931   \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_item:Nn
932     \l__spath_tmpa_tl {2}} {\tl_item:Nn
933     \l__spath_tmpb_tl {2}}}
934   \spath_put:nx {#1} {min bb} {
935     {\dim_use:N \l__spath_tmpa_dim}
936     {\dim_use:N \l__spath_tmpb_dim}
937   }
938 }
939 {
940   \spath_remove:nn {#1} {min bb}
941 }
942 \bool_if:nTF
943 {

```

```

944     \spath_if_in_p:nn {#1} {max bb}
945     &&
946     \spath_if_in_p:nn {#2} {max bb}
947   }
948   {
949     \__spath_get:nnN {#1} {max bb} \l__spath_tmpa_tl
950     \__spath_get:nnN {#2} {max bb} \l__spath_tmpb_tl
951     \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_item:Nn
952       \l__spath_tmpa_tl {1}} {\tl_item:Nn
953         \l__spath_tmpb_tl {1}}}
954     \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_item:Nn
955       \l__spath_tmpa_tl {2}} {\tl_item:Nn
956         \l__spath_tmpb_tl {2}}}
957     \spath_put:nxx {#1} {max bb} {
958       {\dim_use:N \l__spath_tmpa_dim}
959       {\dim_use:N \l__spath_tmpb_dim}
960     }
961   }
962   {
963     \spath_remove:nn {#1} {max bb}
964   }
965   \bool_if:nTF
966   {
967     \spath_if_in_p:nn {#1} {reverse path}
968     &&
969     \spath_if_in_p:nn {#2} {reverse path}
970   }
971   {
972     \__spath_get:nnN {#1} {reverse path} \l__spath_tmpa_tl
973     \__spath_get:nnN {#2} {reverse path} \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_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
976     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
977     \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
978     \spath_put:nnV {#1} {reverse path} \l__spath_tmpb_tl
979   }
980   {
981     \spath_remove:nn {#1} {reverse path}
982   }
983 }
984 }
985 }

```

(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.

```

986 \cs_new_nopar:Npn \spath_append_no_move:nn #1#2
987 {
988   \spath_if_exist:nT {#2}
989   {
990     \spath_get:nnN {#1} {path} \l__spath_tmpa_tl
991     \spath_get:nnN {#2} {path} \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_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

```

```

994 \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
995 \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
996 \spath_put:nnV {#1} {path} \l__spath_tmpa_tl
997 \spath_if_in:nnTF {#2} {final point}
998 {
999   \__spath_get:nnN {#2} {final point} \l__spath_tmpa_tl
1000   \spath_put:nnV {#1} {final point} \l__spath_tmpa_tl
1001 }
1002 {
1003   \spath_remove:nn {#1} {final point}
1004 }
1005 \spath_if_in:nnTF {#2} {final action}
1006 {
1007   \__spath_get:nnN {#2} {final action} \l__spath_tmpa_tl
1008   \spath_put:nnV {#1} {final action} \l__spath_tmpa_tl
1009 }
1010 {
1011   \spath_remove:nn {#1} {final action}
1012 }
1013 \bool_if:nTF
1014 {
1015   \spath_if_in_p:nn {#1} {length}
1016   &&
1017   \spath_if_in_p:nn {#2} {length}
1018 }
1019 {
1020   \__spath_get:nnN {#1} {length} \l__spath_tmpa_tl
1021   \__spath_get:nnN {#2} {length} \l__spath_tmpb_tl
1022   \spath_put:nnx {#1} {length} {\int_eval:n {\l__spath_tmpa_tl +
1023     \l__spath_tmpb_tl - 1}}
1024 }
1025 {
1026   \spath_remove:nn {#1} {length}
1027 }
1028 \bool_if:nTF
1029 {
1030   \spath_if_in_p:nn {#1} {real length}
1031   &&
1032   \spath_if_in_p:nn {#2} {real length}
1033 }
1034 {
1035   \__spath_get:nnN {#1} {real length} \l__spath_tmpa_tl
1036   \__spath_get:nnN {#2} {real length} \l__spath_tmpb_tl
1037   \spath_put:nnx {#1} {real length} {\int_eval:n {\l__spath_tmpa_tl +
1038     \l__spath_tmpb_tl }}
1039 }
1040 {
1041   \spath_remove:nn {#1} {real length}
1042 }
1043 \bool_if:nTF
1044 {
1045   \spath_if_in_p:nn {#1} {number of components}
1046   &&
1047   \spath_if_in_p:nn {#2} {number of components}

```



```

1048 }
1049 {
1050   \__spath_get:nnN {#1} {number of components} \l__spath_tmpa_tl
1051   \__spath_get:nnN {#2} {number of components} \l__spath_tmpb_tl
1052   \spath_put:nx {#1} {number of components} {\int_eval:n {\l__spath_tmpa_tl +
1053     \l__spath_tmpb_tl - 1}}
1054 }
1055 {
1056   \spath_remove:nn {#1} {number of components}
1057 }
1058 \bool_if:nTF
1059 {
1060   \spath_if_in_p:nn {#1} {min bb}
1061   &&
1062   \spath_if_in_p:nn {#2} {min bb}
1063 }
1064 {
1065   \__spath_get:nnN {#1} {min bb} \l__spath_tmpa_tl
1066   \__spath_get:nnN {#2} {min bb} \l__spath_tmpb_tl
1067   \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_item:Nn
1068     \l__spath_tmpa_tl {1}} {\tl_item:Nn
1069     \l__spath_tmpb_tl {1}}}
1070   \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_item:Nn
1071     \l__spath_tmpa_tl {2}} {\tl_item:Nn
1072     \l__spath_tmpb_tl {2}}}
1073   \spath_put:nx {#1} {min bb} {
1074     {\dim_use:N \l__spath_tmpa_dim}
1075     {\dim_use:N \l__spath_tmpb_dim}
1076   }
1077 }
1078 {
1079   \spath_remove:nn {#1} {min bb}
1080 }
1081 \bool_if:nTF
1082 {
1083   \spath_if_in_p:nn {#1} {max bb}
1084   &&
1085   \spath_if_in_p:nn {#2} {max bb}
1086 }
1087 {
1088   \__spath_get:nnN {#1} {max bb} \l__spath_tmpa_tl
1089   \__spath_get:nnN {#2} {max bb} \l__spath_tmpb_tl
1090   \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_item:Nn
1091     \l__spath_tmpa_tl {1}} {\tl_item:Nn
1092     \l__spath_tmpb_tl {1}}}
1093   \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_item:Nn
1094     \l__spath_tmpa_tl {2}} {\tl_item:Nn
1095     \l__spath_tmpb_tl {2}}}
1096   \spath_put:nx {#1} {max bb} {
1097     {\dim_use:N \l__spath_tmpa_dim}
1098     {\dim_use:N \l__spath_tmpb_dim}
1099   }
1100 }
1101 {

```

```

1102     \spath_remove:nn {#1} {max bb}
1103   }
1104   \bool_if:nTF
1105   {
1106     \spath_if_in_p:nn {#1} {reverse path}
1107     &&
1108     \spath_if_in_p:nn {#2} {reverse path}
1109   }
1110   {
1111     \__spath_get:nnN {#2} {reverse path} \l__spath_tmpa_tl
1112     \__spath_get:nnN {#1} {reverse path} \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_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
1116     \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
1117     \spath_put:nnV {#1} {reverse path} \l__spath_tmpb_tl
1118   }
1119   {
1120     \spath_remove:nn {#1} {reverse path}
1121   }
1122 }
1123 }

```

(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.

```

1124 \cs_new_nopar:Npn \spath_bake_round:n #1
1125 {
1126   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
1127   \pgf@processround\l__spath_tmpa_tl\l__spath_tmpb_tl
1128   \spath_put:nnV {#1} {path} \l__spath_tmpb_tl
1129 }

```

(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.

```

1130 \cs_new_nopar:Npn \spath_close_path:n #1
1131 {
1132   \spath_get:nnN {#1} {initial point} \l__spath_tmpb_tl
1133   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
1134   \tl_put_right:NV \l__spath_tmpa_tl \g__spath_closepath_tl
1135   \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
1136   \spath_put:nnV {#1} {path} \l__spath_tmpa_tl
1137   \spath_if_in:nnT {#1} {reverse path}
1138   {
1139     \spath_get:nnN {#1} {final point} \l__spath_tmpb_tl
1140     \__spath_get:nnN {#1} {reverse path} \l__spath_tmpa_tl
1141     \tl_put_right:NV \l__spath_tmpa_tl \g__spath_closepath_tl
1142     \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
1143     \spath_put:nnV {#1} {reverse path} \l__spath_tmpa_tl
1144   }
1145 }

```

(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.

```

1146 \cs_new_nopar:Npn \spath_map_component:Nn #1#2
1147 {
1148   \int_gincr:N \g__spath_map_int
1149   \cs_gset:cpn { __spath_map_ \int_use:N \g__spath_map_int :w } ##1 {#2}
1150   \tl_set:NV \l__spath_tmpa_tl #1
1151   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1152   \tl_put_right:NV \l__spath_tmpa_tl \g__spath_moveto_tl
1153   \tl_set_eq:NN \l__spath_tmpb_tl \g__spath_moveto_tl
1154   \bool_until_do:nn {
1155     \tl_if_empty_p:N \l__spath_tmpa_tl
1156   }
1157   {
1158     \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
1159     \tl_if_eq:NNT \l__spath_tmpc_tl \g__spath_moveto_tl
1160     {
1161       \exp_args:NnV \use:c { __spath_map_ \int_use:N \g__spath_map_int :w } \l__spath_tmpb_tl
1162     }
1163     \tl_clear:N \l__spath_tmpb_tl
1164     \tl_if_single:NTF \l__spath_tmpc_tl
1165     {
1166       \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpc_tl
1167     }
1168     {
1169       \tl_put_right:Nx \l__spath_tmpb_tl {\l__spath_tmpc_tl}
1170     }
1171     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1172   }
1173 }

```

(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.

```

1174 \cs_new_nopar:Npn \spath_map_segment_inline:Nn #1#2
1175 {
1176   \int_gincr:N \g__spath_map_int
1177   \cs_gset:cpn { __spath_map_ \int_use:N \g__spath_map_int :w } ##1 ##2 {#2}
1178   \spath_map_segment_function:Nc #1 { __spath_map_ \int_use:N \g__spath_map_int :w }
1179 }

```

(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.

```

1180 \cs_new_nopar:Npn \spath_map_segment_inline:nn #1#2
1181 {
1182   \int_gincr:N \g__spath_map_int
1183   \cs_gset:cpn { __spath_map_ \int_use:N \g__spath_map_int :w } ##1 ##2 {#2}
1184   \spath_get:nnN {#1} {path} \l__spath_tmpd_tl
1185   \spath_map_segment_function:Nc \l__spath_tmpd_tl { __spath_map_ \int_use:N \g__spath_map_int :w }
1186 }

```

(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.

```

1187 \cs_new_nopar:Npn \spath_map_segment_function:nN #1#2
1188 {
1189   \spath_get:nnN {#1} {path} \l__spath_tmpd_tl
1190   \spath_map_segment_function:NN \l__spath_tmpd_tl #2
1191 }

1192 \cs_new_nopar:Npn \spath_map_segment_function:NN #1#2
1193 {
1194   \tl_set_eq:NN \l__spath_tmpa_tl #1
1195   \tl_clear:N \l__spath_tmpb_tl
1196   \dim_zero:N \l__spath_tmpa_dim
1197   \dim_zero:N \l__spath_tmpb_dim
1198
1199   \bool_until_do:nn {
1200     \tl_if_empty_p:N \l__spath_tmpa_tl
1201   }
1202   {
1203     \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
1204     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1205     \tl_case:NnF \l__spath_tmpc_tl
1206     {
1207       \g__spath_lineto_tl
1208       {
1209         \tl_set_eq:NN \l__spath_tmpb_tl \g__spath_moveto_tl
1210         \tl_put_right:Nx \l__spath_tmpb_tl
1211         {
1212           {\dim_use:N \l__spath_tmpa_dim}
1213           {\dim_use:N \l__spath_tmpb_dim}
1214         }
1215         \tl_put_right:NV \l__spath_tmpb_tl \g__spath_lineto_tl
1216
1217         \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1218         \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
1219         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1220
1221         \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1222         \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
1223         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1224
1225       }
1226
1227       \g__spath_curvetoa_tl
1228       {
1229         \tl_set_eq:NN \l__spath_tmpb_tl \g__spath_moveto_tl
1230         \tl_put_right:Nx \l__spath_tmpb_tl
1231         {
1232           {\dim_use:N \l__spath_tmpa_dim}
1233           {\dim_use:N \l__spath_tmpb_dim}
1234         }
1235         \tl_put_right:NV \l__spath_tmpb_tl \g__spath_curvetoa_tl
1236

```

