

# The `spath3` package

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

v1.0 from 2013/04/09

## 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} {2013/04/09} {1.0} {Functions for
5 manipulating PGF soft paths}
6 \RequirePackage{xparse}
```

We need a slew of temporary variables.

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

```

11 \tl_new:N \l__spath_smuggle_tl
12 \dim_new:N \l__spath_tmpa_dim
13 \dim_new:N \l__spath_tmpb_dim
14 \int_new:N \l__spath_tmpa_int

```

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

```

15 \tl_new:N \g__spath_moveto_tl
16 \tl_new:N \g__spath_lineto_tl
17 \tl_new:N \g__spath_curveto_tl
18 \tl_new:N \g__spath_curvetoa_tl
19 \tl_new:N \g__spath_curvetob_tl
20 \tl_set:Nn \g__spath_moveto_tl {\pgfsyssoftpath@movetotoken}
21 \tl_set:Nn \g__spath_lineto_tl {\pgfsyssoftpath@linetotoken}
22 \tl_set:Nn \g__spath_curveto_tl {\pgfsyssoftpath@curvetotoken}
23 \tl_set:Nn \g__spath_curvetoa_tl {\pgfsyssoftpath@curvetosupportatoken}
24 \tl_set:Nn \g__spath_curvetob_tl {\pgfsyssoftpath@curvetosupportbtoken}

```

## 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.

```

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

```

```

49   {reverse path}
50 }

```

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`

```

51 \cs_new_nopar:Npn \spath_new:n #1
52 {
53   \prop_new:c {l__spath_#1}
54 }

```

`\spath_clear:n`

```

55 \cs_new_nopar:Npn \spath_clear:n #1
56 {
57   \prop_gclear:c {l__spath_#1}
58 }

```

`\spath_clear_new:n`

```

59 \cs_new_nopar:Npn \spath_clear_new:n #1
60 {
61   \prop_gclear_new:c {l__spath_#1}
62 }

```

`\spath_show:n`

```

63 \cs_new_nopar:Npn \spath_show:n #1
64 {
65   \prop_show:c {l__spath_#1}
66 }

```

`\spath_put:nnn`

```

67 \cs_new_nopar:Npn \spath_put:nnn #1#2#3
68 {
69   \prop_gput:cnn {l__spath_#1} {#2} {#3}
70 }

```

`\spath_remove:nn`

```

71 \cs_new_nopar:Npn \spath_remove:nn #1#2
72 {
73   \prop_gremove:cn {l__spath_#1} {#2}
74 }

```

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

```

75 \cs_new_nopar:Npn \__spath_get:nn #1#2
76 {
77   \prop_get:cn {l__spath_#1} {#2}
78 }

```

`\_spath_get:nnN`

```
79 \cs_new_nopar:Npn \_spath_get:nnN #1#2#3
80 {
81   \prop_get:cnN {l__spath_#1} {#2} #3
82 }
```

`\spath_if_in:nn`

```
83 \prg_new_conditional:Npnn \spath_if_in:nn #1#2 {p, T, F, TF}
84 {
85   \prop_if_in:cnTF {l__spath_#1} {#2}
86   { \prg_return_true: }
87   { \prg_return_false: }
88 }
```

`\_spath_get:nnN`

```
89 \cs_generate_variant:Nn \_prop_split:NnTF {cnTF}
90 \prg_new_protected_conditional:Npnn \_spath_get:nnN #1#2#3 {T, F, TF}
91 {
92   \_prop_split:cnTF {l__spath_#1} {#2}
93   { \_prop_get_true:Nnnn #3 }
94   { \prg_return_false: }
95 }
96 \cs_generate_variant:Nn \spath_put:nnn {nnV, nnx, nno}
97 \cs_generate_variant:Nn \_spath_get:nn {Vn}
```

`\spath_if_exist:n`

```
98 \prg_new_conditional:Npnn \spath_if_exist:n #1 {p,T,F,TF}
99 {
100   \prop_if_exist:cTF {l__spath_#1}
101   {
102     \prg_return_true:
103   }
104   {
105     \prg_return_false:
106   }
107 }
```

`\spath_clone:nn` Clones an spath.

```
108 \cs_new_nopar:Npn \spath_clone:nn #1 #2
109 {
110   \spath_clear_new:n {#2}
111   \tl_map_inline:Nn \g__spath_attributes
112   {
113     \spath_if_in:nnT {#1} {##1}
114     {
115       \_spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
116       \spath_put:nnV {#2} {##1} \l__spath_tmpa_tl
117     }
118   }
119 }
```

`\spath_get_current_path:n`

```
120 \cs_new_protected_nopar:Npn \spath_get_current_path:n #1
121 {
122   \pgfsyssoftpath@getcurrentpath\l__knot_tmpa_tl
123   \spath_clear_new:n {#1}
124   \spath_put:nnV {#1} {path} \l__knot_tmpa_tl
125 }
```

`\spath_set_current_path:n`

```
126 \cs_new_protected_nopar:Npn \spath_set_current_path:n #1
127 {
128   \spath_get:nnN {#1} {min bb} \l__spath_tmpa_tl
129   \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl
130
131   \spath_get:nnN {#1} {max bb} \l__spath_tmpa_tl
132   \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl
133
134   \spath_get:nnN {#1} {path} \l__spath_tmpa_tl
135   \pgfsyssoftpath@setcurrentpath\l__spath_tmpa_tl
136   \pgfsyssoftpath@flushcurrentpath
137 }
```

`\spath_use_path:nn`

```
138 \cs_new_protected_nopar:Npn \spath_use_path:nn #1#2
139 {
140   \spath_set_current_path:n {#1}
141   \pgfusepath{#2}
142 }
```

`\spath_protocol_path:n`

```
143 \cs_new_protected_nopar:Npn \spath_protocol_path:n #1
144 {
145   \spath_get:nnN {#1} {min bb} \l__spath_tmpa_tl
146   \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl
147
148   \spath_get:nnN {#1} {max bb} \l__spath_tmpa_tl
149   \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl
150 }
```

`\spath_tikz_path:nn`

```
151 \cs_new_protected_nopar:Npn \spath_tikz_path:nn #1#2
152 {
153   \path[#1] \pgfextra{
154     \spath_get:nnN {#2} {path} \l__spath_tmpa_tl
155     \pgfsyssoftpath@setcurrentpath \l__spath_tmpa_tl
156   };
157 }
158 \cs_generate_variant:Nn \spath_tikz_path:nn {Vn}
```

## 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.

```

159 \cs_new_nopar:Npn \spath_get:nn #1#2
160 {
161   \spath_if_in:nnF {#1} {#2}
162   {
163     \cs_if_exist_use:cT {spath_generate_#2:n} {#{#1}}
164   }
165   \__spath_get:nn {#1} {#2}
166 }

```

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

```

167 \cs_new_nopar:Npn \spath_get:nnN #1#2#3
168 {
169   \spath_if_in:nnF {#1} {#2}
170   {
171     \cs_if_exist_use:cT {spath_generate_#2:n} {#{#1}}
172   }
173   \__spath_get:nnN {#1} {#2} #3
174 }
175 \cs_generate_variant:Nn \spath_get:nnN {nnV}

```

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.

```

176 \cs_new_nopar:Npn \spath_generate_length:n #1
177 {
178   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
179   \spath_put:nx {#1} {length} {\int_eval:n {\tl_count:N \l__spath_tmpa_tl /3 }}
180 }

```

`\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).

```

181 \cs_new_nopar:Npn \spath_generate_reallength:n #1
182 {
183   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
184   \int_set:Nn \l__spath_tmpa_int {0}
185   \tl_map_inline:Nn \l__spath_tmpa_tl {
186     \tl_if_eq:nnT {##1} {\pgfsyssoftpath@linetotoken}
187     {
188       \int_incr:N \l__spath_tmpa_int
189     }
190     \tl_if_eq:nnT {##1} {\pgfsyssoftpath@curvetotoken}
191     {
192       \int_incr:N \l__spath_tmpa_int
193     }
194   }

```

```

194 }
195 \spath_put:nx {#1} {real length} {\int_use:N \l__spath_tmpa_int}
196 }

```

`\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.

```

197 \cs_new_nopar:Npn \spath_generate_numberofcomponents:n #1
198 {
199   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
200   \int_set:Nn \l__spath_tmpa_int {0}
201   \tl_map_inline:Nn \l__spath_tmpa_tl {
202     \tl_if_eq:nnT {##1} {\pgfsyssoftpath@movetotoken}
203     {
204       \int_incr:N \l__spath_tmpa_int
205     }
206   }
207   \spath_put:nx {#1} {number of components} {\int_use:N \l__spath_tmpa_int}
208 }

```

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

```

209 \cs_new_nopar:Npn \spath_generate_initialpoint:n #1
210 {
211   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
212   \tl_clear:N \l__spath_tmpb_tl
213   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
214   \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
215   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
216   \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
217   \spath_put:nnV {#1} {initial point} \l__spath_tmpb_tl
218 }

```

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

```

219 \cs_new_nopar:Npn \spath_generate_finalpoint:n #1
220 {
221   \tl_clear:N \l__spath_tmpb_tl
222   \spath_if_in:nnTF {#1} {reverse path}
223   {
224     \__spath_get:nnN {#1} {reverse path} \l__spath_tmpa_tl
225     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
226     \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
227     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
228     \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
229   }
230   {
231     \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
232     \tl_reverse:N \l__spath_tmpa_tl
233     \tl_put_left:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
234     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
235     \tl_put_left:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}

```

```

236 }
237 \spath_put:nnV {#1} {final point} \l__spath_tmpb_tl
238 }

```

\spath\_generate\_reversepath:n This computes the reverse of the path. TODO: handle closed paths, possibly rectangles.

```

239 \cs_new_nopar:Npn \spath_generate_reversepath:n #1
240 {
241   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
242   \tl_clear:N \l__spath_tmpb_tl
243   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
244   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
245   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
246   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
247   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
248   \tl_put_left:Nx \l__spath_tmpb_tl
249   {
250     {\dim_use:N \l__spath_tmpa_dim}
251     {\dim_use:N \l__spath_tmpb_dim}
252   }
253   \bool_until_do:nn {
254     \tl_if_empty_p:N \l__spath_tmpa_tl
255   }
256   {
257     \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
258     \tl_set:Nn \l__spath_tmpd_tl {}
259     \tl_case:Nnn \l__spath_tmpc_tl
260     {
261       \g__spath_moveto_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_moveto_tl }
262       \g__spath_lineto_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_lineto_tl }
263       \g__spath_curveto_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curvetoa_tl }
264       \g__spath_curvetoa_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curveto_tl }
265       \g__spath_curvetob_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curvetob_tl }
266     }
267     {
268       \tl_show:N \l__spath_tmpc_tl
269     }
270     \tl_put_left:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
271     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
272     \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
273     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
274     \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
275     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
276     \tl_put_left:Nx \l__spath_tmpb_tl
277     {
278       {\dim_use:N \l__spath_tmpa_dim}
279       {\dim_use:N \l__spath_tmpb_dim}
280     }
281   }
282   \tl_put_left:NV \l__spath_tmpb_tl \g__spath_moveto_tl
283   \spath_put:nnV {#1} {reverse path} \l__spath_tmpb_tl

```



284 }

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

```
285 \cs_new_nopar:Npn \spath_generate_initialaction:n #1
286 {
287   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
288   \tl_clear:N \l__spath_tmpb_tl
289   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
290   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
291   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
292   \tl_if_empty:NF \l__spath_tmpa_tl {
293     \tl_set:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
294   }
295   \spath_put:nnV {#1} {initial action} \l__spath_tmpb_tl
296 }
```

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

```
297 \cs_new_nopar:Npn \spath_generate_finalaction:n #1
298 {
299   \tl_clear:N \l__spath_tmpb_tl
300   \spath_if_in:nnTF {#1} {reverse path}
301   {
302     \__spath_get:nnN {#1} {reverse path} \l__spath_tmpa_tl
303     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
304   }
305   {
306     \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
307     \tl_reverse:N \l__spath_tmpa_tl
308   }
309   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
310   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
311   \tl_if_empty:NF \l__spath_tmpa_tl {
312     \tl_set:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
313   }
314   \tl_if_eq:NNT \l__spath_tmpa_tl \g__spath_curvetoa_tl
315   {
316     \tl_set_eq:NN \l__spath_tmpa_tl \g__spath_curveto_tl
317   }
318   \spath_put:nnV {#1} {final action} \l__spath_tmpb_tl
319 }
```

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

```
320 \cs_new_nopar:Npn \spath_generate_minbb:n #1
321 {
322   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
323   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
324   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
325   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
326   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
```

```

327 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
328 \bool_until_do:nn {
329   \tl_if_empty_p:N \l__spath_tmpa_tl
330 }
331 {
332   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
333   \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tmpa_tl}}
334   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
335   \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tmpa_tl}}
336   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
337 }
338 \tl_clear:N \l__spath_tmpb_tl
339 \tl_put_right:Nx \l__spath_tmpb_tl
340 {
341   {\dim_use:N \l__spath_tmpa_dim}
342   {\dim_use:N \l__spath_tmpb_dim}
343 }
344 \spath_put:nnV {#1} {min bb} \l__spath_tmpb_tl
345 }

```

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

```

346 \cs_new_nopar:Npn \spath_generate_maxbb:n #1
347 {
348   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
349   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
350   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
351   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
352   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
353   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
354   \bool_until_do:nn {
355     \tl_if_empty_p:N \l__spath_tmpa_tl
356   }
357   {
358     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
359     \dim_set:Nn \l__spath_tmpa_dim {\dim_max:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tmpa_tl}}
360     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
361     \dim_set:Nn \l__spath_tmpb_dim {\dim_max:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tmpa_tl}}
362     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
363   }
364   \tl_clear:N \l__spath_tmpb_tl
365   \tl_put_right:Nx \l__spath_tmpb_tl
366   {
367     {\dim_use:N \l__spath_tmpa_dim}
368     {\dim_use:N \l__spath_tmpb_dim}
369   }
370   \spath_put:nnV {#1} {max bb} \l__spath_tmpb_tl
371 }

```

`\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

```
372 \tl_new:N \l__spath_rp_tl
373 \int_new:N \l__spath_l_int
374 \int_new:N \l__spath_rl_int
375 \int_new:N \l__spath_nc_int
376 \tl_new:N \l__spath_ip_tl
377 \tl_new:N \l__spath_fp_tl
378 \tl_new:N \l__spath_ia_tl
379 \tl_new:N \l__spath_fa_tl
380 \dim_new:N \l__spath_minx_dim
381 \dim_new:N \l__spath_miny_dim
382 \dim_new:N \l__spath_maxx_dim
383 \dim_new:N \l__spath_maxy_dim
384
385 \cs_new_nopar:Npn \spath_generate_all:n #1
386 {
387   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
388
389   \tl_clear:N \l__spath_rp_tl
390   \int_set:Nn \l__spath_l_int {1}
391   \int_zero:N \l__spath_rl_int
392   \int_set:Nn \l__spath_nc_int {1}
393   \tl_clear:N \l__spath_ip_tl
394   \tl_clear:N \l__spath_fp_tl
395   \tl_clear:N \l__spath_ia_tl
396   \tl_clear:N \l__spath_fa_tl
397   \dim_zero:N \l__spath_minx_dim
398   \dim_zero:N \l__spath_miny_dim
```