```

1237 \prg_replicate:nn {2} {
1238 \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1239 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1240 \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1241 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1242 \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
1243 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1244 }
1245
1246 \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1247 \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
1248 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1249
1250 \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1251 \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
1252 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1253
1254 }
1255
1256 \g__spath_closepath_tl
1257 {
1258 \tl_set_eq:NN \l__spath_tmpb_tl \g__spath_moveto_tl
1259 \tl_put_right:Nx \l__spath_tmpb_tl
1260 {
1261 {\dim_use:N \l__spath_tmpa_dim}
1262 {\dim_use:N \l__spath_tmpb_dim}
1263 }
1264 \tl_put_right:NV \l__spath_tmpb_tl \g__spath_lineto_tl
1265
1266 \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1267 \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
1268 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1269
1270 \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1271 \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
1272 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1273
1274 }
1275
1276 }
1277 {
1278
1279 \tl_set_eq:NN \l__spath_tmpb_tl \l__spath_tmpc_tl
1280 \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1281 \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
1282 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1283
1284 \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1285 \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
1286 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1287
1288 }
1289
1290 #2 \l__spath_tmpc_tl \l__spath_tmpb_tl

```

```

1291     \tl_clear:N \l__spath_tmpb_tl
1292
1293   }
1294 }
1295 \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.

```

1296 \NewDocumentCommand \MakeSPath { m m }
1297 {
1298   \spath_clear_new:n {#1}
1299   \spath_put:nno {#1} {path} {#2}
1300 }

```

(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.

```

1301 \NewDocumentCommand \MakeSPathList { m m }
1302 {
1303   \tl_gclear_new:c {l__spath_list_#1}
1304   \int_zero:N \l__spath_tmpa_int
1305   \spath_map_component:Nn #2 {
1306     \spath_clear_new:n {#1 _ \int_use:N \l__spath_tmpa_int}
1307     \spath_put:nnn {#1 _ \int_use:N \l__spath_tmpa_int} {path} {##1}
1308     \tl_gput_right:cx {l__spath_list_#1} {{#1 _ \int_use:N \l__spath_tmpa_int}}
1309     \int_incr:N \l__spath_tmpa_int
1310   }
1311 }

```

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

`\CloneSPath`

```

1312 \NewDocumentCommand \CloneSPath { m m }
1313 {
1314   \spath_clone:nn {#1} {#2}
1315 }

```

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

`\SPathInfo`

```

1316 \NewDocumentCommand \SPathInfo { m m }
1317 {
1318   \spath_get:nn {#1} {#2}
1319 }

```

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

\SPathPrepare

```
1320 \NewDocumentCommand \SPathPrepare { m }
1321 {
1322   \spath_generate_all:n {#1}
1323 }
```

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

\SPathListPrepare

```
1324 \NewDocumentCommand \SPathListPrepare { m }
1325 {
1326   \tl_map_inline:cn {l__spath_list_#1}
1327   {
1328     \spath_generate_all:n {##1}
1329   }
1330 }
```

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

\SPathInfoInto

```
1331 \NewDocumentCommand \SPathInfoInto { m m m }
1332 {
1333   \tl_clear_new:N #3
1334   \spath_get:nnN {#1} {#2} #3
1335 }
```

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

\SPathShow

```
1336 \NewDocumentCommand \SPathShow { m }
1337 {
1338   \spath_show:n {#1}
1339 }
```

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

\SPathTranslate

```
1340 \NewDocumentCommand \SPathTranslate { m m m }
1341 {
1342   \spath_translate:nnn {#1} {#2} {#3}
1343 }
```

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

\SPathTranslateInto

Clones the path before translating it.

```
1344 \NewDocumentCommand \SPathTranslateInto { m m m m }
1345 {
1346   \spath_clone:nn {#1} {#2}
1347   \spath_translate:nnn {#2} {#3} {#4}
1348 }
```

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

`\SPathScale`

```
1349 \NewDocumentCommand \SPathScale { m m m }
1350 {
1351   \spath_scale:nnn {#1} {#2} {#3}
1352 }
```

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

`\SPathScaleInto` Clones the path first.

```
1353 \NewDocumentCommand \SPathScaleInto { m m m m }
1354 {
1355   \spath_clone:nn {#1} {#2}
1356   \spath_scale:nnn {#2} {#3} {#4}
1357 }
```

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

`\SPathWeld`

```
1358 \NewDocumentCommand \SPathWeld { m m }
1359 {
1360   \spath_weld:nn {#1} {#2}
1361 }
```

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

`\SPathWeldInto`

```
1362 \NewDocumentCommand \SPathWeldInto { m m m }
1363 {
1364   \spath_clone:nn {#1} {#2}
1365   \spath_weld:nn {#2} {#3}
1366 }
```

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

Interfaces via TikZ keys.

```
1367 \tikzset{
1368   save~spath/.code={
1369     \tikz@addmode{
1370       \spath_get_current_path:n {#1}
1371     }
1372   },
1373   restore~spath/.code={
1374     \spath_set_current_path:n {#1}
1375   }
1376 }
```

2.7 Miscellaneous Commands

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

```
1377 \cs_new_nopar:Npn \spath_split_curve:nnNN #1#2#3#4
1378 {
1379   \group_begin:
1380   \tl_gclear:N \l__spath_smuggle_tl
1381   \tl_set_eq:NN \l__spath_tmpa_tl \g__spath_moveto_tl
1382   \tl_put_right:Nx \l__spath_tmpa_tl {
```



```

1383     {\tl_item:nn {#2} {2}}
1384     {\tl_item:nn {#2} {3}}
1385 }
1386 \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curvetoa_tl
1387 \tl_put_right:Nx \l__spath_tmpa_tl
1388 {
1389     {\fp_to_dim:n
1390     {
1391         (1 - #1) * \tl_item:nn {#2} {2} + (#1) * \tl_item:nn {#2} {5}
1392     }}
1393     {\fp_to_dim:n
1394     {
1395         (1 - #1) * \tl_item:nn {#2} {3} + (#1) * \tl_item:nn {#2} {6}
1396     }}
1397 }
1398 \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curvetob_tl
1399 \tl_put_right:Nx \l__spath_tmpa_tl
1400 {
1401     {\fp_to_dim:n
1402     {
1403         (1 - #1)^2 * \tl_item:nn {#2} {2} + 2 * (1 - #1) * (#1) * \tl_item:nn {#2} {5} + (#1)^2 * \tl_item:nn {#2} {6}
1404     }}
1405     {\fp_to_dim:n
1406     {
1407         (1 - #1)^2 * \tl_item:nn {#2} {3} + 2 * (1 - #1) * (#1) * \tl_item:nn {#2} {6} + (#1)^2 * \tl_item:nn {#2} {7}
1408     }}
1409 }
1410 \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curveto_tl
1411 \tl_put_right:Nx \l__spath_tmpa_tl
1412 {
1413     {\fp_to_dim:n
1414     {
1415         (1 - #1)^3 * \tl_item:nn {#2} {2} + 3 * (1 - #1)^2 * (#1) * \tl_item:nn {#2} {5} + 3 * (1 - #1) * (#1)^2 * \tl_item:nn {#2} {6} + (#1)^3 * \tl_item:nn {#2} {7}
1416     }}
1417     {\fp_to_dim:n
1418     {
1419         (1 - #1)^3 * \tl_item:nn {#2} {3} + 3 * (1 - #1)^2 * (#1) * \tl_item:nn {#2} {6} + 3 * (1 - #1) * (#1)^2 * \tl_item:nn {#2} {7} + (#1)^3 * \tl_item:nn {#2} {8}
1420     }}
1421 }
1422 \tl_gset_eq:NN \l__spath_smuggle_tl \l__spath_tmpa_tl
1423 \group_end:
1424 \tl_set_eq:NN #3 \l__spath_smuggle_tl
1425 \group_begin:
1426 \tl_set_eq:NN \l__spath_tmpa_tl \g__spath_moveto_tl
1427 \tl_put_right:Nx \l__spath_tmpa_tl
1428 {
1429     {\fp_to_dim:n
1430     {
1431         (1 - #1)^3 * \tl_item:nn {#2} {2} + 3 * (1 - #1)^2 * (#1) * \tl_item:nn {#2} {5} + 3 * (1 - #1) * (#1)^2 * \tl_item:nn {#2} {6} + (#1)^3 * \tl_item:nn {#2} {7}
1432     }}
1433     {\fp_to_dim:n
1434     {
1435         (1 - #1)^3 * \tl_item:nn {#2} {3} + 3 * (1 - #1)^2 * (#1) * \tl_item:nn {#2} {6} + 3 * (1 - #1) * (#1)^2 * \tl_item:nn {#2} {7} + (#1)^3 * \tl_item:nn {#2} {8}
1436     }}

```

```

1437 }
1438 \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curvetoa_tl
1439 \tl_put_right:Nx \l__spath_tmpa_tl
1440 {
1441   {\fp_to_dim:n
1442     {
1443       (1 - #1)^2 * \tl_item:nn {#2} {5} + 2 * (1 - #1) * (#1) * \tl_item:nn {#2} {8} + (#1)^2 * \tl_item:nn {#2} {11}
1444     }}
1445   {\fp_to_dim:n
1446     {
1447       (1 - #1)^2 * \tl_item:nn {#2} {6} + 2 * (1 - #1) * (#1) * \tl_item:nn {#2} {9} + (#1)^2 * \tl_item:nn {#2} {12}
1448     }}
1449 }
1450 \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curvetob_tl
1451 \tl_put_right:Nx \l__spath_tmpa_tl
1452 {
1453   {\fp_to_dim:n
1454     {
1455       (1 - #1) * \tl_item:nn {#2} {8} + (#1) * \tl_item:nn {#2} {11}
1456     }}
1457   {\fp_to_dim:n
1458     {
1459       (1 - #1) * \tl_item:nn {#2} {9} + (#1) * \tl_item:nn {#2} {12}
1460     }}
1461 }
1462 \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curveto_tl
1463 \tl_put_right:Nx \l__spath_tmpa_tl {
1464   {\tl_item:nn {#2} {11}}
1465   {\tl_item:nn {#2} {12}}
1466 }
1467 \tl_gset_eq:NN \l__spath_smuggle_tl \l__spath_tmpa_tl
1468 \group_end:
1469 \tl_set_eq:NN #4 \l__spath_smuggle_tl
1470 }
1471
1472 \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

```

1473 \RequirePackage{spath3}
1474 \ExplSyntaxOn
1475
1476 \tl_new:N \l__cal_tmpa_tl
1477 \tl_new:N \l__cal_tmpb_tl
1478 \int_new:N \l__cal_tmpa_int
1479 \int_new:N \l__cal_tmpb_int
1480 \int_new:N \l__cal_path_component_int
1481 \int_new:N \l__cal_label_int
1482 \dim_new:N \l__cal_tmpa_dim
1483 \dim_new:N \l__cal_tmpb_dim

```

```

1484 \dim_new:N \l__cal_tmpc_dim
1485 \dim_new:N \l__cal_tmpd_dim
1486 \dim_new:N \l__cal_tmpe_dim
1487 \dim_new:N \l__cal_tmpf_dim
1488 \dim_new:N \l__cal_tmpg_dim
1489 \dim_new:N \l__cal_tmph_dim
1490 \bool_new:N \l__cal_annotate_bool
1491 \bool_new:N \l__cal_taper_start_bool
1492 \bool_new:N \l__cal_taper_end_bool
1493 \bool_new:N \l__cal_taperable_bool
1494 \dim_new:N \l__cal_taper_width_dim
1495 \dim_new:N \l__cal_line_width_dim
1496
1497 \bool_set_true:N \l__cal_taper_start_bool
1498 \bool_set_true:N \l__cal_taper_end_bool
1499
1500 \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.

```

1501 \tikzset{
1502   define~pen/.code={
1503     \tikzset{pen~name=#1}
1504     \pgf@relevantforpicturesizefalse
1505     \tikz@addmode{
1506       \pgfsyssoftpath@getcurrentpath\l__cal_tmpa_tl
1507       \MakeSPathList{calligraphy pen \pgfkeysvalueof{/tikz/pen~name}}{\l__cal_tmpa_tl}
1508       \SPathListPrepare{calligraphy pen \pgfkeysvalueof{/tikz/pen~name}}
1509       \pgfusepath{discard}%
1510     }
1511   },
1512   define~pen/.default={default},
1513   use~pen/.code={
1514     \tikzset{pen~name=#1}
1515     \int_gzero:N \l__cal_path_component_int
1516     \cs_set_eq:NN \pgfpathmoveto \cal_moveto:n
1517     \tikz@addmode{
1518       \pgfsyssoftpath@getcurrentpath\l__cal_tmpa_tl
1519       \MakeSPathList{calligraphy path}{\l__cal_tmpa_tl}
1520       \SPathListPrepare{calligraphy path}
1521       \CalligraphyPathCreate{calligraphy path}{\pgfkeysvalueof{/tikz/pen~name}}
1522     }
1523   },
1524   use~pen/.default={default},
1525   pen~name/.initial={default},
1526   copperplate/.style={pen~name=copperplate},
1527   pen~colour/.initial={black},
1528   weight/.is~choice,
1529   weight/heavy/.style={
1530     line~width=\pgfkeysvalueof{/tikz/heavy~line~width},
1531     taper~width=\pgfkeysvalueof{/tikz/light~line~width},
1532   },
1533   weight/light/.style={