```

399 \dim_zero:N \l__spath_maxx_dim
400 \dim_zero:N \l__spath_maxy_dim
401
402 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
403 \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
404 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
405 \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
406 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
407
408 \tl_clear:N \l__spath_ip_tl
409 \tl_put_right:Nx \l__spath_ip_tl
410 {
411   {\dim_use:N \l__spath_tmpa_dim}
412   {\dim_use:N \l__spath_tmpb_dim}
413 }
414
415 \dim_set_eq:NN \l__spath_minx_dim \l__spath_tmpa_dim
416 \dim_set_eq:NN \l__spath_miny_dim \l__spath_tmpb_dim
417 \dim_set_eq:NN \l__spath_maxx_dim \l__spath_tmpa_dim
418 \dim_set_eq:NN \l__spath_maxy_dim \l__spath_tmpb_dim
419
420 \tl_put_left:Nx \l__spath_rp_tl
421 {
422   {\dim_use:N \l__spath_tmpa_dim}
423   {\dim_use:N \l__spath_tmpb_dim}
424 }
425
426 \tl_set:Nx \l__spath_ia_tl {\tl_head:N \l__spath_tmpa_tl}
427
428 \bool_until_do:nn {
429   \tl_if_empty_p:N \l__spath_tmpa_tl
430 }
431 {
432   \tl_set:Nx \l__spath_tmpe_tl {\tl_head:N \l__spath_tmpa_tl}
433   \tl_set:Nn \l__spath_tmpe_tl {}
434   \tl_set_eq:NN \l__spath_fa_tl \l__spath_tmpe_tl
435   \int_incr:N \l__spath_l_int
436
437   \tl_case:Nnn \l__spath_tmpe_tl
438   {
439     \g__spath_moveto_tl {
440       \tl_set_eq:NN \l__spath_tmpe_tl \g__spath_moveto_tl
441       \int_incr:N \l__spath_nc_int
442     }
443     \g__spath_lineto_tl {
444       \tl_set_eq:NN \l__spath_tmpe_tl \g__spath_lineto_tl
445       \int_incr:N \l__spath_rl_int
446     }
447     \g__spath_curveto_tl {
448       \tl_set_eq:NN \l__spath_tmpe_tl \g__spath_curvetoa_tl

```

```

449     \int_incr:N \l__spath_rl_int
450   }
451   \g__spath_curvetoa_tl {
452     \tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curveto_tl
453   }
454   \g__spath_curvetob_tl {
455     \tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curvetob_tl
456   }
457 }
458 {
459   \tl_show:N \l__spath_tmpc_tl
460 }
461 \tl_put_left:NV \l__spath_rp_tl \l__spath_tmpd_tl
462 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
463
464 \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
465 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
466 \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
467 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
468
469 \dim_set:Nn \l__spath_minx_dim {\dim_min:nn { \l__spath_minx_dim} {\l__spath_tmpa_dim}}
470 \dim_set:Nn \l__spath_miny_dim {\dim_min:nn { \l__spath_miny_dim} {\l__spath_tmpb_dim}}
471 \dim_set:Nn \l__spath_maxx_dim {\dim_max:nn { \l__spath_maxx_dim} {\l__spath_tmpa_dim}}
472 \dim_set:Nn \l__spath_maxy_dim {\dim_max:nn { \l__spath_maxy_dim} {\l__spath_tmpb_dim}}
473
474 \tl_put_left:Nx \l__spath_rp_tl
475 {
476   {\dim_use:N \l__spath_tmpa_dim}
477   {\dim_use:N \l__spath_tmpb_dim}
478 }
479
480 \tl_set:Nx \l__spath_fp_tl
481 {
482   {\dim_use:N \l__spath_tmpa_dim}
483   {\dim_use:N \l__spath_tmpb_dim}
484 }
485
486 }
487
488 \tl_put_left:NV \l__spath_rp_tl \g__spath_moveto_tl
489
490 \spath_put:nnV {#1} {reverse path} \l__spath_rp_tl
491 \spath_put:nnV {#1} {length} \l__spath_l_int
492 \spath_put:nnV {#1} {real length} \l__spath_rl_int
493 \spath_put:nnV {#1} {number of components} \l__spath_nc_int
494 \spath_put:nnV {#1} {initial point} \l__spath_ip_tl
495 \spath_put:nnV {#1} {final point} \l__spath_fp_tl
496 \spath_put:nnV {#1} {initial action} \l__spath_ia_tl
497 \spath_put:nnV {#1} {final action} \l__spath_fa_tl
498

```

```

499 \tl_clear:N \l__spath_tmpb_tl
500 \tl_put_right:Nx \l__spath_tmpb_tl
501 {
502   {\dim_use:N \l__spath_minx_dim}
503   {\dim_use:N \l__spath_miny_dim}
504 }
505 \spath_put:nnV {#1} {min bb} \l__spath_tmpb_tl
506
507 \tl_clear:N \l__spath_tmpb_tl
508 \tl_put_right:Nx \l__spath_tmpb_tl
509 {
510   {\dim_use:N \l__spath_maxx_dim}
511   {\dim_use:N \l__spath_maxy_dim}
512 }
513 \spath_put:nnV {#1} {max bb} \l__spath_tmpb_tl
514
515 }

```

## 2.4 Path Manipulation

`\spath_translate:nnn` Translates a path.

```

516 \cs_new_nopar:Npn \spath_translate:nnn #1#2#3
517 {
518   \tl_map_inline:Nn \g__spath_moveable_attributes
519   {
520     \spath_if_in:nnT {#1} {##1}
521     {
522       \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
523
524       \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl + #2}
525       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
526       \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl + #3}
527       \tl_clear:N \l__spath_tmpb_tl
528       \tl_put_right:Nx \l__spath_tmpb_tl
529       {
530         {\dim_use:N \l__spath_tmpa_dim}
531         {\dim_use:N \l__spath_tmpb_dim}
532       }
533       \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
534     }
535   }
536   \tl_map_inline:Nn \g__spath_path_attributes
537   {
538     \spath_if_in:nnT {#1} {##1}
539     {
540       \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
541       \tl_clear:N \l__spath_tmpb_tl
542       \bool_until_do:nn {
543         \tl_if_empty_p:N \l__spath_tmpa_tl

```

```

544     }
545     {
546     \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
547     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
548
549     \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl + #2}
550     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
551
552     \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl + #3}
553     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
554
555     \tl_put_right:Nx \l__spath_tmpb_tl
556     {
557     {\dim_use:N \l__spath_tmpa_dim}
558     {\dim_use:N \l__spath_tmpb_dim}
559     }
560     }
561     \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
562   }
563 }
564 }
565
566 \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.

```

567 \cs_new_nopar:Npn \spath_translate:nn #1#2
568 {
569   \spath_translate:nnn {#1} #2
570 }
571
572 \cs_generate_variant:Nn \spath_translate:nn {nV}

```

`\spath_scale:nnn` Scale a path.

```

573 \cs_new_nopar:Npn \spath_scale:nnn #1#2#3
574 {
575   \tl_map_inline:Nn \g__spath_moveable_attributes
576   {
577     \spath_if_in:nnT {#1} {##1}
578     {
579       \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
580
581       \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl * #2}
582       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
583       \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl * #3}
584       \tl_clear:N \l__spath_tmpb_tl
585       \tl_put_right:Nx \l__spath_tmpb_tl
586       {
587         {\dim_use:N \l__spath_tmpa_dim}
588         {\dim_use:N \l__spath_tmpb_dim}

```

```

589     }
590     \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
591   }
592 }
593 \tl_map_inline:Nn \g__spath_path_attributes
594 {
595   \spath_if_in:nnT {#1} {##1}
596   {
597     \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
598     \tl_clear:N \l__spath_tmpb_tl
599     \bool_until_do:nn {
600       \tl_if_empty_p:N \l__spath_tmpa_tl
601     }
602     {
603       \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
604       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
605
606       \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl * #2}
607       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
608
609       \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl * #3}
610       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
611
612       \tl_put_right:Nx \l__spath_tmpb_tl
613       {
614         {\dim_use:N \l__spath_tmpa_dim}
615         {\dim_use:N \l__spath_tmpb_dim}
616       }
617     }
618     \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
619   }
620 }
621 }

```

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

```

622 \cs_new_nopar:Npn \spath_reverse:n #1
623 {
624   \spath_if_in:nnF {#1} {reverse path} {
625     \cs_use:c {spath_generate_reverse path}
626   }
627   \spath_swap:nnn {#1} {path} {reverse path}
628   \spath_swap:nnn {#1} {initial point} {final point}
629   \spath_swap:nnn {#1} {initial action} {final action}
630 }

```

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

```

631 \cs_new_nopar:Npn \spath_swap:nnn #1#2#3
632 {

```



```

633 \__spath_get:nnNF {#1} {#2} \l__spath_tmpa_tl {\tl_clear:N \l__spath_tmpa_tl}
634 \__spath_get:nnNF {#1} {#3} \l__spath_tmpb_tl {\tl_clear:N \l__spath_tmpb_tl}
635 \tl_if_empty:NTF \l__spath_tmpb_tl
636 {\spath_remove:nnV {#1} {#2}}
637 {\spath_put:nnV {#1} {#2} \l__spath_tmpb_tl}
638 \tl_if_empty:NTF \l__spath_tmpa_tl
639 {\spath_remove:nnV {#1} {#3}}
640 {\spath_put:nnV {#1} {#3} \l__spath_tmpa_tl}
641 }

```

`\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.

```

642 \cs_new_nopar:Npn \spath_weld:nn #1#2
643 {
644   \spath_clone:nn {#2} {tmp_path}
645   \spath_get:nnN {#1} {final point} \l__spath_tmpa_tl
646
647   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
648   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
649   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
650
651   \spath_get:nnN {#2} {initial point} \l__spath_tmpa_tl
652
653   \dim_sub:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
654   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
655   \dim_sub:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
656
657   \spath_translate:nxx {tmp_path} {\dim_use:N \l__spath_tmpa_dim} {\dim_use:N \l__spath_tmpb_dim}
658
659   \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
660   \__spath_get:nnN {tmp_path} {path} \l__spath_tmpb_tl
661   \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
662   \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
663   \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
664   \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
665
666   \spath_put:nnV {#1} {path} \l__spath_tmpa_tl
667
668   \__spath_get:nnNTF {tmp_path} {final point} \l__spath_tmpa_tl
669   {
670     \spath_put:nnV {#1} {final point} \l__spath_tmpa_tl
671   }
672   {
673     \spath_remove:nn {#1} {final point}
674   }
675
676   \__spath_get:nnNTF {tmp_path} {final action} \l__spath_tmpa_tl
677   {

```

```

678 \spath_put:nnV {#1} {final action} \l__spath_tmpa_tl
679 }
680 {
681 \spath_remove:nn {#1} {final action}
682 }
683
684 \__spath_get:nnNT {tmp_path} {min bb} \l__spath_tmpa_tl
685 {
686 \__spath_get:nnNT {#1} {min bb} \l__spath_tmpb_tl
687 {
688 \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
689 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
690 \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
691
692 \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\l__spath_tmpa_dim} {\tl_head:N
693 \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
694 \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\l__spath_tmpb_dim} {\tl_head:N \l__spath_t
695
696 \tl_clear:N \l__spath_tmpb_tl
697 \tl_put_right:Nx \l__spath_tmpb_tl
698 {
699 {\dim_use:N \l__spath_tmpa_dim}
700 {\dim_use:N \l__spath_tmpb_dim}
701 }
702 \spath_put:nnV {#1} {min bb} \l__spath_tmpb_tl
703 }
704 }
705
706 \__spath_get:nnNT {tmp_path} {max bb} \l__spath_tmpa_tl
707 {
708 \__spath_get:nnNT {#1} {max bb} \l__spath_tmpb_tl
709 {
710 \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
711 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
712 \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
713
714 \dim_set:Nn \l__spath_tmpa_dim {\dim_max:nn {\l__spath_tmpa_dim} {\tl_head:N
715 \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
716 \dim_set:Nn \l__spath_tmpb_dim {\dim_max:nn {\l__spath_tmpb_dim} {\tl_head:N \l__spath_t
717
718 \tl_clear:N \l__spath_tmpb_tl
719 \tl_put_right:Nx \l__spath_tmpb_tl
720 {
721 {\dim_use:N \l__spath_tmpa_dim}
722 {\dim_use:N \l__spath_tmpb_dim}
723 }
724 \spath_put:nnV {#1} {max bb} \l__spath_tmpb_tl
725 }
726 }
727

```

```

728 \__spath_get:nnNT {tmp_path} {reverse path} \l__spath_tmpa_tl
729 {
730   \__spath_get:nnNT {#1} {reverse path} \l__spath_tmpb_tl
731   {
732     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
733     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
734     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
735     \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
736
737     \spath_put:nnV {#1} {reverse path} \l__spath_tmpa_tl
738   }
739 }
740
741 \__spath_get:nnNT {tmp_path} {length} \l__spath_tmpa_tl
742 {
743   \__spath_get:nnNT {#1} {length} \l__spath_tmpb_tl
744   {
745     \int_set:Nn \l__spath_tmpa_int {\l__spath_tmpa_tl + \l__spath_tmpb_tl - 1}
746     \spath_put:nnV {#1} {length} \l__spath_tmpa_int
747   }
748 }
749
750 \__spath_get:nnNT {tmp_path} {real length} \l__spath_tmpa_tl
751 {
752   \__spath_get:nnNT {#1} {real length} \l__spath_tmpb_tl
753   {
754     \int_set:Nn \l__spath_tmpa_int {\l__spath_tmpa_tl + \l__spath_tmpb_tl}
755     \spath_put:nnV {#1} {real length} \l__spath_tmpa_int
756   }
757 }
758
759 \__spath_get:nnNT {tmp_path} {number of components} \l__spath_tmpa_tl
760 {
761   \__spath_get:nnNT {#1} {number of components} \l__spath_tmpb_tl
762   {
763     \int_set:Nn \l__spath_tmpa_int {\l__spath_tmpa_tl + \l__spath_tmpb_tl - 1}
764     \spath_put:nnV {#1} {number of components} \l__spath_tmpa_int
765   }
766 }
767
768 }

```

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

```

769 \cs_new_nopar:Npn \spath_prepend_no_move:nn #1#2
770 {
771   \spath_if_exist:nT {#2}
772   {
773     \__spath_get:nnN {#2} {path} \l__spath_tmpa_tl
774     \__spath_get:nnN {#1} {path} \l__spath_tmpb_tl
775     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}

```

```

776 \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
777 \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
778 \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
779 \spath_put:nnV {#1} {path} \l__spath_tmpa_tl
780
781 \spath_if_in:nnTF {#2} {initial point}
782 {
783   \__spath_get:nnN {#2} {initial point} \l__spath_tmpa_tl
784   \spath_put:nnV {#1} {initial point} \l__spath_tmpa_tl
785 }
786 {
787   \spath_remove:nn {#1} {initial point}
788 }
789
790 \spath_if_in:nnTF {#2} {initial action}
791 {
792   \__spath_get:nnN {#2} {initial action} \l__spath_tmpa_tl
793   \spath_put:nnV {#1} {initial action} \l__spath_tmpa_tl
794 }
795 {
796   \spath_remove:nn {#1} {initial action}
797 }
798
799 \bool_if:nTF
800 {
801   \spath_if_in_p:nn {#1} {length}
802   &&
803   \spath_if_in_p:nn {#2} {length}
804 }
805 {
806   \__spath_get:nnN {#1} {length} \l__spath_tmpa_tl
807   \__spath_get:nnN {#2} {length} \l__spath_tmpb_tl
808   \spath_put:nnx {#1} {length} {\int_eval:n {\l__spath_tmpa_tl +
809     \l__spath_tmpb_tl - 1}}
810 }
811 {
812   \spath_remove:nn {#1} {length}
813 }
814 \bool_if:nTF
815 {
816   \spath_if_in_p:nn {#1} {real length}
817   &&
818   \spath_if_in_p:nn {#2} {real length}
819 }
820 {
821   \__spath_get:nnN {#1} {real length} \l__spath_tmpa_tl
822   \__spath_get:nnN {#2} {real length} \l__spath_tmpb_tl
823   \spath_put:nnx {#1} {real length} {\int_eval:n {\l__spath_tmpa_tl +
824     \l__spath_tmpb_tl }}
825 }

```

```

826 {
827   \spath_remove:nn {#1} {real length}
828 }
829 \bool_if:nTF
830 {
831   \spath_if_in_p:nn {#1} {number of components}
832   &&
833   \spath_if_in_p:nn {#2} {number of components}
834 }
835 {
836   \__spath_get:nnN {#1} {number of components} \l__spath_tmpa_tl
837   \__spath_get:nnN {#2} {number of components} \l__spath_tmpb_tl
838   \spath_put:nxx {#1} {number of components} {\int_eval:n {\l__spath_tmpa_tl +
839     \l__spath_tmpb_tl - 1}}
840 }
841 {
842   \spath_remove:nn {#1} {number of components}
843 }
844 \bool_if:nTF
845 {
846   \spath_if_in_p:nn {#1} {min bb}
847   &&
848   \spath_if_in_p:nn {#2} {min bb}
849 }
850 {
851   \__spath_get:nnN {#1} {min bb} \l__spath_tmpa_tl
852   \__spath_get:nnN {#2} {min bb} \l__spath_tmpb_tl
853   \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_item:Nn
854     \l__spath_tmpa_tl {1}} {\tl_item:Nn
855     \l__spath_tmpb_tl {1}}}
856   \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_item:Nn
857     \l__spath_tmpa_tl {2}} {\tl_item:Nn
858     \l__spath_tmpb_tl {2}}}
859   \spath_put:nxx {#1} {min bb} {
860     {\dim_use:N \l__spath_tmpa_dim}
861     {\dim_use:N \l__spath_tmpb_dim}
862   }
863 }
864 {
865   \spath_remove:nn {#1} {min bb}
866 }
867 \bool_if:nTF
868 {
869   \spath_if_in_p:nn {#1} {max bb}
870   &&
871   \spath_if_in_p:nn {#2} {max bb}
872 }
873 {
874   \__spath_get:nnN {#1} {max bb} \l__spath_tmpa_tl
875   \__spath_get:nnN {#2} {max bb} \l__spath_tmpb_tl

```

```

876 \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_item:Nn
877 \l__spath_tmpa_tl {1}} {\tl_item:Nn
878 \l__spath_tmpb_tl {1}}}}
879 \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_item:Nn
880 \l__spath_tmpa_tl {2}} {\tl_item:Nn
881 \l__spath_tmpb_tl {2}}}}
882 \spath_put:nxx {#1} {max bb} {
883   {\dim_use:N \l__spath_tmpa_dim}
884   {\dim_use:N \l__spath_tmpb_dim}
885 }
886 }
887 {
888   \spath_remove:nn {#1} {max bb}
889 }
890 \bool_if:nTF
891 {
892   \spath_if_in_p:nn {#1} {reverse path}
893   &&
894   \spath_if_in_p:nn {#2} {reverse path}
895 }
896 {
897   \__spath_get:nnN {#1} {reverse path} \l__spath_tmpa_tl
898   \__spath_get:nnN {#2} {reverse path} \l__spath_tmpb_tl
899   \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
900   \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
901   \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
902   \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
903   \spath_put:nnV {#1} {reverse path} \l__spath_tmpb_tl
904 }
905 {
906   \spath_remove:nn {#1} {reverse path}
907 }
908
909 }
910 }

```

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

```

911 \cs_new_nopar:Npn \spath_append_no_move:nn #1#2
912 {
913   \spath_if_exist:nT {#2}
914   {
915     \spath_get:nnN {#1} {path} \l__spath_tmpa_tl
916     \spath_get:nnN {#2} {path} \l__spath_tmpb_tl
917     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
918     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
919     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
920     \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
921     \spath_put:nnV {#1} {path} \l__spath_tmpa_tl
922     \spath_if_in:nnTF {#2} {final point}
923     {

```

```

924     \_spath_get:nnN {#2} {final point} \l__spath_tmpa_tl
925     \spath_put:nnV {#1} {final point} \l__spath_tmpa_tl
926   }
927   {
928     \spath_remove:nn {#1} {final point}
929   }
930   \spath_if_in:nnTF {#2} {final action}
931   {
932     \_spath_get:nnN {#2} {final action} \l__spath_tmpa_tl
933     \spath_put:nnV {#1} {final action} \l__spath_tmpa_tl
934   }
935   {
936     \spath_remove:nn {#1} {final action}
937   }
938   \bool_if:nTF
939   {
940     \spath_if_in_p:nn {#1} {length}
941     &&
942     \spath_if_in_p:nn {#2} {length}
943   }
944   {
945     \_spath_get:nnN {#1} {length} \l__spath_tmpa_tl
946     \_spath_get:nnN {#2} {length} \l__spath_tmpb_tl
947     \spath_put:nnx {#1} {length} {\int_eval:n {\l__spath_tmpa_tl +
948       \l__spath_tmpb_tl - 1}}
949   }
950   {
951     \spath_remove:nn {#1} {length}
952   }
953   \bool_if:nTF
954   {
955     \spath_if_in_p:nn {#1} {real length}
956     &&
957     \spath_if_in_p:nn {#2} {real length}
958   }
959   {
960     \_spath_get:nnN {#1} {real length} \l__spath_tmpa_tl
961     \_spath_get:nnN {#2} {real length} \l__spath_tmpb_tl
962     \spath_put:nnx {#1} {real length} {\int_eval:n {\l__spath_tmpa_tl +
963       \l__spath_tmpb_tl }}
964   }
965   {
966     \spath_remove:nn {#1} {real length}
967   }
968   \bool_if:nTF
969   {
970     \spath_if_in_p:nn {#1} {number of components}
971     &&
972     \spath_if_in_p:nn {#2} {number of components}
973   }

```

```

974 {
975   \_spath_get:nnN {#1} {number of components} \l__spath_tmpa_tl
976   \_spath_get:nnN {#2} {number of components} \l__spath_tmpb_tl
977   \spath_put:nx {#1} {number of components} {\int_eval:n {\l__spath_tmpa_tl +
978     \l__spath_tmpb_tl - 1}}
979 }
980 {
981   \spath_remove:nn {#1} {number of components}
982 }
983 \bool_if:nTF
984 {
985   \spath_if_in_p:nn {#1} {min bb}
986   &&
987   \spath_if_in_p:nn {#2} {min bb}
988 }
989 {
990   \_spath_get:nnN {#1} {min bb} \l__spath_tmpa_tl
991   \_spath_get:nnN {#2} {min bb} \l__spath_tmpb_tl
992   \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_item:Nn
993     \l__spath_tmpa_tl {1}} {\tl_item:Nn
994     \l__spath_tmpb_tl {1}}}
995   \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_item:Nn
996     \l__spath_tmpa_tl {2}} {\tl_item:Nn
997     \l__spath_tmpb_tl {2}}}
998   \spath_put:nx {#1} {min bb} {
999     {\dim_use:N \l__spath_tmpa_dim}
1000     {\dim_use:N \l__spath_tmpb_dim}
1001   }
1002 }
1003 {
1004   \spath_remove:nn {#1} {min bb}
1005 }
1006 \bool_if:nTF
1007 {
1008   \spath_if_in_p:nn {#1} {max bb}
1009   &&
1010   \spath_if_in_p:nn {#2} {max bb}
1011 }
1012 {
1013   \_spath_get:nnN {#1} {max bb} \l__spath_tmpa_tl
1014   \_spath_get:nnN {#2} {max bb} \l__spath_tmpb_tl
1015   \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_item:Nn
1016     \l__spath_tmpa_tl {1}} {\tl_item:Nn
1017     \l__spath_tmpb_tl {1}}}
1018   \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_item:Nn
1019     \l__spath_tmpa_tl {2}} {\tl_item:Nn
1020     \l__spath_tmpb_tl {2}}}
1021   \spath_put:nx {#1} {max bb} {
1022     {\dim_use:N \l__spath_tmpa_dim}
1023     {\dim_use:N \l__spath_tmpb_dim}

```



```

1024     }
1025   }
1026   {
1027     \spath_remove:nn {#1} {max bb}
1028   }
1029   \bool_if:nTF
1030   {
1031     \spath_if_in_p:nn {#1} {reverse path}
1032     &&
1033     \spath_if_in_p:nn {#2} {reverse path}
1034   }
1035   {
1036     \__spath_get:nnN {#2} {reverse path} \l__spath_tmpa_tl
1037     \__spath_get:nnN {#1} {reverse path} \l__spath_tmpb_tl
1038     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
1039     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
1040     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
1041     \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
1042     \spath_put:nnV {#1} {reverse path} \l__spath_tmpb_tl
1043   }
1044   {
1045     \spath_remove:nn {#1} {reverse path}
1046   }
1047 }
1048 }

```

## 2.5 Iteration Functions

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

```

1049 \cs_new_nopar:Npn \spath_map_component:Nn #1#2
1050 {
1051   \int_gincr:N \g__prg_map_int
1052   \cs_gset:cpn { __prg_map_ \int_use:N \g__prg_map_int :w } ##1 {#2}
1053   \tl_set:NV \l__spath_tmpa_tl #1
1054   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1055   \tl_put_right:NV \l__spath_tmpa_tl \g__spath_moveto_tl
1056   \tl_set_eq:NN \l__spath_tmpb_tl \g__spath_moveto_tl
1057   \bool_until_do:nn {
1058     \tl_if_empty_p:N \l__spath_tmpa_tl
1059   }
1060   {
1061     \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
1062     \tl_if_eq:NNT \l__spath_tmpc_tl \g__spath_moveto_tl
1063     {
1064       \exp_args:NnV \use:c { __prg_map_ \int_use:N \g__prg_map_int :w } \l__spath_tmpb_tl
1065     }
1066     \tl_clear:N \l__spath_tmpb_tl
1067     \tl_if_single:NNT \l__spath_tmpc_tl
1068     {

```

```

1069     \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpc_tl
1070   }
1071   {
1072     \tl_put_right:Nx \l__spath_tmpb_tl {\l__spath_tmpc_tl}
1073   }
1074   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1075 }
1076 }

```

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

```

1077 \cs_new_nopar:Npn \spath_map_segment_inline:Nn #1#2
1078 {
1079   \int_gincr:N \g__prg_map_int
1080   \cs_gset:cpn { __prg_map_ \int_use:N \g__prg_map_int :w } ##1 ##2 {#2}
1081   \spath_map_segment_function:Nc #1 { __prg_map_ \int_use:N \g__prg_map_int :w }
1082 }

```

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

```

1083 \cs_new_nopar:Npn \spath_map_segment_inline:nn #1#2
1084 {
1085   \int_gincr:N \g__prg_map_int
1086   \cs_gset:cpn { __prg_map_ \int_use:N \g__prg_map_int :w } ##1 ##2 {#2}
1087   \spath_get:nnN {#1} {path} \l__spath_tmpd_tl
1088   \spath_map_segment_function:Nc \l__spath_tmpd_tl { __prg_map_ \int_use:N \g__prg_map_int :w }
1089 }

```

`\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.

```

1090 \cs_new_nopar:Npn \spath_map_segment_function:nn #1#2
1091 {
1092   \spath_get:nnN {#1} {path} \l__spath_tmpd_tl
1093   \spath_map_segment_function:NN \l__spath_tmpd_tl #2
1094 }
1095 \cs_new_nopar:Npn \spath_map_segment_function:NN #1#2
1096 {
1097   \tl_set_eq:NN \l__spath_tmpa_tl #1
1098   \tl_clear:N \l__spath_tmpb_tl
1099   \dim_zero:N \l__spath_tmpa_dim
1100   \dim_zero:N \l__spath_tmpb_dim
1101
1102   \bool_until_do:nn {
1103     \tl_if_empty_p:N \l__spath_tmpa_tl
1104   }
1105   {
1106     \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
1107     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1108     \tl_case:Nnn \l__spath_tmpc_tl

```

```

1109 {
1110   \g__spath_lineto_tl
1111   {
1112     \tl_set_eq:NN \l__spath_tmpb_tl \g__spath_moveto_tl
1113     \tl_put_right:Nx \l__spath_tmpb_tl
1114     {
1115       {\dim_use:N \l__spath_tmpa_dim}
1116       {\dim_use:N \l__spath_tmpb_dim}
1117     }
1118     \tl_put_right:NV \l__spath_tmpb_tl \g__spath_lineto_tl
1119
1120     \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1121     \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
1122     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1123
1124     \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1125     \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
1126     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1127
1128   }
1129
1130   \g__spath_curvetoa_tl
1131   {
1132     \tl_set_eq:NN \l__spath_tmpb_tl \g__spath_moveto_tl
1133     \tl_put_right:Nx \l__spath_tmpb_tl
1134     {
1135       {\dim_use:N \l__spath_tmpa_dim}
1136       {\dim_use:N \l__spath_tmpb_dim}
1137     }
1138     \tl_put_right:NV \l__spath_tmpb_tl \g__spath_curvetoa_tl
1139
1140     \prg_replicate:nn {2} {
1141       \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1142       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1143       \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1144       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1145       \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
1146       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1147     }
1148
1149     \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1150     \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
1151     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1152
1153     \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
1154     \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
1155     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1156
1157   }
1158 }