```

```

1534     line-width=\pgfkeysvalueof{/tikz/light-line-width},
1535     taper-width=0pt,
1536   },
1537   heavy/.style={
1538     weight=heavy
1539   },
1540   light/.style={
1541     weight=light
1542   },
1543   heavy~line~width/.initial=2pt,
1544   light~line~width/.initial=1pt,
1545   taper/.is~choice,
1546   taper/.default=both,
1547   taper/none/.style={
1548     taper~start=false,
1549     taper~end=false,
1550   },
1551   taper/both/.style={
1552     taper~start=true,
1553     taper~end=true,
1554   },
1555   taper/start/.style={
1556     taper~start=true,
1557     taper~end=false,
1558   },
1559   taper/end/.style={
1560     taper~start=false,
1561     taper~end=true,
1562   },
1563   taper~start/.code={
1564     \tl_if_eq:nnTF {#1} {true}
1565     {
1566       \bool_set_true:N \l__cal_taper_start_bool
1567     }
1568     {
1569       \bool_set_false:N \l__cal_taper_start_bool
1570     }
1571   },
1572   taper~start/.default={true},
1573   taper~end/.code={
1574     \tl_if_eq:nnTF {#1} {true}
1575     {
1576       \bool_set_true:N \l__cal_taper_end_bool
1577     }
1578     {
1579       \bool_set_false:N \l__cal_taper_end_bool
1580     }
1581   },
1582   taper~end/.default={true},
1583   taper~width/.code={\dim_set:Nn \l__cal_taper_width_dim {#1}},
1584   nib~style/.code~2~args={
1585     \tl_clear_new:c {l__cal_nib_style_#1}
1586     \tl_set:cn {l__cal_nib_style_#1} {#2}
1587   },

```

```

1588 stroke-style/.code~2~args={
1589   \tl_clear_new:c {l__cal_stroke_style_#1}
1590   \tl_set:cn {l__cal_stroke_style_#1} {#2}
1591 },
1592 this~stroke-style/.code={
1593   \tl_clear_new:c {l__cal_stroke_inline_style_ \int_use:N \l__cal_path_component_int}
1594   \tl_set:cn {l__cal_stroke_inline_style_ \int_use:N \l__cal_path_component_int} {#1}
1595 },
1596 annotate/.style={
1597   annotate~if,
1598   annotate~reset,
1599   annotation~style/.update~value={#1},
1600 },
1601 annotate~if/.default={true},
1602 annotate~if/.code={
1603   \tl_if_eq:nnTF {#1} {true}
1604   {
1605     \bool_set_true:N \l__cal_annotate_bool
1606   }
1607   {
1608     \bool_set_false:N \l__cal_annotate_bool
1609   }
1610 },
1611 annotate~reset/.code={
1612   \int_gzero:N \l__cal_label_int
1613 },
1614 annotation~style/.initial={draw,->},
1615 annotation~shift/.initial={{0,lex}},
1616 every~annotation~node/.initial={anchor=south~west},
1617 annotation~node~style/.code~2~args={
1618   \tl_set:cn {l__cal_annotation_style_ #1 _tl}{#2}
1619 },
1620 tl~use:N/.code={
1621   \exp_args:Nv \pgfkeysalso #1
1622 },
1623 tl~use:c/.code={
1624   \tl_if_exist:cT {#1}
1625   {
1626     \exp_args:Nv \pgfkeysalso {#1}
1627   }
1628 },
1629 /handlers/.update~style/.code={
1630   \tl_if_eq:nnF {#1} {\pgfkeysnovalue}
1631   {
1632     \pgfkeys{\pgfkeyscurrentpath/.code=\pgfkeysalso{#1}}
1633   }
1634 },
1635 /handlers/.update~value/.code={
1636   \tl_if_eq:nnF {#1} {\pgfkeysnovalue}
1637   {
1638     \pgfkeyssetvalue{\pgfkeyscurrentpath}{#1}
1639   }
1640 }
1641 }

```

Some wrappers around the TikZ keys.

```

1642 \NewDocumentCommand \pen { 0{} }
1643 {
1644   \path[define~ pen,every~ calligraphy~ pen/.try,#1]
1645 }
1646
1647 \NewDocumentCommand \definepen { 0{} }
1648 {
1649   \tikz \path[define~ pen,every~ calligraphy~ pen/.try,#1]
1650 }
1651
1652 \NewDocumentCommand \calligraphy { 0{} }
1653 {
1654   \path[use~ pen,every~ calligraphy/.try,#1]
1655 }

```

3.3 The Path Creation

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

```

1656 \NewDocumentCommand \CalligraphyPathCreate { m m }
1657 {
1658   \int_zero:N \l__cal_tmpa_int
1659   \tl_map_inline:cn {l__spath_list_#1}
1660   {
1661     \int_incr:N \l__cal_tmpa_int
1662     \int_zero:N \l__cal_tmpb_int
1663     \tl_map_inline:cn {l__spath_list_calligraphy pen #2}
1664     {
1665       \int_incr:N \l__cal_tmpb_int
1666       \group_begin:
1667       \pgfsys@beginscope
1668
1669       \cal_apply_style:c {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_int}
1670       \cal_apply_style:c {l__cal_stroke_inline_style_ \int_use:N \l__cal_tmpa_int}
1671       \cal_apply_style:c {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}
1672
1673       \spath_clone:nn {##1} {calligraphy temp path}
1674
1675       \__spath_get:nnN {###1} {initial point} \l__cal_tmpa_tl
1676       \spath_translate:nV {calligraphy temp path} \l__cal_tmpa_tl
1677
1678       \__spath_get:nnN {###1} {length} \l__cal_tmpa_tl
1679
1680       \int_compare:nTF {\l__cal_tmpa_tl = 1}
1681       {
1682         \cal_at_least_three:n {calligraphy temp path}
1683
1684         \spath_protocol_path:n {calligraphy temp path}
1685
1686         \__spath_get:nnN {calligraphy temp path} {path} \l__cal_tmpa_tl
1687
1688         \tikz@options
1689         \dim_set:Nn \l__cal_line_width_dim {\pgflinewidth}

```

```

1690     \cal_maybe_taper:N \l__cal_tmpa_tl
1691   }
1692   {
1693
1694     \spath_weld:nn {calligraphy temp path} {###1}
1695     \spath_reverse:n {##1}
1696     \spath_reverse:n {###1}
1697     \spath_weld:nn {calligraphy temp path} {##1}
1698     \spath_weld:nn {calligraphy temp path} {###1}
1699     \spath_reverse:n {##1}
1700     \spath_reverse:n {###1}
1701
1702     \tl_clear:N \l__cal_tmpa_tl
1703     \tl_set:Nn \l__cal_tmpa_tl {fill=\pgfkeysvalueof{/tikz/pen-colour},draw=none}
1704     \tl_if_exist:cT {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_int}
1705     {
1706       \tl_put_right:Nv \l__cal_tmpa_tl {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_int}
1707     }
1708     \tl_if_exist:cT {l__cal_stroke_inline_style_ \int_use:N \l__cal_tmpa_int}
1709     {
1710       \tl_put_right:Nn \l__cal_tmpa_tl {,}
1711       \tl_put_right:Nv \l__cal_tmpa_tl {l__cal_stroke_inline_style_ \int_use:N \l__cal_t
1712     }
1713     \tl_if_exist:cT {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}
1714     {
1715       \tl_put_right:Nn \l__cal_tmpa_tl {,}
1716       \tl_put_right:Nv \l__cal_tmpa_tl {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}
1717     }
1718     \spath_tikz_path:Vn \l__cal_tmpa_tl {calligraphy temp path}
1719
1720   }
1721   \pgfsys@endscope
1722   \group_end:
1723 }
1724 \bool_if:NT \l__cal_annotate_bool
1725 {
1726   \spath_clone:nn {##1} {calligraphy temp path}
1727   \tl_set_eq:Nc \l_tmpa_tl {l__spath_list_calligraphy pen #2}
1728   \tl_reverse:N \l_tmpa_tl
1729   \tl_set:Nx \l_tmpa_tl {\tl_head:N \l_tmpa_tl}
1730   \spath_generate_finalpoint:V \l_tmpa_tl
1731   \spath_get:VnN \l_tmpa_tl {final point} \l_tmpa_tl
1732   \spath_translate:nV {calligraphy temp path} \l_tmpa_tl
1733   \tikz@scan@one@point\pgfutil@firstofone\pgfkeysvalueof{/tikz/annotation-shift}
1734   \spath_translate:nnn {calligraphy temp path} {\pgf@x} {\pgf@y}
1735
1736   \pgfkeysgetvalue{/tikz/annotation-style}{\l_tmpa_tl}
1737   \spath_tikz_path:Vn \l_tmpa_tl {calligraphy temp path}
1738   \spath_get:nnN {calligraphy temp path} {final point} \l_tmpa_tl
1739   \exp_last_unbraced:NV \pgfqpoint \l_tmpa_tl
1740   \begin{scope}[reset~ cm]
1741     \node[every~annotation~node/.try,tl~use:c = {l__cal_annotate_style_ \int_use:N \l__
1742   \end{scope}
1743 }

```

```

1744 }
1745 }

```

(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.

```

1746 \cs_new_eq:NN \cal_orig_moveto:n \pgfpathmoveto
1747 \cs_new_nopar:Npn \cal_moveto:n #1
1748 {
1749   \int_gincr:N \l__cal_path_component_int
1750   \cal_orig_moveto:n {#1}
1751 }

```

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

`\cal_apply_style:N` Interface for applying `\tikzset` to a token list.

```

1752 \cs_new_nopar:Npn \cal_apply_style:N #1
1753 {
1754   \tl_if_exist:NT #1 {
1755     \exp_args:NV \tikzset #1
1756   }
1757 }
1758 \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.

```

1759 \cs_new_nopar:Npn \cal_at_least_three:n #1
1760 {
1761   \spath_get:nnN {#1} {real length} \l__cal_tmpa_tl
1762   \tl_clear:N \l__cal_tmpb_tl
1763   \int_compare:nTF {\l__cal_tmpa_tl = 1}
1764   {
1765     \spath_get:nnN {#1} {path} \l__cal_tmpa_tl
1766     \spath_map_segment_inline:Nn \l__cal_tmpa_tl
1767     {
1768       \tl_case:NnF ##1 {
1769         \g__spath_lineto_tl {
1770           \cal_split_line_in_three:NN \l__cal_tmpb_tl ##2
1771         }
1772         \g__spath_curvetoa_tl {
1773           \cal_split_curve_in_three:NN \l__cal_tmpb_tl ##2
1774         }
1775       }
1776     }
1777     \tl_put_right:NV \l__cal_tmpb_tl ##2
1778   }
1779 }
1780 \spath_put:nnV {#1} {path} \l__cal_tmpb_tl
1781 }
1782 {
1783   \int_compare:nT {\l__cal_tmpa_tl = 2}
1784   {

```



```

1785     \spath_get:nnN {#1} {path} \l__cal_tmpa_tl
1786     \spath_map_segment_inline:Nn \l__cal_tmpa_tl
1787     {
1788         \tl_case:NnF ##1 {
1789             \g__spath_lineto_tl {
1790                 \cal_split_line_in_two:NN \l__cal_tmpb_tl ##2
1791             }
1792             \g__spath_curvetoa_tl {
1793                 \cal_split_curve_in_two:NN \l__cal_tmpb_tl ##2
1794             }
1795         }
1796         {
1797             \tl_put_right:NV \l__cal_tmpb_tl ##2
1798         }
1799     }
1800     \spath_put:nnV {#1} {path} \l__cal_tmpb_tl
1801 }
1802 }
1803 }

```

(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.

```

1804 \cs_new_nopar:Npn \cal_split_line_in_two:NN #1#2
1805 {
1806     \tl_set_eq:NN \l__cal_tmpc_tl #2
1807
1808     \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1809
1810     \dim_set:Nn \l__cal_tmpa_dim {\tl_head:N \l__cal_tmpc_tl}
1811     \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1812
1813     \dim_set:Nn \l__cal_tmpb_dim {\tl_head:N \l__cal_tmpc_tl}
1814     \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1815
1816     \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1817
1818     \dim_set:Nn \l__cal_tmpc_dim {\tl_head:N \l__cal_tmpc_tl}
1819     \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1820     \dim_set:Nn \l__cal_tmpd_dim {\tl_head:N \l__cal_tmpc_tl}
1821     \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1822
1823     \tl_put_right:NV #1 \g__spath_lineto_tl
1824
1825     \tl_put_right:Nx #1 {
1826         {\dim_eval:n {(\l__cal_tmpa_dim + \l__cal_tmpc_dim)/2}}
1827         {\dim_eval:n {(\l__cal_tmpb_dim + \l__cal_tmpd_dim)/2}}
1828     }
1829
1830     \tl_put_right:NV #1 \g__spath_lineto_tl
1831     \tl_put_right:Nx #1 {
1832         {\dim_use:N \l__cal_tmpc_dim}
1833         {\dim_use:N \l__cal_tmpd_dim}
1834     }
1835 }

```

(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.
1836 \cs_new_nopar:Npn \cal_split_line_in_three:NN #1#2
1837 {
1838   \tl_set_eq:NN \l__cal_tmpc_tl #2
1839
1840   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1841
1842   \dim_set:Nn \l__cal_tmpa_dim {\tl_head:N \l__cal_tmpc_tl}
1843   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1844
1845   \dim_set:Nn \l__cal_tmpb_dim {\tl_head:N \l__cal_tmpc_tl}
1846   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1847
1848   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1849
1850   \dim_set:Nn \l__cal_tmpc_dim {\tl_head:N \l__cal_tmpc_tl}
1851   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1852   \dim_set:Nn \l__cal_tmpd_dim {\tl_head:N \l__cal_tmpc_tl}
1853   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1854
1855   \tl_put_right:NV #1 \g__spath_lineto_tl
1856
1857   \tl_put_right:Nx #1 {
1858     {\dim_eval:n {(2\l__cal_tmpa_dim + \l__cal_tmpc_dim)/3}}
1859     {\dim_eval:n {(2\l__cal_tmpb_dim + \l__cal_tmpd_dim)/3}}
1860   }
1861
1862   \tl_put_right:NV #1 \g__spath_lineto_tl
1863
1864   \tl_put_right:Nx #1 {
1865     {\dim_eval:n {(\l__cal_tmpa_dim + 2\l__cal_tmpc_dim)/3}}
1866     {\dim_eval:n {(\l__cal_tmpb_dim + 2\l__cal_tmpd_dim)/3}}
1867   }
1868
1869   \tl_put_right:NV #1 \g__spath_lineto_tl
1870   \tl_put_right:Nx #1 {
1871     {\dim_use:N \l__cal_tmpc_dim}
1872     {\dim_use:N \l__cal_tmpd_dim}
1873   }
1874 }

```

(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.
1875 \cs_new_nopar:Npn \cal_split_curve_in_two:NN #1#2
1876 {
1877   \spath_split_curve:nVNN {.5} #2 \l_tmpa_tl \l_tmpb_tl
1878   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1879   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1880   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1881   \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
1882   \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}

```

```

1883 \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
1884 \tl_put_right:NV #1 \l_tmpa_tl
1885 \tl_put_right:NV #1 \l_tmpb_tl
1886 }

```

(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.

```

1887 \cs_new_nopar:Npn \cal_split_curve_in_three:NN #1#2
1888 {
1889   \spath_split_curve:nVNN {1/3} #2 \l_tmpa_tl \l_tmpb_tl
1890
1891   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1892   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1893   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1894   \tl_put_right:NV #1 \l_tmpa_tl
1895
1896   \spath_split_curve:nVNN {.5} \l_tmpb_tl \l_tmpa_tl \l_tmpb_tl
1897   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1898   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1899   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1900   \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
1901   \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
1902   \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
1903   \tl_put_right:NV #1 \l_tmpa_tl
1904   \tl_put_right:NV #1 \l_tmpb_tl
1905 }

```

(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.

```

1906 \cs_new_nopar:Npn \cal_maybe_taper:N #1
1907 {
1908   \tl_set_eq:NN \l__cal_tmpa_tl #1
1909
1910   \bool_if:NT \l__cal_taper_start_bool
1911   {
1912
1913     \dim_set:Nn \l__cal_tmpa_dim {\tl_item:Nn \l__cal_tmpa_tl {2}}
1914     \dim_set:Nn \l__cal_tmpb_dim {\tl_item:Nn \l__cal_tmpa_tl {3}}
1915     \tl_set:Nx \l__cal_tmpb_tl {\tl_item:Nn \l__cal_tmpa_tl {4}}
1916
1917     \tl_case:NnF \l__cal_tmpb_tl
1918     {
1919       \g__spath_lineto_tl
1920       {
1921
1922         \bool_set_true:N \l__cal_taperable_bool
1923         \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {5}}
1924         \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {6}}
1925         \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpa_dim + \l__cal_tmph_dim)/3}
1926         \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmph_dim + \l__cal_tmpe_dim)/3}
1927         \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmpa_dim + 2\l__cal_tmpe_dim)/3}
1928         \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmpe_dim + 2\l__cal_tmph_dim)/3}
1929         \prg_replicate:nn {4}

```

```

1930     {
1931     \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
1932     }
1933     \tl_put_left:NV \l__cal_tmpa_tl \g__spath_moveto_tl
1934   }
1935   \g__spath_curvetoa_tl
1936   {
1937     \bool_set_true:N \l__cal_taperable_bool
1938     \dim_set:Nn \l__cal_tmpc_dim {\tl_item:Nn \l__cal_tmpa_tl {5}}
1939     \dim_set:Nn \l__cal_tmpd_dim {\tl_item:Nn \l__cal_tmpa_tl {6}}
1940     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {8}}
1941     \dim_set:Nn \l__cal_tmpf_dim {\tl_item:Nn \l__cal_tmpa_tl {9}}
1942     \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {11}}
1943     \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {12}}
1944     \prg_replicate:nn {10}
1945     {
1946       \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
1947     }
1948     \tl_put_left:NV \l__cal_tmpa_tl \g__spath_moveto_tl
1949   }
1950 }
1951 {
1952   \bool_set_false:N \l__cal_taperable_bool
1953 }
1954
1955 \bool_if:NT \l__cal_taperable_bool
1956 {
1957   \__cal_taper_aux:
1958 }
1959
1960 }
1961
1962 \bool_if:NT \l__cal_taper_end_bool
1963 {
1964
1965   \dim_set:Nn \l__cal_tmpa_dim {\tl_item:Nn \l__cal_tmpa_tl {-2}}
1966   \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {-1}}
1967   \tl_set:Nx \l__cal_tmph_tl {\tl_item:Nn \l__cal_tmpa_tl {-3}}
1968
1969   \tl_case:NnF \l__cal_tmph_tl
1970   {
1971     \g__spath_lineto_tl
1972     {
1973
1974       \bool_set_true:N \l__cal_taperable_bool
1975       \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {-5}}
1976       \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {-4}}
1977       \dim_set:Nn \l__cal_tmph_dim {(2\l__cal_tmpa_dim + \l__cal_tmph_dim)/3}
1978       \dim_set:Nn \l__cal_tmph_dim {(2\l__cal_tmph_dim + \l__cal_tmph_dim)/3}
1979       \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmph_dim + 2\l__cal_tmph_dim)/3}
1980       \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmph_dim + 2\l__cal_tmph_dim)/3}
1981       \tl_reverse:N \l__cal_tmph_tl
1982       \prg_replicate:nn {3}
1983       {

```

```

1984     \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
1985   }
1986   \tl_reverse:N \l__cal_tmpa_tl
1987 }
1988 \g__spath_curveto_tl
1989 {
1990   \bool_set_true:N \l__cal_taperable_bool
1991   \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-5}}
1992   \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-4}}
1993   \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-8}}
1994   \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-7}}
1995   \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-11}}
1996   \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-10}}
1997   \tl_reverse:N \l__cal_tmpa_tl
1998   \prg_replicate:nn {9}
1999   {
2000     \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
2001   }
2002   \tl_reverse:N \l__cal_tmpa_tl
2003 }
2004 }
2005 {
2006   \bool_set_false:N \l__cal_taperable_bool
2007 }
2008
2009 \bool_if:NT \l__cal_taperable_bool
2010 {
2011   \__cal_taper_aux:
2012 }
2013
2014 }
2015
2016 \pgfsyssoftpath@setcurrentpath\l__cal_tmpa_tl
2017 \pgfsetstrokecolor{\pgfkeysvalueof{/tikz/pen~colour}}
2018 \pgfusepath{stroke}
2019
2020 }

```

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

__cal_taper_aux: Auxiliary macro to avoid unnecessary code duplication.

```

2021 \cs_new_nopar:Npn \__cal_taper_aux:
2022 {
2023   \tl_clear:N \l__cal_tmpe_tl
2024   \tl_put_right:NV \l__cal_tmpe_tl \g__spath_moveto_tl
2025
2026   \fp_set:Nn \l__cal_tmpe_fp
2027   {
2028     \l__cal_tmpe_dim - \l__cal_tmpe_dim
2029   }
2030   \fp_set:Nn \l__cal_tmpe_fp
2031   {
2032     \l__cal_tmpe_dim - \l__cal_tmpe_dim
2033   }

```

```

2034 \fp_set:Nn \l__cal_tmpe_fp
2035 {
2036   (\l__cal_tmpe_fp^2 + \l__cal_tmpe_fp)^.5
2037 }
2038
2039 \fp_set:Nn \l__cal_tmpe_fp {.5*\l__cal_taper_width_dim * \l__cal_tmpe_fp / \l__cal_tmpe_fp}
2040 \fp_set:Nn \l__cal_tmpe_fp {.5*\l__cal_taper_width_dim * \l__cal_tmpe_fp / \l__cal_tmpe_fp}
2041
2042 \fp_set:Nn \l__cal_tmpe_fp
2043 {
2044   \l__cal_tmpe_dim - \l__cal_tmpe_dim
2045 }
2046 \fp_set:Nn \l__cal_tmpe_fp
2047 {
2048   \l__cal_tmpe_dim - \l__cal_tmpe_dim
2049 }
2050 \fp_set:Nn \l__cal_tmpe_fp
2051 {
2052   (\l__cal_tmpe_fp^2 + \l__cal_tmpe_fp)^.5
2053 }
2054
2055 \fp_set:Nn \l__cal_tmpe_fp {.5*\l__cal_line_width_dim * \l__cal_tmpe_fp / \l__cal_tmpe_fp}
2056 \fp_set:Nn \l__cal_tmpe_fp {.5*\l__cal_line_width_dim * \l__cal_tmpe_fp / \l__cal_tmpe_fp}
2057
2058 \tl_put_right:Nx \l__cal_tmpe_tl
2059 {
2060   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim}}
2061   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim}}
2062 }
2063
2064 \tl_put_right:NV \l__cal_tmpe_tl \g__spath_curvetoa_tl
2065
2066 \tl_put_right:Nx \l__cal_tmpe_tl
2067 {
2068   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim}}
2069   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim}}
2070 }
2071
2072 \tl_put_right:NV \l__cal_tmpe_tl \g__spath_curvetob_tl
2073
2074 \tl_put_right:Nx \l__cal_tmpe_tl
2075 {
2076   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim}}
2077   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim}}
2078 }
2079
2080 \tl_put_right:NV \l__cal_tmpe_tl \g__spath_curveto_tl
2081
2082 \tl_put_right:Nx \l__cal_tmpe_tl
2083 {
2084   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim}}
2085   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim}}
2086 }
2087

```

```

2088 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetoa_tl
2089
2090 \tl_put_right:Nx \l__cal_tmpb_tl
2091 {
2092   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim - \fp_to_dim:n{ 1.32 * \l
2093     {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \fp_to_dim:n {1.32* \l
2094   }
2095 }
2096 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetob_tl
2097
2098 \tl_put_right:Nx \l__cal_tmpb_tl
2099 {
2100   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \fp_to_dim:n {1.32 * \
2101     {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \fp_to_dim:n {1.32 * \
2102   }
2103 }
2104 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetoa_tl
2105
2106 \tl_put_right:Nx \l__cal_tmpb_tl
2107 {
2108   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \fp_to_dim:n {1.32 * \
2109     {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \fp_to_dim:n {1.32 * \
2110   }
2111 }
2112 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetoa_tl
2113
2114 \tl_put_right:Nx \l__cal_tmpb_tl
2115 {
2116   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \fp_to_dim:n {1.32 * \
2117     {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \fp_to_dim:n {1.32 * \
2118   }
2119 }
2120 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetob_tl
2121
2122 \tl_put_right:Nx \l__cal_tmpb_tl
2123 {
2124   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \fp_to_dim:n {1.32 * \
2125     {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \fp_to_dim:n {1.32 * \
2126   }
2127 }
2128 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetoa_tl
2129
2130 \tl_put_right:Nx \l__cal_tmpb_tl
2131 {
2132   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \fp_to_dim:n {1.32 * \
2133     {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \fp_to_dim:n {1.32 * \
2134   }
2135 }
2136 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetoa_tl
2137
2138 \tl_put_right:Nx \l__cal_tmpb_tl
2139 {
2140   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \fp_to_dim:n {1.32 * \
2141     {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \fp_to_dim:n {1.32 * \

```

```

2142 }
2143
2144 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetob_tl
2145
2146 \tl_put_right:Nx \l__cal_tmpb_tl
2147 {
2148   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim + \fp_to_dim:n {1.32 * \l
2149     {\dim_eval:n { \fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim - \fp_to_dim:n {1.32 * \l
2150   }
2151
2152 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curveto_tl
2153
2154 \tl_put_right:Nx \l__cal_tmpb_tl
2155 {
2156   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim}}
2157   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim}}
2158 }
2159
2160 \pgfsyssoftpath@setcurrentpath\l__cal_tmpb_tl
2161 \pgfsetfillcolor{\pgfkeysvalueof{/tikz/pen~colour}}
2162 \pgfusepath{fill}
2163 }

```

(End definition for `__cal_taper_aux:.`)

Defines a copperplate pen.