```

```

1159 {
1160
1161     \tl_set_eq:NN \l__spath_tmpb_tl \l__spath_tmpc_tl
1162     \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
1163     \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
1164     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1165
1166     \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
1167     \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
1168     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1169
1170 }
1171
1172 #2 \l__spath_tmpc_tl \l__spath_tmpb_tl
1173 \tl_clear:N \l__spath_tmpb_tl
1174
1175 }
1176 }
1177 \cs_generate_variant:Nn \spath_map_segment_function:NN {Nc}

```

## 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.

```

1178 \NewDocumentCommand \MakeSPath { m m }
1179 {
1180     \spath_clear_new:n {#1}
1181     \spath_put:nno {#1} {path} {#2}
1182 }

```

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

```

1183 \NewDocumentCommand \MakeSPathList { m m }
1184 {
1185     \tl_gclear_new:c {l__spath_list_#1}
1186     \int_zero:N \l__spath_tmpa_int
1187     \spath_map_component:Nn #2 {
1188         \spath_clear_new:n {#1 _ \int_use:N \l__spath_tmpa_int}
1189         \spath_put:nnn {#1 _ \int_use:N \l__spath_tmpa_int} {path} {##1}
1190         \tl_gput_right:cx {l__spath_list_#1} {\int_use:N \l__spath_tmpa_int}
1191         \int_incr:N \l__spath_tmpa_int
1192     }
1193 }

```

**\CloneSPath**

```

1194 \NewDocumentCommand \CloneSPath { m m }
1195 {

```

```

1196   \spath_clone:nn {#1} {#2}
1197 }

\SPathInfo
1198 \NewDocumentCommand \SPathInfo { m m }
1199 {
1200   \spath_get:nn {#1} {#2}
1201 }

\SPathPrepare
1202 \NewDocumentCommand \SPathPrepare { m }
1203 {
1204   \spath_generate_all:n {#1}
1205 }

\SPathListPrepare
1206 \NewDocumentCommand \SPathListPrepare { m }
1207 {
1208   \tl_map_inline:cn {l__spath_list_#1}
1209   {
1210     \spath_generate_all:n {##1}
1211   }
1212 }

\SPathInfoInto
1213 \NewDocumentCommand \SPathInfoInto { m m m }
1214 {
1215   \tl_set:Nn \l_tmpa_tl #3
1216   \spath_get:nnV {#1} {#2} \l_tmpa_tl
1217 }

\SPathShow
1218 \NewDocumentCommand \SPathShow { m m }
1219 {
1220   \spath_show:n {#1}
1221 }

\SPathTranslate
1222 \NewDocumentCommand \SPathTranslate { m m m }
1223 {
1224   \spath_translate:nnn {#1} {#2} {#3}
1225 }

\SPathTranslateInto Clones the path before translating it.
1226 \NewDocumentCommand \SPathTranslateInto { m m m m }
1227 {
1228   \spath_clone:nn {#1} {#2}
1229   \spath_translate:nnn {#2} {#3} {#4}
1230 }

```

`\SPathScale`

```
1231 \NewDocumentCommand \SPathScale { m m m }
1232 {
1233   \spath_translate:nnn {#1} {#2} {#3}
1234 }
```

`\SPathScaleInto` Clones the path first.

```
1235 \NewDocumentCommand \SPathScaleInto { m m m m }
1236 {
1237   \spath_clone:nn {#1} {#2}
1238   \spath_translate:nnn {#2} {#3} {#4}
1239 }
```

`\SPathWeld`

```
1240 \NewDocumentCommand \SPathWeld { m m }
1241 {
1242   \spath_weld:nn {#1} {#2}
1243 }
```

`\SPathWeldInto`

```
1244 \NewDocumentCommand \SPathWeldInto { m m m }
1245 {
1246   \spath_clone:nn {#1} {#2}
1247   \spath_weld:nn {#2} {#3}
1248 }
```

Interfaces via TikZ keys.

```
1249 \tikzset{
1250   save~spath/.code={
1251     \tikz@addmode{
1252       \spath_get_current_path:n {#1}
1253     }
1254   },
1255   restore~spath/.code={
1256     \spath_set_current_path:n {#1}
1257   },
1258 }
```

## 2.7 Miscellaneous Commands

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

```
1259 \cs_new_nopar:Npn \spath_split_curve:nnNN #1#2#3#4
1260 {
1261   \group_begin:
1262   \tl_gclear:N \l__spath_smuggle_tl
1263   \tl_set_eq:NN \l__spath_tmpa_tl \g__spath_moveto_tl
1264   \tl_put_right:Nx \l__spath_tmpa_tl {
1265     {\tl_item:nn {#2} {3}}
1266     {\tl_item:nn {#2} {4}}

```

```

1267 }
1268 \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curvetoa_tl
1269 \tl_put_right:Nx \l__spath_tmpa_tl
1270 {
1271   {\fp_to_dim:n
1272     {
1273       (1 - #1) * \tl_item:nn {#2} {2} + (#1) * \tl_item:nn {#2} {5}
1274     }}
1275   {\fp_to_dim:n
1276     {
1277       (1 - #1) * \tl_item:nn {#2} {3} + (#1) * \tl_item:nn {#2} {6}
1278     }}
1279 }
1280 \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curvetob_tl
1281 \tl_put_right:Nx \l__spath_tmpa_tl
1282 {
1283   {\fp_to_dim:n
1284     {
1285       (1 - #1)^2 * \tl_item:nn {#2} {2} + 2 * (1 - #1) * (#1) * \tl_item:nn {#2} {5} + (#1)^2 * \tl_item:nn {#2} {6}
1286     }}
1287   {\fp_to_dim:n
1288     {
1289       (1 - #1)^2 * \tl_item:nn {#2} {3} + 2 * (1 - #1) * (#1) * \tl_item:nn {#2} {6} + (#1)^2 * \tl_item:nn {#2} {7}
1290     }}
1291 }
1292 \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curveto_tl
1293 \tl_put_right:Nx \l__spath_tmpa_tl
1294 {
1295   {\fp_to_dim:n
1296     {
1297       (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}
1298     }}
1299   {\fp_to_dim:n
1300     {
1301       (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}
1302     }}
1303 }
1304 \tl_gset_eq:NN \l__spath_smuggle_tl \l__spath_tmpa_tl
1305 \group_end:
1306 \tl_set_eq:NN #3 \l__spath_smuggle_tl
1307 \group_begin:
1308 \tl_set_eq:NN \l__spath_tmpa_tl \g__spath_moveto_tl
1309 \tl_put_right:Nx \l__spath_tmpa_tl
1310 {
1311   {\fp_to_dim:n
1312     {
1313       (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}
1314     }}
1315   {\fp_to_dim:n
1316     {

```

```

1317     (1 - #1)^3 * \tl_item:nn {#2} {3} + 3 * (1 - #1)^2 * (#1) * \tl_item:nn {#2} {6} + 3 *
1318   }}
1319 }
1320 \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curvetoa_tl
1321 \tl_put_right:Nx \l__spath_tmpa_tl
1322 {
1323   {\fp_to_dim:n
1324   {
1325     (1 - #1)^2 * \tl_item:nn {#2} {5} + 2 * (1 - #1) * (#1) * \tl_item:nn {#2} {8} + (#1)^2
1326   }}
1327   {\fp_to_dim:n
1328   {
1329     (1 - #1)^2 * \tl_item:nn {#2} {6} + 2 * (1 - #1) * (#1) * \tl_item:nn {#2} {9} + (#1)^2
1330   }}
1331 }
1332 \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curvetob_tl
1333 \tl_put_right:Nx \l__spath_tmpa_tl
1334 {
1335   {\fp_to_dim:n
1336   {
1337     (1 - #1) * \tl_item:nn {#2} {8} + (#1) * \tl_item:nn {#2} {11}
1338   }}
1339   {\fp_to_dim:n
1340   {
1341     (1 - #1) * \tl_item:nn {#2} {9} + (#1) * \tl_item:nn {#2} {12}
1342   }}
1343 }
1344 \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curveto_tl
1345 \tl_put_right:Nx \l__spath_tmpa_tl {
1346   {\tl_item:nn {#2} {11}}
1347   {\tl_item:nn {#2} {12}}
1348 }
1349 \tl_gset_eq:NN \l__spath_smuggle_tl \l__spath_tmpa_tl
1350 \group_end:
1351 \tl_set_eq:NN #4 \l__spath_smuggle_tl
1352 }
1353
1354 \cs_generate_variant:Nn \spath_split_curve:nnNN {nVNN, VVNN}

```

## 3 The Calligraphy Package

### 3.1 Initialisation

```

1355 \RequirePackage{spath3}
1356 \ExplSyntaxOn
1357
1358 \tl_new:N \l__cal_tmpa_tl
1359 \tl_new:N \l__cal_tmpb_tl
1360 \int_new:N \l__cal_tmpa_int

```



```

1361 \int_new:N \l__cal_tmpb_int
1362 \int_new:N \l__cal_path_component_int
1363 \dim_new:N \l__cal_tmpa_dim
1364 \dim_new:N \l__cal_tmpb_dim
1365 \dim_new:N \l__cal_tmpe_dim
1366 \dim_new:N \l__cal_tmpe_dim
1367 \dim_new:N \l__cal_tmpe_dim
1368 \dim_new:N \l__cal_tmpe_dim
1369 \dim_new:N \l__cal_tmpe_dim
1370 \dim_new:N \l__cal_tmpe_dim
1371 \bool_new:N \l__cal_taper_start_bool
1372 \bool_new:N \l__cal_taper_end_bool
1373 \bool_new:N \l__cal_taperable_bool
1374 \dim_new:N \l__cal_taper_width_dim
1375 \dim_new:N \l__cal_line_width_dim
1376
1377 \bool_set_true:N \l__cal_taper_start_bool
1378 \bool_set_true:N \l__cal_taper_end_bool

```

## 3.2 TikZ Keys

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

```

1379 \tikzset{
1380   define~pen/.code={
1381     \tikzset{pen~name=#1}
1382     \pgf@relevantforpicturesizefalse
1383     \tikz@addmode{
1384       \pgfsyssoftpath@getcurrentpath\l__cal_tmpa_tl
1385       \MakeSPathList{calligraphy pen \pgfkeysvalueof{/tikz/pen~name}}{\l__cal_tmpa_tl}
1386       \SPathListPrepare{calligraphy pen \pgfkeysvalueof{/tikz/pen~name}}
1387       \pgfusepath{discard}%
1388     }
1389   },
1390   define~pen/.default={default},
1391   use~pen/.code={
1392     \tikzset{pen~name=#1}
1393     \int_gzero:N \l__cal_path_component_int
1394     \cs_set_eq:NN \pgfpathmoveto \cal_moveto:n
1395     \tikz@addmode{
1396       \pgfsyssoftpath@getcurrentpath\l__cal_tmpa_tl
1397       \MakeSPathList{calligraphy path}{\l__cal_tmpa_tl}
1398       \SPathListPrepare{calligraphy path}
1399       \CalligraphyPathCreate{calligraphy path}{\pgfkeysvalueof{/tikz/pen~name}}
1400     }
1401   },
1402   use~pen/.default={default},
1403   pen~name/.initial={default},
1404   copperplate/.style={pen~name=copperplate},
1405   pen~colour/.initial={black},
1406   weight/.is~choice,

```

```

1407 weight/heavy/.style={
1408   line~width=\pgfkeysvalueof{/tikz/heavy~line~width},
1409   taper~width=\pgfkeysvalueof{/tikz/light~line~width},
1410 },
1411 weight/light/.style={
1412   line~width=\pgfkeysvalueof{/tikz/light~line~width},
1413   taper~width=0pt,
1414 },
1415 heavy/.style={
1416   weight=heavy
1417 },
1418 light/.style={
1419   weight=light
1420 },
1421 heavy~line~width/.initial=2pt,
1422 light~line~width/.initial=1pt,
1423 taper/.is~choice,
1424 taper/.default=both,
1425 taper/none/.style={
1426   taper~start=false,
1427   taper~end=false,
1428 },
1429 taper/both/.style={
1430   taper~start=true,
1431   taper~end=true,
1432 },
1433 taper/start/.style={
1434   taper~start=true,
1435   taper~end=false,
1436 },
1437 taper/end/.style={
1438   taper~start=false,
1439   taper~end=true,
1440 },
1441 taper~start/.code={
1442   \tl_if_eq:nnTF {#1} {true}
1443   {
1444     \bool_set_true:N \l__cal_taper_start_bool
1445   }
1446   {
1447     \bool_set_false:N \l__cal_taper_start_bool
1448   }
1449 },
1450 taper~start/.default={true},
1451 taper~end/.code={
1452   \tl_if_eq:nnTF {#1} {true}
1453   {
1454     \bool_set_true:N \l__cal_taper_end_bool
1455   }
1456   {

```

```

1457     \bool_set_false:N \l__cal_taper_end_bool
1458   }
1459 },
1460 taper~end/.default={true},
1461 taper~width/.code={\dim_set:Nn \l__cal_taper_width_dim {#1}},
1462 nib~style/.code~2~args={
1463   \tl_clear_new:c {l__cal_nib_style_#1}
1464   \tl_set:cn {l__cal_nib_style_#1} {#2}
1465 },
1466 stroke~style/.code~2~args={
1467   \tl_clear_new:c {l__cal_stroke_style_#1}
1468   \tl_set:cn {l__cal_stroke_style_#1} {#2}
1469 },
1470 this~stroke~style/.code={
1471   \tl_clear_new:c {l__cal_stroke_inline_style_ \int_use:N \l__cal_path_component_int}
1472   \tl_set:cn {l__cal_stroke_inline_style_ \int_use:N \l__cal_path_component_int} {#1}
1473 },
1474 }

```

Some wrappers around the TikZ keys.

```

1475 \NewDocumentCommand \pen { 0{ } }
1476 {
1477   \path[define~ pen,#1]
1478 }
1479
1480 \NewDocumentCommand \definepen { 0{ } }
1481 {
1482   \tikz \path[define~ pen,#1]
1483 }
1484
1485 \NewDocumentCommand \calligraphy { 0{ } }
1486 {
1487   \path[use~ pen,#1]
1488 }

```

### 3.3 The Path Creation

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

```

1489 \NewDocumentCommand \CalligraphyPathCreate { m m }
1490 {
1491   \int_zero:N \l__cal_tmpa_int
1492   \tl_map_inline:cn {l__spath_list_#1}
1493   {
1494     \int_incr:N \l__cal_tmpa_int
1495     \int_zero:N \l__cal_tmpb_int
1496     \tl_map_inline:cn {l__spath_list_calligraphy pen #2}
1497     {
1498       \int_incr:N \l__cal_tmpb_int
1499       \group_begin:
1500       \cal_apply_style:c {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_int}

```

```

1501 \cal_apply_style:c {l__cal_stroke_inline_style_ \int_use:N \l__cal_tmpa_int}
1502 \cal_apply_style:c {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}
1503
1504 \spath_clone:nn {##1} {calligraphy temp path}
1505
1506 \__spath_get:nnN {####1} {initial point} \l__spath_tmpa_tl
1507 \spath_translate:nV {calligraphy temp path} \l__spath_tmpa_tl
1508
1509 \__spath_get:nnN {####1} {length} \l__spath_tmpa_tl
1510
1511 \int_compare:nTF {\l__spath_tmpa_tl = 1}
1512 {
1513   \cal_at_least_three:n {calligraphy temp path}
1514
1515   \spath_protocol_path:n {calligraphy temp path}
1516
1517   \__spath_get:nnN {calligraphy temp path} {path} \l__spath_tmpa_tl
1518
1519   \tikz@options
1520   \dim_set:Nn \l__cal_line_width_dim {\pgflinewidth}
1521   \cal_maybe_taper:N \l__spath_tmpa_tl
1522 }
1523 {
1524
1525   \spath_weld:nn {calligraphy temp path} {####1}
1526   \spath_reverse:n {##1}
1527   \spath_reverse:n {####1}
1528   \spath_weld:nn {calligraphy temp path} {##1}
1529   \spath_weld:nn {calligraphy temp path} {####1}
1530   \spath_reverse:n {##1}
1531   \spath_reverse:n {####1}
1532
1533   \spath_set_current_path:n {calligraphy temp path}
1534
1535   \tikz@options
1536   \pgfsetfillcolor{\pgfkeysvalueof{/tikz/pen-colour}}
1537   \pgfusepath{fill}
1538 }
1539 \group_end:
1540 }
1541 }
1542 }

```

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

```

1543 \cs_new_eq:NN \cal_orig_moveto:n \pgfpathmoveto
1544 \cs_new_nopar:Npn \cal_moveto:n #1
1545 {
1546   \int_gincr:N \l__cal_path_component_int
1547   \cal_orig_moveto:n {#1}

```

```
1548 }
```

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

```
1549 \cs_new_nopar:Npn \cal_apply_style:N #1
1550 {
1551   \tl_if_exist:NT #1 {
1552     \exp_args:NV \tikzset #1
1553   }
1554 }
1555 \cs_generate_variant:Nn \cal_apply_style:N {c}
```

`\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.

```
1556 \cs_new_nopar:Npn \cal_at_least_three:n #1
1557 {
1558   \spath_get:nnN {#1} {real length} \l__cal_tmpa_tl
1559   \tl_clear:N \l__cal_tmpb_tl
1560   \int_compare:nTF {\l__cal_tmpa_tl = 1}
1561   {
1562     \spath_get:nnN {#1} {path} \l__cal_tmpa_tl
1563     \spath_map_segment_inline:Nn \l__cal_tmpa_tl
1564     {
1565       \tl_case:Nnn ##1 {
1566         \g__spath_lineto_tl {
1567           \cal_split_line_in_three:NN \l__cal_tmpb_tl ##2
1568         }
1569         \g__spath_curvetoa_tl {
1570           \cal_split_curve_in_three:NN \l__cal_tmpb_tl ##2
1571         }
1572       }
1573     }
1574     \tl_put_right:NV \l__cal_tmpb_tl ##2
1575   }
1576 }
1577 \spath_put:nnV {#1} {path} \l__cal_tmpb_tl
1578 }
1579 {
1580   \int_compare:nT {\l__cal_tmpa_tl = 2}
1581   {
1582     \spath_get:nnN {#1} {path} \l__cal_tmpa_tl
1583     \spath_map_segment_inline:Nn \l__cal_tmpa_tl
1584     {
1585       \tl_case:Nnn ##1 {
1586         \g__spath_lineto_tl {
1587           \cal_split_line_in_two:NN \l__cal_tmpb_tl ##2
1588         }
1589         \g__spath_curvetoa_tl {
1590           \cal_split_curve_in_two:NN \l__cal_tmpb_tl ##2
1591         }

```

```

1592     }
1593     {
1594     \tl_put_right:NV \l__cal_tmpb_tl ##2
1595     }
1596   }
1597   \spath_put:nnV {#1} {path} \l__cal_tmpb_tl
1598 }
1599 }
1600 }

```

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

```

1601 \cs_new_nopar:Npn \cal_split_line_in_two:NN #1#2
1602 {
1603   \tl_set_eq:NN \l__cal_tmpc_tl #2
1604
1605   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1606
1607   \dim_set:Nn \l__cal_tmpa_dim {\tl_head:N \l__cal_tmpc_tl}
1608   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1609
1610   \dim_set:Nn \l__cal_tmpb_dim {\tl_head:N \l__cal_tmpc_tl}
1611   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1612
1613   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1614
1615   \dim_set:Nn \l__cal_tmpc_dim {\tl_head:N \l__cal_tmpc_tl}
1616   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1617   \dim_set:Nn \l__cal_tmpd_dim {\tl_head:N \l__cal_tmpc_tl}
1618   \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1619
1620   \tl_put_right:NV #1 \g__spath_lineto_tl
1621
1622   \tl_put_right:Nx #1 {
1623     {\dim_eval:n {(\l__cal_tmpa_dim + \l__cal_tmpc_dim)/2}}
1624     {\dim_eval:n {(\l__cal_tmpb_dim + \l__cal_tmpd_dim)/2}}
1625   }
1626
1627   \tl_put_right:NV #1 \g__spath_lineto_tl
1628   \tl_put_right:Nx #1 {
1629     {\dim_use:N \l__cal_tmpc_dim}
1630     {\dim_use:N \l__cal_tmpd_dim}
1631   }
1632 }

```

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

```

1633 \cs_new_nopar:Npn \cal_split_line_in_three:NN #1#2
1634 {
1635   \tl_set_eq:NN \l__cal_tmpc_tl #2
1636

```

```

1637 \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1638
1639 \dim_set:Nn \l__cal_tmpa_dim {\tl_head:N \l__cal_tmpc_tl}
1640 \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1641
1642 \dim_set:Nn \l__cal_tmpb_dim {\tl_head:N \l__cal_tmpc_tl}
1643 \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1644
1645 \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1646
1647 \dim_set:Nn \l__cal_tmpc_dim {\tl_head:N \l__cal_tmpc_tl}
1648 \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1649 \dim_set:Nn \l__cal_tmpd_dim {\tl_head:N \l__cal_tmpc_tl}
1650 \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
1651
1652 \tl_put_right:NV #1 \g__spath_lineto_tl
1653
1654 \tl_put_right:Nx #1 {
1655   {\dim_eval:n {(2\l__cal_tmpa_dim + \l__cal_tmpc_dim)/3}}
1656   {\dim_eval:n {(2\l__cal_tmpb_dim + \l__cal_tmpd_dim)/3}}
1657 }
1658
1659 \tl_put_right:NV #1 \g__spath_lineto_tl
1660
1661 \tl_put_right:Nx #1 {
1662   {\dim_eval:n {(\l__cal_tmpa_dim + 2\l__cal_tmpc_dim)/3}}
1663   {\dim_eval:n {(\l__cal_tmpb_dim + 2\l__cal_tmpd_dim)/3}}
1664 }
1665
1666 \tl_put_right:NV #1 \g__spath_lineto_tl
1667 \tl_put_right:Nx #1 {
1668   {\dim_use:N \l__cal_tmpc_dim}
1669   {\dim_use:N \l__cal_tmpd_dim}
1670 }
1671 }

```

\cal\_split\_curve\_in\_two:NN Splits a curve in two, adding the splits to the first token list.

```

1672 \cs_new_nopar:Npn \cal_split_curve_in_two:NN #1#2
1673 {
1674   \spath_split_curve:nVNN {.5} #2 \l_tmpa_tl \l_tmpb_tl
1675   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1676   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1677   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1678   \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
1679   \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
1680   \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
1681   \tl_put_right:NV #1 \l_tmpa_tl
1682   \tl_put_right:NV #1 \l_tmpb_tl
1683 }

```

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

```
1684 \cs_new_nopar:Npn \cal_split_curve_in_three:NN #1#2
1685 {
1686   \spath_split_curve:nVNN {1/3} #2 \l_tmpa_tl \l_tmpb_tl
1687
1688   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1689   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1690   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1691   \tl_put_right:NV #1 \l_tmpa_tl
1692
1693   \spath_split_curve:nVNN {.5} \l_tmpb_tl \l_tmpa_tl \l_tmpb_tl
1694   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1695   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1696   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
1697   \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
1698   \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
1699   \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
1700   \tl_put_right:NV #1 \l_tmpa_tl
1701   \tl_put_right:NV #1 \l_tmpb_tl
1702 }
```

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

```
1703 \cs_new_nopar:Npn \cal_maybe_taper:N #1
1704 {
1705   \tl_set_eq:NN \l__cal_tmpa_tl #1
1706
1707   \bool_if:NT \l__cal_taper_start_bool
1708   {
1709
1710     \dim_set:Nn \l__cal_tmpa_dim {\tl_item:Nn \l__cal_tmpa_tl {2}}
1711     \dim_set:Nn \l__cal_tmpb_dim {\tl_item:Nn \l__cal_tmpa_tl {3}}
1712     \tl_set:Nx \l__cal_tmpb_tl {\tl_item:Nn \l__cal_tmpa_tl {4}}
1713
1714     \tl_case:Nnn \l__cal_tmpb_tl
1715     {
1716       \g__spath_lineto_tl
1717       {
1718
1719         \bool_set_true:N \l__cal_taperable_bool
1720         \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {5}}
1721         \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {6}}
1722         \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpa_dim + \l__cal_tmpe_dim)/3}
1723         \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpe_dim + \l__cal_tmpe_dim)/3}
1724         \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmpe_dim + 2\l__cal_tmpe_dim)/3}
1725         \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmpe_dim + 2\l__cal_tmpe_dim)/3}
1726         \prg_replicate:nn {4}
1727         {
1728           \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
1729         }
1730       }
1731     }
1732 }
```



```

1730     \tl_put_left:NV \l__cal_tmpa_tl \g__spath_moveto_tl
1731   }
1732   \g__spath_curvetoa_tl
1733   {
1734     \bool_set_true:N \l__cal_taperable_bool
1735     \dim_set:Nn \l__cal_tmpc_dim {\tl_item:Nn \l__cal_tmpa_tl {5}}
1736     \dim_set:Nn \l__cal_tmpd_dim {\tl_item:Nn \l__cal_tmpa_tl {6}}
1737     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {8}}
1738     \dim_set:Nn \l__cal_tmpf_dim {\tl_item:Nn \l__cal_tmpa_tl {9}}
1739     \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {11}}
1740     \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {12}}
1741     \prg_replicate:nn {10}
1742     {
1743       \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
1744     }
1745     \tl_put_left:NV \l__cal_tmpa_tl \g__spath_moveto_tl
1746   }
1747 }
1748 {
1749   \bool_set_false:N \l__cal_taperable_bool
1750 }
1751
1752 \bool_if:NT \l__cal_taperable_bool
1753 {
1754   \__cal_taper_aux:
1755 }
1756
1757 }
1758
1759 \bool_if:NT \l__cal_taper_end_bool
1760 {
1761
1762   \dim_set:Nn \l__cal_tmpa_dim {\tl_item:Nn \l__cal_tmpa_tl {-2}}
1763   \dim_set:Nn \l__cal_tmpb_dim {\tl_item:Nn \l__cal_tmpa_tl {-1}}
1764   \tl_set:Nx \l__cal_tmpb_tl {\tl_item:Nn \l__cal_tmpa_tl {-3}}
1765
1766   \tl_case:Nnn \l__cal_tmpb_tl
1767   {
1768     \g__spath_lineto_tl
1769     {
1770
1771       \bool_set_true:N \l__cal_taperable_bool
1772       \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {-5}}
1773       \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {-4}}
1774       \dim_set:Nn \l__cal_tmpc_dim {(2\l__cal_tmpa_dim + \l__cal_tmpg_dim)/3}
1775       \dim_set:Nn \l__cal_tmpd_dim {(2\l__cal_tmpb_dim + \l__cal_tmph_dim)/3}
1776       \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmpa_dim + 2\l__cal_tmpg_dim)/3}
1777       \dim_set:Nn \l__cal_tmpf_dim {(\l__cal_tmpb_dim + 2\l__cal_tmph_dim)/3}
1778       \tl_reverse:N \l__cal_tmpa_tl
1779       \prg_replicate:nn {3}

```

```

1780     {
1781       \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
1782     }
1783     \tl_reverse:N \l__cal_tmpa_tl
1784   }
1785   \g__spath_curveto_tl
1786   {
1787     \bool_set_true:N \l__cal_taperable_bool
1788     \dim_set:Nn \l__cal_tmpc_dim {\tl_item:Nn \l__cal_tmpa_tl {-5}}
1789     \dim_set:Nn \l__cal_tmpd_dim {\tl_item:Nn \l__cal_tmpa_tl {-4}}
1790     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-8}}
1791     \dim_set:Nn \l__cal_tmpf_dim {\tl_item:Nn \l__cal_tmpa_tl {-7}}
1792     \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {-11}}
1793     \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {-10}}
1794     \tl_reverse:N \l__cal_tmpa_tl
1795     \prg_replicate:nn {9}
1796     {
1797       \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
1798     }
1799     \tl_reverse:N \l__cal_tmpa_tl
1800   }
1801 }
1802 {
1803   \bool_set_false:N \l__cal_taperable_bool
1804 }
1805
1806 \bool_if:NT \l__cal_taperable_bool
1807 {
1808   \__cal_taper_aux:
1809 }
1810
1811 }
1812
1813 \pgfsyssoftpath@setcurrentpath\l__cal_tmpa_tl
1814 \pgfsetstrokecolor{\pgfkeysvalueof{/tikz/pen~colour}}
1815 \pgfusepath{stroke}
1816
1817 }

```

`\__cal_taper_aux:` Auxiliary macro to avoid unnecessary code duplication.

```

1818 \cs_new_nopar:Npn \__cal_taper_aux:
1819 {
1820   \tl_clear:N \l__cal_tmpb_tl
1821   \tl_put_right:NV \l__cal_tmpb_tl \g__spath_moveto_tl
1822
1823   \fp_set:Nn \l__cal_tmpa_fp
1824   {
1825     \l__cal_tmpd_dim - \l__cal_tmpb_dim
1826   }
1827   \fp_set:Nn \l__cal_tmpb_fp