```

2164 \tl_set:Nn \l__cal_tmpa_tl {\pgfsyssoftpath@movetotoken{Opt}{Opt}}
2165 \MakeSPathList{calligraphy pen copperplate}{\l__cal_tmpa_tl}
2166 \SPathListPrepare{calligraphy pen copperplate}
2167 \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.

```

2168 \expandafter\ifx\csname pgfdeclaredecoration\endcsname\relax
2169 \else
2170 \pgfdeclaredecoration{calligraphic brace}{brace}
2171 {
2172   \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]
2173   {
2174     \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}
2175     \pgfpathmoveto{\pgfpointorigin}
2176     \pgfpathcurveto
2177     {\pgfqpoint{.15\pgfdecorationsegmentamplitude}{.3\pgfdecorationsegmentamplitude}}
2178     {\pgfqpoint{.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
2179     {\pgfqpoint{\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
2180     {
2181       \pgftransformxshift{+\pgfdecorationsegmentaspect\pgfdecoratedremainingdistance}
2182       \pgfpathlineto{\pgfqpoint{-\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
2183       \pgfpathcurveto
2184       {\pgfqpoint{-.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
2185       {\pgfqpoint{-.15\pgfdecorationsegmentamplitude}{.7\pgfdecorationsegmentamplitude}}

```



```

2186     {\pgfqpoint{0\pgfdecorationsegmentamplitude}{1\pgfdecorationsegmentamplitude}}
2187     \pgfpathmoveto{\pgfqpoint{0\pgfdecorationsegmentamplitude}{1\pgfdecorationsegmentamplitude}}
2188     \pgfpathcurveto
2189     {\pgfqpoint{.15\pgfdecorationsegmentamplitude}{.7\pgfdecorationsegmentamplitude}}
2190     {\pgfqpoint{.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
2191     {\pgfqpoint{\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
2192   }
2193   {
2194     \pgftransformxshift{+\pgfdecoratedremainingdistance}
2195     \pgfpathlineto{\pgfqpoint{-\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
2196     \pgfpathcurveto
2197     {\pgfqpoint{-.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
2198     {\pgfqpoint{-.15\pgfdecorationsegmentamplitude}{.3\pgfdecorationsegmentamplitude}}
2199     {\pgfqpoint{Opt}{Opt}}
2200   }
2201   \tikzset{
2202     taper width=.5\pgflinewidth,
2203     taper
2204   }%
2205   \pgfsyssoftpath@getcurrentpath\cal@tmp@path
2206   \MakeSPPathList{calligraphy path}{\cal@tmp@path}%
2207   \SPPathListPrepare{calligraphy path}%
2208   \CalligraphyPathCreate{calligraphy path}{copperplate}%
2209 }
2210 \state{final}{}
2211 }

```

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

```

2212 \pgfdeclaredecoration{calligraphic straight parenthesis}{brace}
2213 {
2214   \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]
2215   {
2216     \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}
2217     \pgfpathmoveto{\pgfpointorigin}
2218     \pgfpathcurveto
2219     {\pgfqpoint{.76604\pgfdecorationsegmentamplitude}{.64279\pgfdecorationsegmentamplitude}}
2220     {\pgfqpoint{2.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
2221     {\pgfqpoint{3.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
2222     {
2223       \pgftransformxshift{+\pgfdecoratedremainingdistance}
2224       \pgfpathlineto{\pgfqpoint{-3.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
2225       \pgfpathcurveto
2226       {\pgfqpoint{-2.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
2227       {\pgfqpoint{-.76604\pgfdecorationsegmentamplitude}{.64279\pgfdecorationsegmentamplitude}}
2228       {\pgfqpoint{Opt}{Opt}}
2229     }
2230     \tikzset{
2231       taper width=.5\pgflinewidth,
2232       taper
2233     }%
2234     \pgfsyssoftpath@getcurrentpath\cal@tmp@path
2235     \MakeSPPathList{calligraphy path}{\cal@tmp@path}%
2236     \SPPathListPrepare{calligraphy path}%

```

```

2237     \CalligraphyPathCreate{calligraphy path}{copperplate}%
2238   }
2239   \state{final}{}%
2240 }

The third is a curved parenthesis.

2241 \pgfdeclaredecoration{calligraphic curved parenthesis}{brace}
2242 {
2243   \state{brace}[width+=\pgfdecoratedremainingdistance,next state=final]
2244   {
2245     \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}
2246     \pgfpathmoveto{\pgfpointorigin}
2247     \pgf@xa=\pgfdecoratedremainingdistance\relax
2248     \advance\pgf@xa by -1.5890\pgfdecorationsegmentamplitude\relax
2249     \edef\cgrphy@xa{\the\pgf@xa}
2250     \pgfpathcurveto
2251     {\pgfpoint{1.5890\pgfdecorationsegmentamplitude}{1.3333\pgfdecorationsegmentamplitude}}
2252     {\pgfpoint{\cgrphy@xa}{1.3333\pgfdecorationsegmentamplitude}}
2253     {\pgfpoint{\pgfdecoratedremainingdistance}{0pt}}
2254     \tikzset{
2255       taper width=.5\pgflinewidth,
2256       taper
2257     }%
2258     \pgfsyssoftpath@getcurrentpath\cal@tmp@path
2259     \MakeSPPathList{calligraphy path}{\cal@tmp@path}%
2260     \SPPathListPrepare{calligraphy path}%
2261     \CalligraphyPathCreate{calligraphy path}{copperplate}%
2262   }
2263   \state{final}{}%
2264 }

End the conditional for if pgfdecoration module is loaded
2265 \fi

```

4 Drawing Knots

4.1 Initialisation

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

```

2266 \RequirePackage{spath3}
2267 \usetikzlibrary{intersections}
2268
2269 \ExplSyntaxOn
2270
2271 \tl_new:N \l__knot_tmpa_tl
2272 \tl_new:N \l__knot_tmpb_tl
2273 \tl_new:N \l__knot_tmpe_tl
2274 \tl_new:N \l__knot_tmpe_tl
2275 \tl_new:N \l__knot_tmpe_tl
2276 \tl_new:N \l__knot_tmpe_tl
2277 \tl_new:N \l__knot_tmpe_tl
2278 \tl_new:N \l__knot_redraws_tl
2279 \tl_new:N \l__knot_clip_width_tl
2280 \tl_new:N \l__knot_name_tl

```

```

2281 \tl_new:N \l__knot_node_tl
2282 \tl_new:N \l__knot_aux_tl
2283 \tl_new:N \l__knot_auxa_tl
2284
2285 \int_new:N \l__knot_tmpa_int
2286 \int_new:N \l__knot_strands_int
2287 \int_new:N \l__knot_intersections_int
2288 \int_new:N \l__knot_filaments_int
2289 \int_new:N \l__knot_component_start_int
2290
2291 \dim_new:N \l__knot_tmpa_dim
2292 \dim_new:N \l__knot_tmpb_dim
2293 \dim_new:N \l__knot_tmpe_dim
2294 \dim_new:N \l__knot_tolerance_dim
2295 \dim_new:N \l__knot_clip_bg_radius_dim
2296 \dim_new:N \l__knot_clip_draw_radius_dim
2297
2298 \bool_new:N \l__knot_draft_bool
2299 \bool_new:N \l__knot_ignore_ends_bool
2300 \bool_new:N \l__knot_self_intersections_bool
2301 \bool_new:N \l__knot_splits_bool
2302 \bool_new:N \l__knot_super_draft_bool
2303
2304 \bool_new:N \l__knot_prepend_prev_bool
2305 \bool_new:N \l__knot_append_next_bool
2306 \bool_new:N \l__knot_skip_bool
2307 \bool_new:N \l__knot_save_bool
2308
2309 \seq_new:N \l__knot_nodes_seq
2310
2311 \bool_set_true:N \l__knot_ignore_ends_bool
Configuration is via TikZ keys and styles.
2312 \tikzset{
2313   knot/.code={
2314     \tl_if_eq:nnTF {#1} {none}
2315     {
2316       \tikz@addmode{\tikz@mode@doublefalse}
2317     }
2318     {
2319       \tikz@addmode{\tikz@mode@doubletrue}
2320       \tl_if_eq:nnTF {\pgfkeysnovalue} {#1}
2321       {
2322         \tikz@addoption{\pgfsetinnerstrokecolor{.}}
2323       }
2324       {
2325         \pgfsetinnerstrokecolor{#1}
2326       }
2327       \tikz@addoption{
2328         \pgfsetstrokecolor{knotbg}
2329       }
2330       \tl_set:Nn \tikz@double@setup{
2331         \pgfsetinnerlinewidth{\pgflinewidth}
2332         \pgfsetlinewidth{\dim_eval:n {\tl_use:N \l__knot_gap_tl \pgflinewidth}}
2333       }

```

```

2334     }
2335   },
2336   knot~ gap/.store~ in=\l__knot_gap_tl,
2337   knot~ gap=3,
2338   knot~ diagram/.is~family,
2339   knot~ diagram/.unknown/.code={
2340     \tl_set_eq:NN \l__knot_tmpa_tl \pgfkeyscurrentname
2341     \pgfkeysalso{
2342       /tikz/\l__knot_tmpa_tl=#1
2343     }
2344   },
2345   background~ colour/.code={%
2346     \colorlet{knotbg}{#1}%
2347   },
2348   background~ color/.code={%
2349     \colorlet{knotbg}{#1}%
2350   },
2351   background~ colour=white,
2352   knot~ diagram,
2353   name/.store~ in=\l__knot_name_tl,
2354   name={knot},
2355   save~ intersections/.is~ choice,
2356   save~ intersections/.default=true,
2357   save~ intersections/true/.code={
2358     \bool_set_true:N \l__knot_save_bool
2359   },
2360   save~ intersections/false/.code={
2361     \bool_set_false:N \l__knot_save_bool
2362   },
2363   every~ strand/.style={draw},
2364   ignore~ endpoint~ intersections/.code={
2365     \tl_if_eq:nnTF {#1} {true}
2366     {
2367       \bool_set_true:N \l__knot_ignore_ends_bool
2368     }
2369     {
2370       \bool_set_false:N \l__knot_ignore_ends_bool
2371     }
2372   },
2373   ignore~ endpoint~ intersections/.default=true,
2374   consider~ self~ intersections/.is~choice,
2375   consider~ self~ intersections/true/.code={
2376     \bool_set_true:N \l__knot_self_intersections_bool
2377     \bool_set_true:N \l__knot_splits_bool
2378   },
2379   consider~ self~ intersections/false/.code={
2380     \bool_set_false:N \l__knot_self_intersections_bool
2381     \bool_set_false:N \l__knot_splits_bool
2382   },
2383   consider~ self~ intersections/no~ splits/.code={
2384     \bool_set_true:N \l__knot_self_intersections_bool
2385     \bool_set_false:N \l__knot_splits_bool
2386   },
2387   consider~ self~ intersections/.default={true},

```

```

2388 clip~ radius/.code={
2389   \dim_set:Nn \l__knot_clip_bg_radius_dim {#1}
2390   \dim_set:Nn \l__knot_clip_draw_radius_dim {#1+2pt}
2391 },
2392 clip~ draw~ radius/.code={
2393   \dim_set:Nn \l__knot_clip_draw_radius_dim {#1}
2394 },
2395 clip~ background~ radius/.code={
2396   \dim_set:Nn \l__knot_clip_bg_radius_dim {#1}
2397 },
2398 clip~ radius=10pt,
2399 end~ tolerance/.code={
2400   \dim_set:Nn \l__knot_tolerance_dim {#1}
2401 },
2402 end~ tolerance=14pt,
2403 clip/.style={
2404   clip
2405 },
2406 background~ clip/.style={
2407   clip
2408 },
2409 clip~ width/.code={
2410   \tl_set:Nn \l__knot_clip_width_tl {#1}
2411 },
2412 clip~ width=3,
2413 flip~ crossing/.code={%
2414   \tl_clear_new:c {l__knot_crossing_#1}
2415   \tl_set:cn {l__knot_crossing_#1} {x}
2416 },
2417 ignore~ crossing/.code={%
2418   \tl_clear_new:c {l__knot_ignore_crossing_#1}
2419   \tl_set:cn {l__knot_ignore_crossing_#1} {x}
2420 },
2421 draft~ mode/.is~ choice,
2422 draft~ mode/off/.code={%
2423   \bool_set_false:N \l__knot_draft_bool
2424   \bool_set_false:N \l__knot_super_draft_bool
2425 },
2426 draft~ mode/crossings/.code={%
2427   \bool_set_true:N \l__knot_draft_bool
2428   \bool_set_false:N \l__knot_super_draft_bool
2429 },
2430 draft~ mode/strands/.code={%
2431   \bool_set_true:N \l__knot_draft_bool
2432   \bool_set_true:N \l__knot_super_draft_bool
2433 },
2434 draft/.is~ family,
2435 draft,
2436 crossing~ label/.style={
2437   overlay,
2438   fill=white,
2439   fill~ opacity=.5,
2440   text~ opacity=1,
2441   text=blue,