```

```

1828 {
1829   \l__cal_tmpe_dim - \l__cal_tmpe_dim
1830 }
1831 \fp_set:Nn \l__cal_tmpe_fp
1832 {
1833   (\l__cal_tmpe_dim^2 + \l__cal_tmpe_dim)^.5
1834 }
1835
1836 \fp_set:Nn \l__cal_tmpe_dim {.5*\l__cal_tmpe_dim * \l__cal_tmpe_dim / \l__cal_tmpe_dim}
1837 \fp_set:Nn \l__cal_tmpe_dim {.5*\l__cal_tmpe_dim * \l__cal_tmpe_dim / \l__cal_tmpe_dim}
1838
1839 \fp_set:Nn \l__cal_tmpe_dim
1840 {
1841   \l__cal_tmpe_dim - \l__cal_tmpe_dim
1842 }
1843 \fp_set:Nn \l__cal_tmpe_dim
1844 {
1845   \l__cal_tmpe_dim - \l__cal_tmpe_dim
1846 }
1847 \fp_set:Nn \l__cal_tmpe_dim
1848 {
1849   (\l__cal_tmpe_dim^2 + \l__cal_tmpe_dim)^.5
1850 }
1851
1852 \fp_set:Nn \l__cal_tmpe_dim {.5*\l__cal_tmpe_dim * \l__cal_tmpe_dim / \l__cal_tmpe_dim}
1853 \fp_set:Nn \l__cal_tmpe_dim {.5*\l__cal_tmpe_dim * \l__cal_tmpe_dim / \l__cal_tmpe_dim}
1854
1855 \tl_put_right:Nx \l__cal_tmpe_dim
1856 {
1857   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
1858   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
1859 }
1860
1861 \tl_put_right:NV \l__cal_tmpe_dim \g__spath_curvetoa_tl
1862
1863 \tl_put_right:Nx \l__cal_tmpe_dim
1864 {
1865   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
1866   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
1867 }
1868
1869 \tl_put_right:NV \l__cal_tmpe_dim \g__spath_curvetob_tl
1870
1871 \tl_put_right:Nx \l__cal_tmpe_dim
1872 {
1873   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
1874   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
1875 }
1876
1877 \tl_put_right:NV \l__cal_tmpe_dim \g__spath_curveto_tl

```

```

1878
1879 \tl_put_right:Nx \l__cal_tmpb_tl
1880 {
1881   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
1882   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
1883 }
1884
1885 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetoa_tl
1886
1887 \tl_put_right:Nx \l__cal_tmpb_tl
1888 {
1889   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim - \fp_to_dim:n{ 1.32 * \l__
1890   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim + \fp_to_dim:n {1.32* \l__
1891 }
1892
1893 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetob_tl
1894
1895 \tl_put_right:Nx \l__cal_tmpb_tl
1896 {
1897   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim - \fp_to_dim:n {1.32 * \l__
1898   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim + \fp_to_dim:n {1.32 * \l__
1899 }
1900
1901 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curveto_tl
1902
1903 \tl_put_right:Nx \l__cal_tmpb_tl
1904 {
1905   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
1906   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
1907 }
1908
1909 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetoa_tl
1910
1911 \tl_put_right:Nx \l__cal_tmpb_tl
1912 {
1913   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
1914   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
1915 }
1916
1917 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetob_tl
1918
1919 \tl_put_right:Nx \l__cal_tmpb_tl
1920 {
1921   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
1922   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
1923 }
1924
1925 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curveto_tl
1926
1927 \tl_put_right:Nx \l__cal_tmpb_tl

```

```

1928 {
1929   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim}}
1930   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim}}
1931 }
1932
1933 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetoa_tl
1934
1935 \tl_put_right:Nx \l__cal_tmpb_tl
1936 {
1937   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim + \fp_to_dim:n {1.32 * \l__
1938   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim - \fp_to_dim:n {1.32 * \l__
1939 }
1940
1941 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetob_tl
1942
1943 \tl_put_right:Nx \l__cal_tmpb_tl
1944 {
1945   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim + \fp_to_dim:n {1.32 * \l__
1946   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim - \fp_to_dim:n {1.32 * \l__
1947 }
1948
1949 \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curveto_tl
1950
1951 \tl_put_right:Nx \l__cal_tmpb_tl
1952 {
1953   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim}}
1954   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim}}
1955 }
1956
1957 \pgfsyssofpath@setcurrentpath\l__cal_tmpb_tl
1958 \pgfsetfillcolor{\pgfkeysvalueof{/tikz/pen~colour}}
1959 \pgfusepath{fill}
1960 }
    Defines a copperplate pen.
1961 \tl_set:Nn \l__cal_tmpa_tl {\pgfsyssofpath@movetotoken{0pt}{0pt}}
1962 \MakeSPathList{calligraphy pen copperplate}{\l__cal_tmpa_tl}
1963 \SPathListPrepare{calligraphy pen copperplate}
1964 \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.

```

1965 \expandafter\ifx\csname pgfdeclaredecoration\endcsname\relax
1966 \else
1967 \pgfdeclaredecoration{calligraphic brace}{brace}
1968 {

```

```

1969 \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]
1970 {
1971   \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}
1972   \pgfpathmoveto{\pgfpointorigin}
1973   \pgfpathcurveto
1974   {\pgfqpoint{.15\pgfdecorationsegmentamplitude}{.3\pgfdecorationsegmentamplitude}}
1975   {\pgfqpoint{.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
1976   {\pgfqpoint{\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
1977   {
1978     \pgftransformxshift{+\pgfdecorationsegmentaspect\pgfdecoratedremainingdistance}
1979     \pgfpathlineto{\pgfqpoint{-\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
1980     \pgfpathcurveto
1981     {\pgfqpoint{-.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
1982     {\pgfqpoint{-.15\pgfdecorationsegmentamplitude}{.7\pgfdecorationsegmentamplitude}}
1983     {\pgfqpoint{0\pgfdecorationsegmentamplitude}{1\pgfdecorationsegmentamplitude}}
1984     \pgfpathmoveto{\pgfqpoint{0\pgfdecorationsegmentamplitude}{1\pgfdecorationsegmentamplitude}}
1985     \pgfpathcurveto
1986     {\pgfqpoint{.15\pgfdecorationsegmentamplitude}{.7\pgfdecorationsegmentamplitude}}
1987     {\pgfqpoint{.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
1988     {\pgfqpoint{\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
1989   }
1990   {
1991     \pgftransformxshift{+\pgfdecoratedremainingdistance}
1992     \pgfpathlineto{\pgfqpoint{-\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
1993     \pgfpathcurveto
1994     {\pgfqpoint{-.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
1995     {\pgfqpoint{-.15\pgfdecorationsegmentamplitude}{.3\pgfdecorationsegmentamplitude}}
1996     {\pgfqpoint{0pt}{0pt}}
1997   }
1998   \tikzset{
1999     taper width=.5\pgflinewidth,
2000     taper
2001   }%
2002   \pgfsyssoftpath@getcurrentpath\cal@tmp@path
2003   \MakeSPathList{calligraphy path}{\cal@tmp@path}%
2004   \SPathListPrepare{calligraphy path}%
2005   \CalligraphyPathCreate{calligraphy path}{copperplate}%
2006 }
2007 \state{final}{}
2008 }

```

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

```

2009 \pgfdeclaredecoration{calligraphic straight parenthesis}{brace}
2010 {
2011   \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]
2012   {
2013     \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}
2014     \pgfpathmoveto{\pgfpointorigin}
2015     \pgfpathcurveto

```

```

2016 {\pgfqpoint{.76604\pgfdecorationsegmentamplitude}{.64279\pgfdecorationsegmentamplitude}}
2017 {\pgfqpoint{2.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
2018 {\pgfqpoint{3.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
2019 {
2020   \pgftransformxshift{+\pgfdecoratedremainingdistance}
2021   \pgfpathlineto{\pgfqpoint{-3.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
2022   \pgfpathcurveto
2023     {\pgfqpoint{-2.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
2024     {\pgfqpoint{-.76604\pgfdecorationsegmentamplitude}{.64279\pgfdecorationsegmentamplitude}}
2025     {\pgfqpoint{0pt}{0pt}}
2026 }
2027 \tikzset{
2028   taper width=.5\pgflinewidth,
2029   taper
2030 }%
2031 \pgfsyssoftpath@getcurrentpath\cal@tmp@path
2032 \MakeSPathList{calligraphy path}{\cal@tmp@path}%
2033 \SPathListPrepare{calligraphy path}%
2034 \CalligraphyPathCreate{calligraphy path}{copperplate}%
2035 }
2036 \state{final}{}%
2037 }

```

The third is a curved parenthesis.

```

2038 \pgfdeclaredecoration{calligraphic curved parenthesis}{brace}
2039 {
2040   \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]
2041   {
2042     \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}
2043     \pgfpathmoveto{\pgfpointorigin}
2044     \pgf@xa=\pgfdecoratedremainingdistance\relax
2045     \advance\pgf@xa by -1.5890\pgfdecorationsegmentamplitude\relax
2046     \edef\cgrphy@xa{\the\pgf@xa}
2047     \pgfpathcurveto
2048       {\pgfqpoint{1.5890\pgfdecorationsegmentamplitude}{1.3333\pgfdecorationsegmentamplitude}}
2049       {\pgfqpoint{\cgrphy@xa}{1.3333\pgfdecorationsegmentamplitude}}
2050       {\pgfqpoint{\pgfdecoratedremainingdistance}{0pt}}
2051     \tikzset{
2052       taper width=.5\pgflinewidth,
2053       taper
2054     }%
2055     \pgfsyssoftpath@getcurrentpath\cal@tmp@path
2056     \MakeSPathList{calligraphy path}{\cal@tmp@path}%
2057     \SPathListPrepare{calligraphy path}%
2058     \CalligraphyPathCreate{calligraphy path}{copperplate}%
2059   }
2060   \state{final}{}%
2061 }

```

End the conditional for if pgfdecoration module is loaded

```
2062 \fi
```

## 4 Drawing Knots

### 4.1 Initialisation

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

```
2063 \RequirePackage{spath3}
2064 \usetikzlibrary{intersections}
2065
2066 \ExplSyntaxOn
2067
2068 \tl_new:N \l__knot_tmpa_tl
2069 \tl_new:N \l__knot_tmpb_tl
2070 \tl_new:N \l__knot_tmpc_tl
2071 \tl_new:N \l__knot_tmpd_tl
2072 \tl_new:N \l__knot_tmpe_tl
2073 \tl_new:N \l__knot_tmpf_tl
2074 \tl_new:N \l__knot_tmpg_tl
2075 \tl_new:N \l__knot_redraws_tl
2076 \tl_new:N \l__knot_clip_width_tl
2077 \tl_new:N \l__knot_name_tl
2078 \tl_new:N \l__knot_node_tl
2079
2080 \int_new:N \l__knot_tmpa_int
2081 \int_new:N \l__knot_strands_int
2082 \int_new:N \l__knot_intersections_int
2083 \int_new:N \l__knot_filaments_int
2084 \int_new:N \l__knot_component_start_int
2085
2086 \dim_new:N \l__knot_tmpa_dim
2087 \dim_new:N \l__knot_tmpb_dim
2088 \dim_new:N \l__knot_tmpc_dim
2089 \dim_new:N \l__knot_tolerance_dim
2090 \dim_new:N \l__knot_clip_radius_dim
2091
2092 \bool_new:N \l__knot_draft_bool
2093 \bool_new:N \l__knot_ignore_ends_bool
2094 \bool_new:N \l__knot_self_intersections_bool
2095 \bool_new:N \l__knot_splits_bool
2096 \bool_new:N \l__knot_super_draft_bool
2097
2098 \bool_new:N \l__knot_prepend_prev_bool
2099 \bool_new:N \l__knot_append_next_bool
2100 \bool_new:N \l__knot_skip_bool
2101
2102 \bool_set_true:N \l__knot_ignore_ends_bool
    Configuration is via TikZ keys and styles.
2103 \tikzset{
2104   knot/.code={
2105     \tl_if_eq:nnTF {#1} {none}
```



```

2106 {
2107   \tikz@addmode{\tikz@mode@doublefalse}
2108 }
2109 {
2110   \tikz@addmode{\tikz@mode@doubletrue}
2111   \tl_if_empty:nTF {#1}
2112   {
2113     \pgfsetinnerstrokecolor{\tikz@strokecolor}
2114   }
2115   {
2116     \pgfsetinnerstrokecolor{#1}
2117   }
2118   \tikz@addoption{\pgfsetstrokecolor{knotbg}}
2119   \tl_set:Nn \tikz@double@setup{
2120     \pgfsetinnerlinewidth{\pgflinewidth}
2121     \pgfsetlinewidth{\dim_eval:n {\tl_use:N \l__knot_gap_tl \pgflinewidth}}
2122   }
2123 }
2124 },
2125 knot~ gap/.store~ in=\l__knot_gap_tl,
2126 knot~ gap=3,
2127 knot~ diagram/.is~family,
2128 knot~ diagram/.unknown/.code={
2129   \tl_set_eq:NN \l__knot_tmpa_tl \pgfkeyscurrentname
2130   \pgfkeysalso{
2131     /tikz/\l__knot_tmpa_tl=#1
2132   }
2133 },
2134 background~ colour/.code={%
2135   \colorlet{knotbg}{#1}%
2136 },
2137 background~ color/.code={%
2138   \colorlet{knotbg}{#1}%
2139 },
2140 background~ colour=white,
2141 knot~ diagram,
2142 name/.store~in=\l__knot_name_tl,
2143 name={knot},
2144 every~ strand/.style={draw},
2145 ignore~ endpoint~ intersections/.code={
2146   \tl_if_eq:nnTF {#1} {true}
2147   {
2148     \bool_set_true:N \l__knot_ignore_ends_bool
2149   }
2150   {
2151     \bool_set_false:N \l__knot_ignore_ends_bool
2152   }
2153 },
2154 ignore~ endpoint~ intersections/.default=true,
2155 consider~ self~ intersections/.is~choice,

```