```

```

2442     pin~ edge={blue,<-}
2443   },
2444   strand~ label/.style={
2445     overlay,
2446     circle,
2447     draw=purple,
2448     fill=white,
2449     fill~ opacity=.5,
2450     text~ opacity=1,
2451     text=purple,
2452     inner~ sep=0pt
2453   },
2454 }

```

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

```

2455 \cs_new_nopar:Npn \knot_apply_style:N #1
2456 {
2457   \tl_if_exist:NT #1 {
2458     \exp_args:NV \tikzset #1
2459   }
2460 }
2461 \cs_generate_variant:Nn \knot_apply_style:N {c}

```

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

```

2462 \NewDocumentCommand \flipcrossings {m}
2463 {
2464   \tikzset{knot~ diagram/flip~ crossing/.list={#1}}%
2465 }

```

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

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

```

2466 \NewDocumentCommand \strand { O{} }
2467 {
2468   \int_incr:N \l__knot_strands_int
2469   \tl_clear_new:c {l__knot_options_strand \int_use:N \l__knot_strands_int}
2470   \tl_set:cn {l__knot_options_strand \int_use:N \l__knot_strands_int} {#1}
2471   \path[#1,save~ spath=knot strand \int_use:N \l__knot_strands_int]
2472 }

```

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

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

```

2473 \NewDocumentEnvironment{knot} { O{} }
2474 {
2475   \knot_initialise:n {#1}
2476 }
2477 {
2478   \knot_render:
2479 }

```

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

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

```
2480 \cs_new_protected_nopar:Npn \knot_initialise:n #1
2481 {
2482   \tikzset{knot~ diagram/.cd,every~ knot~ diagram/.try,#1}
2483   \int_zero:N \l__knot_strands_int
2484   \tl_clear:N \l__knot_redraws_tl
2485   \seq_gclear:N \l__knot_nodes_seq
2486 }
```

(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.

```
2487 \cs_new_protected_nopar:Npn \knot_render:
2488 {
```

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

```
2489   \pgfscope
2490   \pgftransformreset
```

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

```
2491   \int_step_function:nnnN {1} {1} {\l__knot_strands_int} \knot_draw_strand:n
```

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

```
2492   \bool_if:NF \l__knot_super_draft_bool
2493   {
```

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.

```
2494     \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.

```
2495     \bool_if:NTF \l__knot_self_intersections_bool
2496     {
2497       \knot_split_strands:
2498       \int_set_eq:NN \l__knot_tmpa_int \l__knot_filaments_int
2499       \tl_set:Nn \l__knot_prefix_tl {filament}
2500     }
2501     {
2502       \int_set_eq:NN \l__knot_tmpa_int \l__knot_strands_int
2503       \tl_set:Nn \l__knot_prefix_tl {strand}
2504     }
```

Initialise the intersection count.

```
2505     \int_gzero:N \l__knot_intersections_int
```

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

```
2506     \bool_if:NTF \l__knot_draft_bool
2507     {
2508       \tl_set:Nn \l__knot_node_tl {
2509         \exp_not:N \node[coordinate,
2510           pin={\node~ contents={\int_use:N \l__knot_intersections_int},knot~ diagram/draft/
2511           }]
2512       }
2513     }
```

```

2514     {
2515     \tl_set:Nn \l__knot_node_tl {\exp_not:N \node[coordinate]}
2516     }

```

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

```

2517     \int_step_variable:nnnNn {1} {1} {\l__knot_tmpa_int - 1} \l__knot_tmpa_tl
2518     {
2519     \int_step_variable:nnnNn {\tl_use:N \l__knot_tmpa_tl + 1} {1}      {\l__knot_tmpa_int}
2520     {
2521     \knot_intersections:VV \l__knot_tmpa_tl \l__knot_tmpb_tl
2522     }
2523     }

```

If any redraws were requested, do them here.

```

2524     \tl_use:N \l__knot_redraws_tl

```

Draw the crossing nodes

```

2525     \seq_use:Nn \l__knot_nodes_seq {}
2526     }

```

Close the scope

```

2527     \endpgfscope
2528     }

```

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

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

```

2529 \cs_new_protected_nopar:Npn \knot_draw_strand:n #1
2530 {
2531 \pgfscope
2532 \group_begin:
2533 \tl_set:Nn \l_tmpa_tl {knot~ diagram/every~ strand/.try,}
2534 \tl_put_right:Nv \l_tmpa_tl {\l__knot_options_strand #1}
2535 \tl_put_right:Nn \l_tmpa_tl {,knot~ diagram/only~ when~ rendering/.try,only~ when~ rendering}
2536 \spath_bake_round:n {knot strand #1}
2537 \spath_tikz_path:Vn \l_tmpa_tl {knot strand #1}
2538 \group_end:
2539 \endpgfscope
2540 }
2541 \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.

```

2542 \cs_new_protected_nopar:Npn \knot_draw_labels:n #1
2543 {
2544 \bool_if:NT \l__knot_draft_bool
2545 {
2546 \spath_get:nnN {knot strand #1} {final point} \l__knot_tmpb_tl
2547 \dim_set:Nn \l__knot_tmpa_dim {\tl_item:Nn \l__knot_tmpb_tl {1}}
2548 \dim_set:Nn \l__knot_tmpb_dim {\tl_item:Nn \l__knot_tmpb_tl {2}}
2549 \node[knot~ diagram/draft/strand~label] at (\l__knot_tmpa_dim,\l__knot_tmpb_dim) {#1};
2550 \spath_get:nnN {knot strand #1} {initial point} \l__knot_tmpb_tl
2551 \dim_set:Nn \l__knot_tmpa_dim {\tl_item:Nn \l__knot_tmpb_tl {1}}

```



```

2552     \dim_set:Nn \l__knot_tmpb_dim {\tl_item:Nn \l__knot_tmpb_tl {2}}
2553     \node[knot~ diagram/draft/strand~label] at (\l__knot_tmpa_dim,\l__knot_tmpb_dim) {#1};
2554   }
2555   \bool_if:nT {
2556     \l__knot_self_intersections_bool
2557     &&
2558     \l__knot_splits_bool
2559   }
2560   {
2561     \tl_clear:N \l__knot_tmpa_tl
2562     \spath_map_segment_function:nN {knot strand #1} \knot_split_self_intersects:NN
2563     \spath_put:nnV {knot strand #1} {path} \l__knot_tmpa_tl
2564   }
2565 }

```

(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!

```

2566 \cs_new_protected_nopar:Npn \knot_split_self_intersects:NN #1#2
2567 {
2568   \tl_case:NnF #1
2569   {
2570     \g__spath_curvetoa_tl
2571     {
2572       \fp_set:Nn \l_tmpa_fp
2573       {
2574         (\tl_item:Nn #2 {3} - 3 * \tl_item:Nn #2 {6} + 3 * \tl_item:Nn #2 {9} - \tl_item:Nn
2575         *
2576         (3 * \tl_item:Nn #2 {8} - 3 * \tl_item:Nn #2 {11}))
2577         -
2578         (\tl_item:Nn #2 {2} - 3 * \tl_item:Nn #2 {5} + 3 * \tl_item:Nn #2 {8} - \tl_item:Nn
2579         *
2580         (3 * \tl_item:Nn #2 {9} - 3 * \tl_item:Nn #2 {12}))
2581       }
2582       \fp_set:Nn \l_tmpb_fp
2583       {
2584         (\tl_item:Nn #2 {2} - 3 * \tl_item:Nn #2 {5} + 3 * \tl_item:Nn #2 {8} - \tl_item:Nn
2585         *
2586         (3 * \tl_item:Nn #2 {6} - 6 * \tl_item:Nn #2 {9} + 3 * \tl_item:Nn #2 {12}))
2587         -
2588         (\tl_item:Nn #2 {3} - 3 * \tl_item:Nn #2 {6} + 3 * \tl_item:Nn #2 {9} - \tl_item:Nn
2589         *
2590         (3 * \tl_item:Nn #2 {5} - 6 * \tl_item:Nn #2 {8} + 3 * \tl_item:Nn #2 {11}))
2591       }
2592       \fp_compare:nTF
2593       {
2594         \l_tmpb_fp != 0
2595       }
2596       {
2597         \fp_set:Nn \l_tmpa_fp {.5 * \l_tmpa_fp / \l_tmpb_fp}
2598         \fp_compare:nTF

```

```

2599     {
2600     0 < \l_tmpa_fp && \l_tmpa_fp < 1
2601     }
2602     {
2603     \spath_split_curve:VVNN \l_tmpa_fp #2 \l_tmpa_tl \l_tmpb_tl
2604     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2605     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2606     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2607     \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
2608     \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
2609     \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
2610     \tl_put_right:NV \l__knot_tmpa_tl \l_tmpa_tl
2611     \tl_put_right:NV \l__knot_tmpa_tl \l_tmpb_tl
2612     }
2613     {
2614     \tl_set_eq:NN \l_tmpa_tl #2
2615     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2616     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2617     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2618     \tl_put_right:NV \l__knot_tmpa_tl \l_tmpa_tl
2619     }
2620     }
2621     {
2622     \tl_set_eq:NN \l_tmpa_tl #2
2623     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2624     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2625     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2626     \tl_put_right:NV \l__knot_tmpa_tl \l_tmpa_tl
2627     }
2628     }
2629     \g__spath_lineto_tl
2630     {
2631     \tl_set_eq:NN \l_tmpa_tl #2
2632     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2633     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2634     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2635     \tl_put_right:NV \l__knot_tmpa_tl \l_tmpa_tl
2636     }
2637     }
2638     {
2639     \tl_put_right:NV \l__knot_tmpa_tl #2
2640     }
2641     }

```

(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.

```

2642 \cs_new_protected_nopar:Npn \knot_intersections:nn #1#2
2643 {
2644   \group_begin:
2645   \tl_set_eq:NN \l__knot_tmpa_tl \l__knot_prefix_tl
2646   \tl_put_right:Nn \l__knot_tmpa_tl {#1}
2647   \tl_set_eq:NN \l__knot_tmpb_tl \l__knot_prefix_tl
2648   \tl_put_right:Nn \l__knot_tmpb_tl {#2}

```

```

2649 \spath_get:nnN {knot \tl_use:N \l__knot_tmpa_tl} {path} \l__knot_tmpc_tl
2650 \spath_get:nnN {knot \tl_use:N \l__knot_tmpb_tl} {path} \l__knot_tmpd_tl
2651
2652 \bool_if:nTF {
2653   \l__knot_save_bool
2654   &&
2655   \tl_if_exist_p:c {knot~ intersections~ \tl_use:N \l__knot_name_tl - \tl_use:N \l__knot_t
2656 }
2657 {
2658   \tl_use:c {knot~ intersections~ \tl_use:N \l__knot_name_tl - \tl_use:N \l__knot_tmpa_tl
2659 }
2660 {
2661 \pgfintersectionofpaths{\pgfsetpath\l__knot_tmpc_tl}{\pgfsetpath\l__knot_tmpd_tl}
2662
2663 }
2664 \int_compare:nT {\pgfintersectionsolutions > 0}
2665 {
2666   \int_step_function:nnnN {1} {1} {\pgfintersectionsolutions} \knot_do_intersection:n
2667 }
2668 \knot_save_intersections:VV \l__knot_tmpa_tl \l__knot_tmpb_tl
2669 \group_end:
2670 }

```

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

\knot_save_intersections:nn

```

2671 \cs_new_protected_nopar:Npn \knot_save_intersections:nn #1#2
2672 {
2673   \bool_if:NT \l__knot_save_bool
2674   {
2675     \tl_clear:N \l__knot_aux_tl
2676     \tl_put_right:Nn \l__knot_aux_tl
2677     {
2678       \def\pgfintersectionsolutions
2679     }
2680     \tl_put_right:Nx \l__knot_aux_tl
2681     {
2682       {\int_eval:n {\pgfintersectionsolutions}}
2683     }
2684     \int_compare:nT {\pgfintersectionsolutions > 0}
2685     {
2686       \int_step_inline:nnnn {1} {1} {\pgfintersectionsolutions}
2687       {
2688         \pgfpointintersectionsolution{##1}
2689         \dim_set:Nn \l__knot_tmpa_dim {\pgf@x}
2690         \dim_set:Nn \l__knot_tmpb_dim {\pgf@y}
2691         \tl_put_right:Nn \l__knot_aux_tl
2692         {
2693           \expandafter\def\csname pgfpoint@intersect@solution@##1\endcsname
2694         }
2695         \tl_put_right:Nx \l__knot_aux_tl
2696         {
2697           {\exp_not:N \pgf@x=\dim_use:N \l__knot_tmpa_dim\exp_not:N\relax\exp_not:N \pgf@y =
2698         }

```

```

2699     }
2700     \tl_set:Nn \l__knot_auxa_tl {\expandafter \gdef \csname knot~ intersections~}
2701     \tl_put_right:Nx \l__knot_auxa_tl {\tl_use:N \l__knot_name_tl - #1 - #2}
2702     \tl_put_right:Nn \l__knot_auxa_tl {\endcsname}
2703     \tl_put_right:Nx \l__knot_auxa_tl {{\tl_to_str:N \l__knot_aux_tl}}
2704     \protected@write\auxout{{\tl_to_str:N \l__knot_auxa_tl}
2705   }
2706 }
2707 }
2708 \cs_generate_variant:Nn \knot_save_intersections:nn {VV}

```

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

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

```

2709 \cs_new_protected_nopar:Npn \knot_do_intersection:n #1
2710 {

```

Get the intersection coordinates.

```

2711 \pgfpointintersectionsolution{#1}
2712 \dim_set:Nn \l__knot_tmpa_dim {\pgf@x}
2713 \dim_set:Nn \l__knot_tmpb_dim {\pgf@y}

```

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

```

2714 \bool_set_false:N \l__knot_skip_bool
2715 \bool_if:NT \l__knot_self_intersections_bool
2716 {

```

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

```

2717 \tl_set:Nn \l_tmpa_tl {knot previous}
2718 \tl_put_right:NV \l_tmpa_tl \l__knot_tmpa_tl
2719 \tl_set:Nv \l_tmpa_tl \l_tmpa_tl
2720 \tl_if_eq:NNT \l_tmpa_tl \l__knot_tmpb_tl
2721 {
2722   \knot_test_endpoint:VnT \l__knot_tmpb_tl {final point}
2723   {
2724     \bool_set_true:N \l__knot_skip_bool
2725   }
2726 }

```

```

2727
2728 \tl_set:Nn \l_tmpa_tl {knot previous}
2729 \tl_put_right:NV \l_tmpa_tl \l__knot_tmpb_tl
2730 \tl_set:Nv \l_tmpa_tl \l_tmpa_tl
2731 \tl_if_eq:NNT \l_tmpa_tl \l__knot_tmpa_tl
2732 {
2733   \knot_test_endpoint:VnT \l__knot_tmpa_tl {final point}
2734   {
2735     \bool_set_true:N \l__knot_skip_bool
2736   }
2737 }
2738 }

```

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

```

2739 \bool_if:NT \l__knot_ignore_ends_bool
2740 {

```

```

2741 \knot_test_endpoint:VnT \l__knot_tmpa_tl {initial point}
2742 {
2743   \bool_set_true:N \l__knot_skip_bool
2744 }
2745 \knot_test_endpoint:VnT \l__knot_tmpa_tl {final point}
2746 {
2747   \bool_set_true:N \l__knot_skip_bool
2748 }
2749 \knot_test_endpoint:VnT \l__knot_tmpb_tl {initial point}
2750 {
2751   \bool_set_true:N \l__knot_skip_bool
2752 }
2753 \knot_test_endpoint:VnT \l__knot_tmpb_tl {final point}
2754 {
2755   \bool_set_true:N \l__knot_skip_bool
2756 }
2757 }

```

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

```

2758 \bool_if:NF \l__knot_skip_bool
2759 {
2760
2761   \int_gincr:N \l__knot_intersections_int

```

This is the intersection test. If the intersection finder finds too many, it might be useful to ignore some.

```

2762 \bool_if:nF
2763 {
2764   \tl_if_exist_p:c {l__knot_ignore_crossing_ \int_use:N
2765     \l__knot_intersections_int}
2766   &&
2767   ! \tl_if_empty_p:c {l__knot_ignore_crossing_ \int_use:N
2768     \l__knot_intersections_int}
2769 }
2770 {

```

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

```

2771 \bool_if:nTF
2772 {
2773   \tl_if_exist_p:c {l__knot_crossing_ \int_use:N
2774     \l__knot_intersections_int}
2775   &&
2776   ! \tl_if_empty_p:c {l__knot_crossing_ \int_use:N
2777     \l__knot_intersections_int}
2778 }
2779 {
2780   \tl_set_eq:NN \l__knot_tmpg_tl \l__knot_tmpb_tl
2781 }
2782 {
2783   \tl_set_eq:NN \l__knot_tmpg_tl \l__knot_tmpa_tl
2784 }

```

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.

```

2785 \bool_if:NT \l__knot_self_intersections_bool
2786 {
2787   \knot_test_endpoint:VnT \l__knot_tmpg_tl {initial point}
2788   {
2789     \bool_set_true:N \l__knot_prepend_prev_bool
2790   }
2791   {
2792     \bool_set_false:N \l__knot_prepend_prev_bool
2793   }
2794   \knot_test_endpoint:VnT \l__knot_tmpg_tl {final point}
2795   {
2796     \bool_set_true:N \l__knot_append_next_bool
2797   }
2798   {
2799     \bool_set_false:N \l__knot_append_next_bool
2800   }

```

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

```

2801 \bool_if:nT
2802 {
2803   \l__knot_prepend_prev_bool || \l__knot_append_next_bool
2804 }
2805 {
2806   \spath_clone:nn {knot \tl_use:N \l__knot_tmpg_tl}
2807   {knot \tl_use:N \l__knot_prefix_tl -1}
2808
2809   \tl_set_eq:cc {l__knot_options_ \tl_use:N \l__knot_prefix_tl -1} {l__knot_options_
2810
2811   \bool_if:nT
2812   {
2813     \l__knot_prepend_prev_bool
2814     &&
2815     \tl_if_exist_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
2816     &&
2817     !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
2818   }
2819   {
2820     \spath_prepend_no_move:nn {knot \tl_use:N \l__knot_prefix_tl -1} {knot \tl_use:c

```

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

```

2821 \bool_if:nT
2822 {
2823   \l__knot_splits_bool
2824   &&
2825   \tl_if_exist_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
2826   &&
2827   !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
2828 }
2829 {
2830   \knot_test_endpoint:vnT {knot previous \tl_use:N \l__knot_tmpg_tl} {initial po
2831   {
2832     \spath_get:nnN {knot \tl_use:N \l__knot_prefix_tl -1} {path} \l_tmpa_tl
2833     \spath_prepend_no_move:nn {knot \tl_use:N \l__knot_prefix_tl -
1} {knot \tl_use:c {knot previous \tl_use:c {knot previous \tl_use:N \l__knot_tmpg_tl}}}}

```

```

2834         \spath_get:nnN {knot \tl_use:N \l__knot_prefix_tl -1} {path} \l_tmpa_tl
2835     }
2836 }
2837 }
2838 }

```

Now the same for appending.

```

2839     \bool_if:nT
2840     {
2841         \l__knot_append_next_bool
2842         &&
2843         \tl_if_exist_p:c {knot next \tl_use:N \l__knot_tmpg_tl}
2844         &&
2845         !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
2846     }
2847     {
2848         \spath_append_no_move:nn {knot \tl_use:N \l__knot_prefix_tl -1} {knot \tl_use:c
2849         \bool_if:nT
2850         {
2851             \l__knot_splits_bool
2852             &&
2853             \tl_if_exist_p:c {knot previous \tl_use:N
2854             \l__knot_tmpg_tl}
2855             &&
2856             !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
2857         }
2858         {
2859             \knot_test_endpoint:vnT {knot previous \tl_use:N \l__knot_tmpg_tl} {final poin
2860             {
2861                 \spath_append_no_move:nn {knot \tl_use:N \l__knot_prefix_tl -1} {knot \tl_u
2862             }
2863         }
2864     }
2865 }
2866
2867     \tl_set:Nn \l__knot_tmpg_tl {\tl_use:N \l__knot_prefix_tl -1}
2868 }
2869 }

```

Now we render the crossing.

```

2870     \pgfscope
2871     \group_begin:
2872     \tikzset{knot~ diagram/every~ intersection/.try, every~ intersection/.try, knot~ diagr
2873     \knot_draw_crossing:nVV {\tl_use:N \l__knot_tmpg_tl} \l__knot_tmpa_dim \l__knot_tmpb_d
2874     \group_end:
2875     \endpgfscope

```

This ends the boolean as to whether to consider the intersection at all

```

2876     }

```

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

```

2877     \seq_gpush:Nx \l__knot_nodes_seq { \l__knot_node_tl (\l__knot_name_tl \c_space_tl \int_u
2878
2879     }
2880 }
2881

```

2882 `\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.

```
2883 \prg_new_conditional:Npnn \knot_test_endpoint:N #1 {p,T,F,TF}
2884 {
2885   \dim_compare:nTF
2886   {
2887     \dim_abs:n { \l__knot_tmpa_dim - \tl_item:Nn #1 {1}}
2888     +
2889     \dim_abs:n { \l__knot_tmpb_dim - \tl_item:Nn #1 {2}}
2890     <
2891     \l__knot_tolerance_dim
2892   }
2893   {
2894     \prg_return_true:
2895   }
2896   {
2897     \prg_return_false:
2898   }
2899 }
```

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

`\knot_test_endpoint:nn` Wrapper around the above.

```
2900 \prg_new_protected_conditional:Npnn \knot_test_endpoint:nn #1#2 {T,F,TF}
2901 {
2902   \spath_get:nnN {knot #1} {#2} \l__knot_tmpd_tl
2903   \knot_test_endpoint:NpTF \l__knot_tmpd_tl
2904   {
2905     \prg_return_true:
2906   }
2907   {
2908     \prg_return_false:
2909   }
2910 }
2911
2912 \cs_generate_variant:Nn \knot_test_endpoint:nnT {VnT,vnT}
2913 \cs_generate_variant:Nn \knot_test_endpoint:nnF {VnF,vnF}
2914 \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.

```
2915 \cs_new_nopar:Npn \knot_draw_crossing:nnn #1#2#3
2916 {
2917   \group_begin:
2918   \pgfscope
2919   \path[knot~ diagram/background~ clip] (#2, #3) circle[radius=\l__knot_clip_bg_radius_dim];
2920
2921   \tl_set:Nn \l_tmpa_tl {knot~ diagram/every~ strand/.try,}
2922   \tl_if_exist:cT {l__knot_options_ #1}
2923   {
2924     \tl_put_right:Nv \l_tmpa_tl {l__knot_options_ #1}
```



```

2925 }
2926 \tl_put_right:Nn \l_tmpa_tl {,knotbg,line~ width= \tl_use:N \l__knot_clip_width_tl * \pgfl
2927 \spath_tikz_path:Vn \l_tmpa_tl {knot #1}
2928
2929 \endpgfscope
2930
2931 \pgfscope
2932 \path[knot~ diagram/clip] (#2, #3) circle[radius=\l__knot_clip_draw_radius_dim];
2933
2934 \tl_set:Nn \l_tmpa_tl {knot~ diagram/every~ strand/.try,}
2935 \tl_if_exist:cT {l__knot_options_ #1}
2936 {
2937 \tl_put_right:Nv \l_tmpa_tl {l__knot_options_ #1}
2938 }
2939 \tl_put_right:Nn \l_tmpa_tl {,knot~ diagram/only~ when~ rendering/.try,only~ when~ renderi
2940 \spath_tikz_path:Vn \l_tmpa_tl {knot #1}
2941
2942 \endpgfscope
2943 \group_end:
2944 }
2945
2946 \cs_generate_variant:Nn \knot_draw_crossing:nnn {nVV}
2947
2948 \cs_new_nopar:Npn \knot_draw_crossing:nn #1#2
2949 {
2950 \tikz@scan@one@point\pgfutil@firstofone #2 \relax
2951 \knot_draw_crossing:nVV {#1} \pgf@x \pgf@y
2952 }

```

(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.

```

2953 \cs_new_protected_nopar:Npn \knot_split_strands:
2954 {
2955 \int_gzero:N \l__knot_filaments_int
2956 \int_step_function:nnnN {1} {1} {\l__knot_strands_int} \knot_split_strand:n
2957 \int_step_function:nnnN {1} {1} {\l__knot_filaments_int} \knot_compute_nexts:n
2958 }

```

(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.

```

2959 \cs_new_protected_nopar:Npn \knot_compute_nexts:n #1
2960 {
2961 \tl_clear_new:c {knot next \tl_use:c {knot previous filament #1}}
2962 \tl_set:cn {knot next \tl_use:c {knot previous filament #1}} {filament #1}
2963 }

```

(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.

```

2964 \cs_new_protected_nopar:Npn \knot_split_strand:n #1
2965 {
2966 \int_set_eq:NN \l__knot_component_start_int \l__knot_filaments_int

```

```

2967 \int_incr:N \l__knot_component_start_int
2968 \tl_set_eq:Nc \l__knot_tmpa_tl {l__knot_options_strand #1}
2969 \spath_map_segment_function:nN {knot strand #1} \knot_save_filament:NN
2970 }

```

(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.