```

2156 consider~ self~ intersections/true/.code={
2157   \bool_set_true:N \l__knot_self_intersections_bool
2158   \bool_set_true:N \l__knot_splits_bool
2159 },
2160 consider~ self~ intersections/false/.code={
2161   \bool_set_false:N \l__knot_self_intersections_bool
2162   \bool_set_false:N \l__knot_splits_bool
2163 },
2164 consider~ self~ intersections/no~ splits/.code={
2165   \bool_set_true:N \l__knot_self_intersections_bool
2166   \bool_set_false:N \l__knot_splits_bool
2167 },
2168 consider~ self~ intersections/.default={true},
2169 clip~ radius/.code={
2170   \dim_set:Nn \l__knot_clip_radius_dim {#1}
2171 },
2172 clip~ radius=10pt,
2173 end~ tolerance/.code={
2174   \dim_set:Nn \l__knot_tolerance_dim {#1}
2175 },
2176 end~ tolerance=14pt,
2177 clip~ width/.code={
2178   \tl_set:Nn \l__knot_clip_width_tl {#1}
2179 },
2180 clip~ width=3,
2181 flip~ crossing/.code={%
2182   \tl_clear_new:c {l__knot_crossing_#1}
2183   \tl_set:cn {l__knot_crossing_#1} {x}
2184 },
2185 draft~ mode/.is~ choice,
2186 draft~ mode/off/.code={%
2187   \bool_set_false:N \l__knot_draft_bool
2188   \bool_set_false:N \l__knot_super_draft_bool
2189 },
2190 draft~ mode/crossings/.code={%
2191   \bool_set_true:N \l__knot_draft_bool
2192   \bool_set_false:N \l__knot_super_draft_bool
2193 },
2194 draft~ mode/strands/.code={%
2195   \bool_set_true:N \l__knot_draft_bool
2196   \bool_set_true:N \l__knot_super_draft_bool
2197 },
2198 draft/.is~ family,
2199 draft,
2200 crossing~ label/.style={
2201   overlay,
2202   fill=white,
2203   fill~ opacity=.5,
2204   text~ opacity=1,
2205   text=blue,

```

```

2206     pin~ edge={blue,<-}
2207   },
2208   strand~ label/.style={
2209     overlay,
2210     circle,
2211     draw=purple,
2212     fill=white,
2213     fill~ opacity=.5,
2214     text~ opacity=1,
2215     text=purple,
2216     inner~ sep=0pt
2217   },
2218 }

```

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

```

2219 \cs_new_nopar:Npn \knot_apply_style:N #1
2220 {
2221   \tl_if_exist:NT #1 {
2222     \exp_args:NW \tikzset #1
2223   }
2224 }
2225 \cs_generate_variant:Nn \knot_apply_style:N {c}

```

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

```

2226 \NewDocumentCommand \flipcrossings {m}
2227 {
2228   \tikzset{knot~ diagram/flip~ crossing/.list={#1}}%
2229 }

```

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

```

2230 \NewDocumentCommand \strand { O{} }
2231 {
2232   \int_incr:N \l__knot_strands_int
2233   \tl_clear_new:c {l__knot_options_strand \int_use:N \l__knot_strands_int}
2234   \tl_set:cn {l__knot_options_strand \int_use:N \l__knot_strands_int} {#1}
2235   \path[#1,save~ spath=knot strand \int_use:N \l__knot_strands_int]
2236 }

```

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

```

2237 \NewDocumentEnvironment{knot} { O{} }
2238 {
2239   \knot_initialise:n {#1}
2240 }
2241 {
2242   \knot_render:
2243 }

```

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

```
2244 \cs_new_protected_nopar:Npn \knot_initialise:n #1
2245 {
2246   \tikzset{knot~ diagram/.cd,every~ knot~ diagram/.try,#1}
2247   \int_zero:N \l__knot_strands_int
2248   \tl_clear:N \l__knot_redraws_tl
2249 }
```

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

```
2250 \cs_new_protected_nopar:Npn \knot_render:
2251 {
```

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

```
2252   \pgfscope
2253   \pgftransformreset
```

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

```
2254   \int_step_function:nnnN {1} {1} {\l__knot_strands_int} \knot_draw_strand:n
```

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

```
2255   \bool_if:NF \l__knot_super_draft_bool
2256   {
```

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.

```
2257   \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.

```
2258   \bool_if:NTF \l__knot_self_intersections_bool
2259   {
2260     \knot_split_strands:
2261     \int_set_eq:NN \l__knot_tmpa_int \l__knot_filaments_int
2262     \tl_set:Nn \l__knot_prefix_tl {filament}
2263   }
2264   {
2265     \int_set_eq:NN \l__knot_tmpa_int \l__knot_strands_int
2266     \tl_set:Nn \l__knot_prefix_tl {strand}
2267   }
```

Initialise the intersection count.

```
2268   \int_gzero:N \l__knot_intersections_int
```

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

```
2269   \bool_if:NTF \l__knot_draft_bool
2270   {
2271     \tl_set:Nn \l__knot_node_tl
2272     {\node[coordinate,pin={knot~ diagram/draft/crossing~ label}]{\int_use:N \l__knot_interse
2273   }
2274   {
2275     \tl_set:Nn \l__knot_node_tl {\node[coordinate]}
2276   }
```

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

```

2277 \int_step_variable:nnnNn {1} {1} {\l__knot_tmpa_int - 1} \l__knot_tmpa_tl
2278 {
2279 \int_step_variable:nnnNn {\tl_use:N \l__knot_tmpa_tl + 1} {1} {\l__knot_tmpa_int} \
2280 {
2281 \knot_intersections:VV \l__knot_tmpa_tl \l__knot_tmpb_tl
2282 }
2283 }

```

If any redraws were requested, do them here.

```

2284 \tl_use:N \l__knot_redraws_tl
2285 }

```

Close the scope

```

2286 \endpgfscope
2287 }

```

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

```

2288 \cs_new_protected_nopar:Npn \knot_draw_strand:n #1
2289 {
2290 \pgfscope
2291 \group_begin:
2292 \tl_set:Nn \l_tmpa_tl {knot~ diagram/every~ strand/.try,}
2293 \tl_put_right:Nv \l_tmpa_tl {l__knot_options_strand #1}
2294 \tl_put_right:Nn \l_tmpa_tl {,knot~ diagram/only~ when~ rendering/.try,only~ when~ rendering}
2295 \spath_tikz_path:Vn \l_tmpa_tl {knot strand #1}
2296 \group_end:
2297 \endpgfscope
2298 }
2299 \cs_generate_variant:Nn \tl_put_right:Nn {Nv}

```

`\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.

```

2300 \cs_new_protected_nopar:Npn \knot_draw_labels:n #1
2301 {
2302 \bool_if:NT \l__knot_draft_bool
2303 {
2304 \spath_get:nnN {knot strand #1} {final point} \l__knot_tmpb_tl
2305 \dim_set:Nn \l__knot_tmpa_dim {\tl_item:Nn \l__knot_tmpb_tl {1}}
2306 \dim_set:Nn \l__knot_tmpb_dim {\tl_item:Nn \l__knot_tmpb_tl {2}}
2307 \node[knot~ diagram/draft/strand~label] at (\l__knot_tmpa_dim,\l__knot_tmpb_dim) {#1};
2308 \spath_get:nnN {knot strand #1} {initial point} \l__knot_tmpb_tl
2309 \dim_set:Nn \l__knot_tmpa_dim {\tl_item:Nn \l__knot_tmpb_tl {1}}
2310 \dim_set:Nn \l__knot_tmpb_dim {\tl_item:Nn \l__knot_tmpb_tl {2}}
2311 \node[knot~ diagram/draft/strand~label] at (\l__knot_tmpa_dim,\l__knot_tmpb_dim) {#1};
2312 }
2313 \bool_if:nT {
2314 \l__knot_self_intersections_bool
2315 &&

```

```

2316 \l__knot_splits_bool
2317 }
2318 {
2319 \tl_clear:N \l__knot_tmpa_tl
2320 \spath_map_segment_function:nN {knot strand #1} \knot_split_self_intersects:NN
2321 \spath_put:nnV {knot strand #1} {path} \l__knot_tmpa_tl
2322 }
2323 }

```

\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!

```

2324 \cs_new_protected_nopar:Npn \knot_split_self_intersects:NN #1#2
2325 {
2326 \tl_case:Nnn #1
2327 {
2328 \g__spath_curvetoa_tl
2329 {
2330 \fp_set:Nn \l_tmpa_fp
2331 {
2332 (\tl_item:Nn #2 {3} - 3 * \tl_item:Nn #2 {6} + 3 * \tl_item:Nn #2 {9} - \tl_item:Nn #2
2333 *
2334 (3 * \tl_item:Nn #2 {8} - 3 * \tl_item:Nn #2 {11})
2335 -
2336 (\tl_item:Nn #2 {2} - 3 * \tl_item:Nn #2 {5} + 3 * \tl_item:Nn #2 {8} - \tl_item:Nn #2
2337 *
2338 (3 * \tl_item:Nn #2 {9} - 3 * \tl_item:Nn #2 {12})
2339 }
2340 \fp_set:Nn \l_tmpb_fp
2341 {
2342 (\tl_item:Nn #2 {2} - 3 * \tl_item:Nn #2 {5} + 3 * \tl_item:Nn #2 {8} - \tl_item:Nn #2
2343 *
2344 (3 * \tl_item:Nn #2 {6} - 6 * \tl_item:Nn #2 {9} + 3 * \tl_item:Nn #2 {12})
2345 -
2346 (\tl_item:Nn #2 {3} - 3 * \tl_item:Nn #2 {6} + 3 * \tl_item:Nn #2 {9} - \tl_item:Nn #2
2347 *
2348 (3 * \tl_item:Nn #2 {5} - 6 * \tl_item:Nn #2 {8} + 3 * \tl_item:Nn #2 {11})
2349 }
2350 \fp_compare:nTF
2351 {
2352 \l_tmpb_fp != 0
2353 }
2354 {
2355 \fp_set:Nn \l_tmpa_fp {.5 * \l_tmpa_fp / \l_tmpb_fp}
2356 \fp_compare:nTF
2357 {
2358 0 < \l_tmpa_fp && \l_tmpa_fp < 1
2359 }

```

```

2360     {
2361         \spath_split_curve:VVNN \l_tmpa_fp #2 \l_tmpa_tl \l_tmpb_tl
2362         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2363         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2364         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2365         \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
2366         \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
2367         \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
2368         \tl_put_right:NV \l__knot_tmpa_tl \l_tmpa_tl
2369         \tl_put_right:NV \l__knot_tmpa_tl \l_tmpb_tl
2370     }
2371     {
2372         \tl_set_eq:NN \l_tmpa_tl #2
2373         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2374         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2375         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2376         \tl_put_right:NV \l__knot_tmpa_tl \l_tmpa_tl
2377     }
2378 }
2379 {
2380     \tl_set_eq:NN \l_tmpa_tl #2
2381     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2382     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2383     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2384     \tl_put_right:NV \l__knot_tmpa_tl \l_tmpa_tl
2385 }
2386 }
2387 \g__spath_lineto_tl
2388 {
2389     \tl_set_eq:NN \l_tmpa_tl #2
2390     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2391     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2392     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
2393     \tl_put_right:NV \l__knot_tmpa_tl \l_tmpa_tl
2394 }
2395 }
2396 {
2397     \tl_put_right:NV \l__knot_tmpa_tl #2
2398 }
2399 }

```

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

```

2400 \cs_new_protected_nopar:Npn \knot_intersections:nn #1#2
2401 {
2402     \group_begin:
2403     \tl_set_eq:NN \l__knot_tmpa_tl \l__knot_prefix_tl
2404     \tl_put_right:Nn \l__knot_tmpa_tl {#1}
2405     \tl_set_eq:NN \l__knot_tmpb_tl \l__knot_prefix_tl
2406     \tl_put_right:Nn \l__knot_tmpb_tl {#2}
2407     \spath_get:nnN {knot \tl_use:N \l__knot_tmpa_tl} {path} \l__knot_tmpe_tl

```

```

2408 \spath_get:nnN {knot \tl_use:N \l__knot_tmpb_tl} {path} \l__knot_tmpd_tl
2409
2410 \pgfintersectionofpaths{\pgfsetpath\l__knot_tmpc_tl}{\pgfsetpath\l__knot_tmpd_tl}
2411
2412 \int_compare:nT {\pgfintersectionsolutions > 0}
2413 {
2414   \int_step_function:nnnN {1} {1} {\pgfintersectionsolutions} \knot_do_intersection:n
2415 }
2416 \group_end:
2417 }

```

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

```

2418 \cs_new_protected_nopar:Npn \knot_do_intersection:n #1
2419 {

```

Get the intersection coordinates.

```

2420 \pgfpointintersectionsolution{#1}
2421 \dim_set:Nn \l__knot_tmpa_dim {\pgf@x}
2422 \dim_set:Nn \l__knot_tmpb_dim {\pgf@y}

```

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

```

2423 \bool_set_false:N \l__knot_skip_bool
2424 \bool_if:NT \l__knot_self_intersections_bool
2425 {

```

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

```

2426 \tl_set:Nn \l_tmpa_tl {knot previous}
2427 \tl_put_right:NV \l_tmpa_tl \l__knot_tmpa_tl
2428 \tl_set:Nv \l_tmpa_tl \l_tmpa_tl
2429 \tl_if_eq:NNT \l_tmpa_tl \l__knot_tmpb_tl
2430 {
2431   \knot_test_endpoint:VnT \l__knot_tmpb_tl {final point}
2432   {
2433     \bool_set_true:N \l__knot_skip_bool
2434   }
2435 }

```

```

2436
2437 \tl_set:Nn \l_tmpa_tl {knot previous}
2438 \tl_put_right:NV \l_tmpa_tl \l__knot_tmpb_tl
2439 \tl_set:Nv \l_tmpa_tl \l_tmpa_tl
2440 \tl_if_eq:NNT \l_tmpa_tl \l__knot_tmpa_tl
2441 {
2442   \knot_test_endpoint:VnT \l__knot_tmpa_tl {final point}
2443   {
2444     \bool_set_true:N \l__knot_skip_bool
2445   }
2446 }
2447 }

```



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

```

2448 \bool_if:NT \l__knot_ignore_ends_bool
2449 {
2450 \knot_test_endpoint:VnT \l__knot_tmpa_tl {initial point}
2451 {
2452 \bool_set_true:N \l__knot_skip_bool
2453 }
2454 \knot_test_endpoint:VnT \l__knot_tmpa_tl {final point}
2455 {
2456 \bool_set_true:N \l__knot_skip_bool
2457 }
2458 \knot_test_endpoint:VnT \l__knot_tmpb_tl {initial point}
2459 {
2460 \bool_set_true:N \l__knot_skip_bool
2461 }
2462 \knot_test_endpoint:VnT \l__knot_tmpb_tl {final point}
2463 {
2464 \bool_set_true:N \l__knot_skip_bool
2465 }
2466 }

```

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

```

2467 \bool_if:NF \l__knot_skip_bool
2468 {
2469
2470 \int_gincr:N \l__knot_intersections_int

```

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

```

2471 \bool_if:nTF
2472 {
2473 \tl_if_exist_p:c {l__knot_crossing_ \int_use:N
2474 \l__knot_intersections_int}
2475 &&
2476 ! \tl_if_empty_p:c {l__knot_crossing_ \int_use:N
2477 \l__knot_intersections_int}
2478 }
2479 {
2480 \tl_set_eq:NN \l__knot_tmpg_tl \l__knot_tmpb_tl
2481 }
2482 {
2483 \tl_set_eq:NN \l__knot_tmpg_tl \l__knot_tmpa_tl
2484 }

```

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.

```

2485 \bool_if:NT \l__knot_self_intersections_bool
2486 {