```

2971 \cs_new_protected_nopar:Npn \knot_save_filament:NN #1#2
2972 {
2973   \tl_case:NnF #1
2974   {
2975     \g__spath_moveto_tl
2976     {
2977       \int_compare:nT {\l__knot_component_start_int < \l__knot_filaments_int}
2978       {
2979         \int_set_eq:NN \l__knot_component_start_int \l__knot_filaments_int
2980       }
2981     }
2982     \g__spath_lineto_tl
2983     {
2984       \int_gincr:N \l__knot_filaments_int
2985       \spath_clear_new:n {knot filament \int_use:N \l__knot_filaments_int}
2986       \spath_put:nnV {knot filament \int_use:N \l__knot_filaments_int} {path} #2
2987       \tl_set_eq:cN {l__knot_options_filament \int_use:N \l__knot_filaments_int} \l__knot_tm
2988     }
2989     \tl_clear_new:c {knot previous filament \int_use:N \l__knot_filaments_int}
2990     \int_compare:nF {\l__knot_component_start_int == \l__knot_filaments_int}
2991     {
2992       \tl_set:cx {knot previous filament \int_use:N \l__knot_filaments_int} {filament \int
2993     }
2994   }
2995   \g__spath_curvetoa_tl
2996   {
2997     \int_gincr:N \l__knot_filaments_int
2998     \spath_clear_new:n {knot filament \int_use:N \l__knot_filaments_int}
2999     \spath_put:nnV {knot filament \int_use:N \l__knot_filaments_int} {path} #2
3000     \tl_set_eq:cN {l__knot_options_filament \int_use:N \l__knot_filaments_int} \l__knot_tm
3001   }
3002   \tl_clear_new:c {knot previous filament \int_use:N \l__knot_filaments_int}
3003   \int_compare:nF {\l__knot_component_start_int == \l__knot_filaments_int}
3004   {
3005     \tl_set:cx {knot previous filament \int_use:N \l__knot_filaments_int} {filament \int
3006   }
3007 }
3008 \g__spath_closepath_tl
3009 {
3010   \tl_show:N #2
3011   \int_gincr:N \l__knot_filaments_int
3012   \spath_clear_new:n {knot filament \int_use:N \l__knot_filaments_int}
3013   \tl_clear:N \l_tmpa_tl
3014   \tl_put_right:Nx {\tl_item:Nn #2 {1}}\tl_item:Nn #2 {2}}\tl_item:Nn #2 {3}}
3015   \tl_put_right:NV \l_tmpa_tl \g__spath_lineto_tl
3016   \tl_put_right:Nx {\tl_item:Nn #2 {5}}\tl_item:Nn #2 {6}}

```

```

3017     \tl_show:N \l_tmpa_tl
3018     \spath_put:nnV {knot filament \int_use:N \l__knot_filaments_int} {path} \l_tmpa_tl
3019     \tl_set_eq:cN {l__knot_options_filament \int_use:N \l__knot_filaments_int} \l__knot_tm
3020     \tl_clear_new:c {knot previous filament \int_use:N \l__knot_filaments_int}
3021     \int_compare:nF {\l__knot_component_start_int == \l__knot_filaments_int}
3022     {
3023       \tl_set:cx {knot previous filament \int_use:N \l__knot_filaments_int} {filament \int
3024     }
3025     \tl_set:cx {knot previous filament \int_use:N \l__knot_component_start_int} {filament
3026   }
3027   }
3028   {
3029   }
3030 }

```

(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.

```

3031 \NewDocumentCommand \redraw { m m }
3032 {
3033 % \tikz@scan@one@point\pgfutil@firstofone #2 \relax
3034 \tl_put_right:Nn \l__knot_redraws_tl {\knot_draw_crossing:nn}
3035 \tl_put_right:Nx \l__knot_redraws_tl {
3036   {strand #1} {#2}% {\dim_use:N \pgf@x} {\dim_use:N \pgf@y}
3037 }
3038 }

```

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

```

3039 \ExplSyntaxOff

```

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

```

3040 \def\pgf@sh@@knotanchor#1#2{%
3041   \anchor{#2 north west}{%
3042     \csname pgf@anchor@knot #1@north west\endcsname%
3043     \pgf@x=#2\pgf@x%
3044     \pgf@y=#2\pgf@y%
3045   }%
3046   \anchor{#2 north east}{%
3047     \csname pgf@anchor@knot #1@north east\endcsname%
3048     \pgf@x=#2\pgf@x%
3049     \pgf@y=#2\pgf@y%
3050   }%
3051   \anchor{#2 south west}{%
3052     \csname pgf@anchor@knot #1@south west\endcsname%
3053     \pgf@x=#2\pgf@x%
3054     \pgf@y=#2\pgf@y%
3055   }%
3056   \anchor{#2 south east}{%
3057     \csname pgf@anchor@knot #1@south east\endcsname%
3058     \pgf@x=#2\pgf@x%
3059     \pgf@y=#2\pgf@y%
3060   }%
3061   \anchor{#2 north}{%
3062     \csname pgf@anchor@knot #1@north\endcsname%

```

```

3063     \pgf@x=#2\pgf@x%
3064     \pgf@y=#2\pgf@y%
3065   }%
3066   \anchor{#2 east}{%
3067     \csname pgf@anchor@knot #1@east\endcsname%
3068     \pgf@x=#2\pgf@x%
3069     \pgf@y=#2\pgf@y%
3070   }%
3071   \anchor{#2 west}{%
3072     \csname pgf@anchor@knot #1@west\endcsname%
3073     \pgf@x=#2\pgf@x%
3074     \pgf@y=#2\pgf@y%
3075   }%
3076   \anchor{#2 south}{%
3077     \csname pgf@anchor@knot #1@south\endcsname%
3078     \pgf@x=#2\pgf@x%
3079     \pgf@y=#2\pgf@y%
3080   }%
3081 }

```

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

knot_crossing

```

3082 \pgfdeclareshape{knot crossing}
3083 {
3084   \inheritssavedanchors[from=circle] % this is nearly a circle
3085   \inheritanchorborder[from=circle]
3086   \inheritanchor[from=circle]{north}
3087   \inheritanchor[from=circle]{north west}
3088   \inheritanchor[from=circle]{north east}
3089   \inheritanchor[from=circle]{center}
3090   \inheritanchor[from=circle]{west}
3091   \inheritanchor[from=circle]{east}
3092   \inheritanchor[from=circle]{mid}
3093   \inheritanchor[from=circle]{mid west}
3094   \inheritanchor[from=circle]{mid east}
3095   \inheritanchor[from=circle]{base}
3096   \inheritanchor[from=circle]{base west}
3097   \inheritanchor[from=circle]{base east}
3098   \inheritanchor[from=circle]{south}
3099   \inheritanchor[from=circle]{south west}
3100   \inheritanchor[from=circle]{south east}
3101   \inheritanchorborder[from=circle]
3102   \pgf@sh@@knotanchor{crossing}{2}
3103   \pgf@sh@@knotanchor{crossing}{3}
3104   \pgf@sh@@knotanchor{crossing}{4}
3105   \pgf@sh@@knotanchor{crossing}{8}
3106   \pgf@sh@@knotanchor{crossing}{16}
3107   \pgf@sh@@knotanchor{crossing}{32}
3108   \backgroundpath{
3109     \pgfutil@tempdima=\radius%
3110     \pgfmathsetlength{\pgf@xb}{\pgfkeysvalueof{/pgf/outer xsep}}%
3111     \pgfmathsetlength{\pgf@yb}{\pgfkeysvalueof{/pgf/outer ysep}}%
3112     \ifdim\pgf@xb<\pgf@yb%

```

```

3113     \advance\pgfutil@tempdima by-\pgf@yb%
3114     \else%
3115     \advance\pgfutil@tempdima by-\pgf@xb%
3116     \fi%
3117   }
3118 }

```

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

knot_over_cross

```

3119 \pgfdeclareshape{knot over cross}
3120 {
3121   \inheritsavedanchors[from=rectangle] % this is nearly a circle
3122   \inheritanchorborder[from=rectangle]
3123   \inheritanchor[from=rectangle]{north}
3124   \inheritanchor[from=rectangle]{north west}
3125   \inheritanchor[from=rectangle]{north east}
3126   \inheritanchor[from=rectangle]{center}
3127   \inheritanchor[from=rectangle]{west}
3128   \inheritanchor[from=rectangle]{east}
3129   \inheritanchor[from=rectangle]{mid}
3130   \inheritanchor[from=rectangle]{mid west}
3131   \inheritanchor[from=rectangle]{mid east}
3132   \inheritanchor[from=rectangle]{base}
3133   \inheritanchor[from=rectangle]{base west}
3134   \inheritanchor[from=rectangle]{base east}
3135   \inheritanchor[from=rectangle]{south}
3136   \inheritanchor[from=rectangle]{south west}
3137   \inheritanchor[from=rectangle]{south east}
3138   \inheritanchorborder[from=rectangle]
3139   \backgroundpath{
3140     \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
3141     \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
3142     \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
3143     \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
3144   }
3145   \foregroundpath{
3146     % store lower right in xa/ya and upper right in xb/yb
3147     \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
3148     \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
3149     \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
3150     \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
3151   }
3152 }

```

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

knot_under_cross

```

3153 \pgfdeclareshape{knot under cross}
3154 {
3155   \inheritsavedanchors[from=rectangle] % this is nearly a circle
3156   \inheritanchorborder[from=rectangle]
3157   \inheritanchor[from=rectangle]{north}
3158   \inheritanchor[from=rectangle]{north west}
3159   \inheritanchor[from=rectangle]{north east}

```

```

3160 \inheritanchor[from=rectangle]{center}
3161 \inheritanchor[from=rectangle]{west}
3162 \inheritanchor[from=rectangle]{east}
3163 \inheritanchor[from=rectangle]{mid}
3164 \inheritanchor[from=rectangle]{mid west}
3165 \inheritanchor[from=rectangle]{mid east}
3166 \inheritanchor[from=rectangle]{base}
3167 \inheritanchor[from=rectangle]{base west}
3168 \inheritanchor[from=rectangle]{base east}
3169 \inheritanchor[from=rectangle]{south}
3170 \inheritanchor[from=rectangle]{south west}
3171 \inheritanchor[from=rectangle]{south east}
3172 \inheritanchorborder[from=rectangle]
3173 \backgroundpath{
3174   \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
3175   \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
3176   \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
3177   \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
3178 }
3179 \foregroundpath{
3180 % store lower right in xa/ya and upper right in xb/yb
3181   \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
3182   \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
3183   \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
3184   \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
3185 }
3186 }

```

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

`knot⊥vert`

```

3187 \pgfdeclareshape{knot vert}
3188 {
3189   \inheritshapedanchors[from=rectangle] % this is nearly a circle
3190   \inheritanchorborder[from=rectangle]
3191   \inheritanchor[from=rectangle]{north}
3192   \inheritanchor[from=rectangle]{north west}
3193   \inheritanchor[from=rectangle]{north east}
3194   \inheritanchor[from=rectangle]{center}
3195   \inheritanchor[from=rectangle]{west}
3196   \inheritanchor[from=rectangle]{east}
3197   \inheritanchor[from=rectangle]{mid}
3198   \inheritanchor[from=rectangle]{mid west}
3199   \inheritanchor[from=rectangle]{mid east}
3200   \inheritanchor[from=rectangle]{base}
3201   \inheritanchor[from=rectangle]{base west}
3202   \inheritanchor[from=rectangle]{base east}
3203   \inheritanchor[from=rectangle]{south}
3204   \inheritanchor[from=rectangle]{south west}
3205   \inheritanchor[from=rectangle]{south east}
3206   \inheritanchorborder[from=rectangle]
3207   \backgroundpath{
3208 % store lower right in xa/ya and upper right in xb/yb
3209   \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y

```

```

3210 \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
3211 \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
3212 \pgfpathlineto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
3213 \pgfpathmoveto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
3214 \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
3215 }
3216 }

```

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

knot_horiz

```

3217 \pgfdeclareshape{knot horiz}
3218 {
3219 \inheritsavedanchors[from=rectangle] % this is nearly a circle
3220 \inheritanchorborder[from=rectangle]
3221 \inheritanchor[from=rectangle]{north}
3222 \inheritanchor[from=rectangle]{north west}
3223 \inheritanchor[from=rectangle]{north east}
3224 \inheritanchor[from=rectangle]{center}
3225 \inheritanchor[from=rectangle]{west}
3226 \inheritanchor[from=rectangle]{east}
3227 \inheritanchor[from=rectangle]{mid}
3228 \inheritanchor[from=rectangle]{mid west}
3229 \inheritanchor[from=rectangle]{mid east}
3230 \inheritanchor[from=rectangle]{base}
3231 \inheritanchor[from=rectangle]{base west}
3232 \inheritanchor[from=rectangle]{base east}
3233 \inheritanchor[from=rectangle]{south}
3234 \inheritanchor[from=rectangle]{south west}
3235 \inheritanchor[from=rectangle]{south east}
3236 \inheritanchorborder[from=rectangle]
3237 \foregroundpath{
3238 % store lower right in xa/ya and upper right in xb/yb
3239 \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
3240 \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
3241 \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
3242 \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
3243 \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
3244 \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
3245 }
3246 }

```

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