```

```

2487 \knot_test_endpoint:VnT \l__knot_tmpg_tl {initial point}
2488 {
2489   \bool_set_true:N \l__knot_prepend_prev_bool
2490 }
2491 {
2492   \bool_set_false:N \l__knot_prepend_prev_bool
2493 }
2494
2495 \knot_test_endpoint:VnT \l__knot_tmpg_tl {final point}
2496 {
2497   \bool_set_true:N \l__knot_append_next_bool
2498 }
2499 {
2500   \bool_set_false:N \l__knot_append_next_bool
2501 }

```

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

```

2502 \bool_if:nT
2503 {
2504   \l__knot_prepend_prev_bool || \l__knot_append_next_bool
2505 }
2506 {
2507   \spath_clone:nn {knot \tl_use:N \l__knot_tmpg_tl}
2508   {knot \tl_use:N \l__knot_prefix_tl -1}
2509
2510   \tl_set_eq:cc {l__knot_options_ \tl_use:N \l__knot_prefix_tl -1} {l__knot_options_ \tl_use:N \l__knot_prefix_tl -1}
2511
2512   \bool_if:NT \l__knot_prepend_prev_bool
2513   {
2514     \spath_prepend_no_move:nn {knot \tl_use:N \l__knot_prefix_tl -1} {knot \tl_use:c {knot \tl_use:N \l__knot_prefix_tl -1}}

```

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

```

2515   \bool_if:NT \l__knot_splits_bool
2516   {
2517     \knot_test_endpoint:vnT {knot previous \tl_use:N \l__knot_tmpg_tl} {initial point}
2518     {
2519       \spath_get:nnN {knot \tl_use:N \l__knot_prefix_tl -1} {path} \l_tmpa_tl
2520       \spath_prepend_no_move:nn {knot \tl_use:N \l__knot_prefix_tl -1} {knot \tl_use:c {knot \tl_use:N \l__knot_prefix_tl -1}}
2521       \spath_get:nnN {knot \tl_use:N \l__knot_prefix_tl -1} {path} \l_tmpa_tl
2522     }
2523   }
2524 }
2525 }

```

Now the same for appending.

```

2526 \bool_if:NT \l__knot_append_next_bool
2527 {
2528   \spath_append_no_move:nn {knot \tl_use:N \l__knot_prefix_tl -1} {knot \tl_use:c {knot \tl_use:N \l__knot_prefix_tl -1}}
2529   \bool_if:NT \l__knot_splits_bool
2530   {

```

```

2531         \knot_test_endpoint:vnT {knot previous \tl_use:N \l__knot_tmpg_tl} {final point}
2532         {
2533             \spath_append_no_move:nn {knot \tl_use:N \l__knot_prefix_tl -1} {knot \tl_use:c
2534
2535         }
2536     }
2537 }
2538
2539 \tl_set:Nn \l__knot_tmpg_tl {\tl_use:N \l__knot_prefix_tl -1}
2540 }
2541 }

```

Now we render the crossing.

```

2542 \group_begin:
2543 \tikzset{knot~ diagram/intersection~ \int_use:N \l__knot_intersections_int/.try}
2544 \knot_draw_crossing:nVV {\tl_use:N \l__knot_tmpg_tl} \l__knot_tmpa_dim \l__knot_tmpb_dim
2545 \group_end:

```

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

```

2546 \tl_use:N \l__knot_node_tl (\l__knot_name_tl \c_space_tl \int_use:N \l__knot_intersections
2547 }
2548 }
2549
2550 \cs_generate_variant:Nn \knot_intersections:nn {VV}

```

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

```

2551 \prg_new_conditional:Npnn \knot_test_endpoint:N #1 {p,T,F,TF}
2552 {
2553     \dim_compare:nTF
2554     {
2555         \dim_abs:n {\pgf@x - \tl_item:Nn #1 {1}}
2556         +
2557         \dim_abs:n {\pgf@y - \tl_item:Nn #1 {2}}
2558         <
2559         \l__knot_tolerance_dim
2560     }
2561     {
2562         \prg_return_true:
2563     }
2564     {
2565         \prg_return_false:
2566     }
2567 }

```

`\knot_test_endpoint:nn` Wrapper around the above.

```

2568 \prg_new_protected_conditional:Npnn \knot_test_endpoint:nn #1#2 {T,F,TF}
2569 {
2570     \spath_get:nnN {knot #1} {#2} \l__knot_tmpd_tl
2571     \knot_test_endpoint:NTF \l__knot_tmpd_tl
2572     {

```

```

2573   \prg_return_true:
2574   }
2575   {
2576   \prg_return_false:
2577   }
2578 }
2579
2580 \cs_generate_variant:Nn \knot_test_endpoint:nnT {VnT,vnT}
2581 \cs_generate_variant:Nn \knot_test_endpoint:nnF {VnF,vnF}
2582 \cs_generate_variant:Nn \knot_test_endpoint:nnTF {VnTF,vnTF}

```

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

```

2583 \cs_new_nopar:Npn \knot_draw_crossing:nnn #1#2#3
2584 {
2585   \group_begin:
2586   \pgfscope
2587   \clip (#2, #3) circle[radius=\l__knot_clip_radius_dim];
2588
2589   \tl_set:Nn \l_tmpa_tl {knot~ diagram/every~ strand/.try,}
2590   \tl_if_exist:cT {l__knot_options_ #1}
2591   {
2592   \tl_put_right:Nv \l_tmpa_tl {l__knot_options_ #1}
2593   }
2594   \tl_put_right:Nn \l_tmpa_tl {,knotbg,line~ width= \tl_use:N \l__knot_clip_width_tl * \pgflinewidth}
2595   \spath_tikz_path:Vn \l_tmpa_tl {knot #1}
2596
2597   \endpgfscope
2598
2599   \pgfscope
2600   \clip (#2, #3) circle[radius=1.1\l__knot_clip_radius_dim];
2601
2602   \tl_set:Nn \l_tmpa_tl {knot~ diagram/every~ strand/.try,}
2603   \tl_if_exist:cT {l__knot_options_ #1}
2604   {
2605   \tl_put_right:Nv \l_tmpa_tl {l__knot_options_ #1}
2606   }
2607   \tl_put_right:Nn \l_tmpa_tl {,knot~ diagram/only~ when~ rendering/.try,only~ when~ rendering}
2608   \spath_tikz_path:Vn \l_tmpa_tl {knot #1}
2609
2610   \endpgfscope
2611   \group_end:
2612 }
2613
2614 \cs_generate_variant:Nn \knot_draw_crossing:nnn {nVV}

```

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

```

2615 \cs_new_protected_nopar:Npn \knot_split_strands:
2616 {
2617   \int_gzero:N \l__knot_filaments_int

```

```

2618 \int_step_function:nnnN {1} {1} {\l__knot_strands_int} \knot_split_strand:n
2619 \int_step_function:nnnN {1} {1} {\l__knot_filaments_int} \knot_compute_nexts:n
2620 }

```

`\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.

```

2621 \cs_new_protected_nopar:Npn \knot_compute_nexts:n #1
2622 {
2623   \tl_clear_new:c {knot next \tl_use:c {knot previous filament #1}}
2624   \tl_set:cn {knot next \tl_use:c {knot previous filament #1}} {filament #1}
2625 }

```

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

```

2626 \cs_new_protected_nopar:Npn \knot_split_strand:n #1
2627 {
2628   \int_set_eq:NN \l__knot_component_start_int \l__knot_filaments_int
2629   \int_incr:N \l__knot_component_start_int
2630   \spath_map_segment_function:nN {knot strand #1} \knot_save_filament:NN
2631 }

```

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

```

2632 \cs_new_protected_nopar:Npn \knot_save_filament:NN #1#2
2633 {
2634   \tl_case:Nnn #1
2635   {
2636     \g__spath_moveto_tl
2637     {
2638       \int_compare:nT {\l__knot_component_start_int < \l__knot_filaments_int}
2639       {
2640         \int_set_eq:NN \l__knot_component_start_int \l__knot_filaments_int
2641       }
2642     }
2643     \g__spath_lineto_tl
2644     {
2645       \int_gincr:N \l__knot_filaments_int
2646       \spath_clear_new:n {knot filament \int_use:N \l__knot_filaments_int}
2647       \spath_put:nnV {knot filament \int_use:N \l__knot_filaments_int} {path} #2
2648       \tl_clear_new:c {knot previous filament \int_use:N \l__knot_filaments_int}
2649       \int_compare:nF {\l__knot_component_start_int == \l__knot_filaments_int}
2650       {
2651         \tl_set:cx {knot previous filament \int_use:N \l__knot_filaments_int} {filament \int_
2652       }
2653     }
2654     \g__spath_curvetoa_tl
2655     {
2656       \int_gincr:N \l__knot_filaments_int
2657       \spath_clear_new:n {knot filament \int_use:N \l__knot_filaments_int}
2658       \spath_put:nnV {knot filament \int_use:N \l__knot_filaments_int} {path} #2
2659       \tl_clear_new:c {knot previous filament \int_use:N \l__knot_filaments_int}

```

```

2660     \int_compare:nF {\l__knot_component_start_int == \l__knot_filaments_int}
2661     {
2662         \tl_set:cx {knot previous filament \int_use:N \l__knot_filaments_int} {filament \int_
2663     }
2664 }
2665 \g__spath_close_tl
2666 {
2667     \int_gincr:N \l__knot_filaments_int
2668     \spath_clear_new:n {knot filament \int_use:N \l__knot_filaments_int}
2669     \tl_set:eq:NN \l_tmpa_tl #2
2670     \tl_set:Nx \l_tmpa_tl {\tl_item:Nn #2 {1}\tl_item:Nn #2 {2}\tl_item:Nn #2 {3}}
2671     \tl_put_right:NV \l_tmpa_tl \g__spath_lineto_tl
2672     \tl_set:Nx \l_tmpa_tl {\tl_item:Nn #2 {5}\tl_item:Nn #2 {6}}
2673     \spath_put:nnV {knot filament \int_use:N \l__knot_filaments_int} {path} \l_tmpa_tl
2674     \tl_clear_new:c {knot previous filament \int_use:N \l__knot_filaments_int}
2675     \int_compare:nF {\l__knot_component_start_int == \l__knot_filaments_int}
2676     {
2677         \tl_set:cx {knot previous filament \int_use:N \l__knot_filaments_int} {filament \int_
2678     }
2679     \tl_set:cx {knot previous filament \int_use:N \l__knot_component_start_int} {filament \
2680 }
2681 }
2682 {
2683 }
2684 }

```

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

```

2685 \NewDocumentCommand \redraw { m m }
2686 {
2687     \tikz@scan@one@point\pgfutil@firstofone #2 \relax
2688     \tl_put_right:Nn \l__knot_redraws_tl {\knot_draw_crossing:nnn}
2689     \tl_put_right:Nx \l__knot_redraws_tl {
2690         {strand #1} {\dim_use:N \pgf@x} {\dim_use:N \pgf@y}
2691     }
2692 }
2693 \ExplSyntaxOff

```

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

```

2694 \def\pgf@sh@@knotanchor#1#2{%
2695     \anchor{#2 north west}{%
2696         \csname pgf@anchor@knot #1@north west\endcsname%
2697         \pgf@x=#2\pgf@x%
2698         \pgf@y=#2\pgf@y%
2699     }%
2700     \anchor{#2 north east}{%
2701         \csname pgf@anchor@knot #1@north east\endcsname%
2702         \pgf@x=#2\pgf@x%
2703         \pgf@y=#2\pgf@y%
2704     }%

```

```

2705 \anchor{#2 south west}{%
2706   \csname pgf@anchor@knot #1@south west\endcsname%
2707   \pgf@x=#2\pgf@x%
2708   \pgf@y=#2\pgf@y%
2709 }%
2710 \anchor{#2 south east}{%
2711   \csname pgf@anchor@knot #1@south east\endcsname%
2712   \pgf@x=#2\pgf@x%
2713   \pgf@y=#2\pgf@y%
2714 }%
2715 \anchor{#2 north}{%
2716   \csname pgf@anchor@knot #1@north\endcsname%
2717   \pgf@x=#2\pgf@x%
2718   \pgf@y=#2\pgf@y%
2719 }%
2720 \anchor{#2 east}{%
2721   \csname pgf@anchor@knot #1@east\endcsname%
2722   \pgf@x=#2\pgf@x%
2723   \pgf@y=#2\pgf@y%
2724 }%
2725 \anchor{#2 west}{%
2726   \csname pgf@anchor@knot #1@west\endcsname%
2727   \pgf@x=#2\pgf@x%
2728   \pgf@y=#2\pgf@y%
2729 }%
2730 \anchor{#2 south}{%
2731   \csname pgf@anchor@knot #1@south\endcsname%
2732   \pgf@x=#2\pgf@x%
2733   \pgf@y=#2\pgf@y%
2734 }%
2735 }

```

knotcrossing

```

2736 \pgfdeclareshape{knot crossing}
2737 {
2738   \inheritsavedanchors[from=circle] % this is nearly a circle
2739   \inheritanchorborder[from=circle]
2740   \inheritanchor[from=circle]{north}
2741   \inheritanchor[from=circle]{north west}
2742   \inheritanchor[from=circle]{north east}
2743   \inheritanchor[from=circle]{center}
2744   \inheritanchor[from=circle]{west}
2745   \inheritanchor[from=circle]{east}
2746   \inheritanchor[from=circle]{mid}
2747   \inheritanchor[from=circle]{mid west}
2748   \inheritanchor[from=circle]{mid east}
2749   \inheritanchor[from=circle]{base}
2750   \inheritanchor[from=circle]{base west}
2751   \inheritanchor[from=circle]{base east}
2752   \inheritanchor[from=circle]{south}

```

```

2753 \inheritanchor[from=circle]{south west}
2754 \inheritanchor[from=circle]{south east}
2755 \inheritanchorborder[from=circle]
2756 \pgf@sh@@knotanchor{crossing}{2}
2757 \pgf@sh@@knotanchor{crossing}{3}
2758 \pgf@sh@@knotanchor{crossing}{4}
2759 \pgf@sh@@knotanchor{crossing}{8}
2760 \pgf@sh@@knotanchor{crossing}{16}
2761 \pgf@sh@@knotanchor{crossing}{32}
2762 \backgroundpath{
2763   \pgfutil@tempdima=\radius%
2764   \pgfmathsetlength{\pgf@xb}{\pgfkeysvalueof{/pgf/outer xsep}}%
2765   \pgfmathsetlength{\pgf@yb}{\pgfkeysvalueof{/pgf/outer ysep}}%
2766   \ifdim\pgf@xb<\pgf@yb%
2767     \advance\pgfutil@tempdima by-\pgf@yb%
2768   \else%
2769     \advance\pgfutil@tempdima by-\pgf@xb%
2770   \fi%
2771 }
2772 }

```

#### knotovercross

```

2773 \pgfdeclareshape{knot over cross}
2774 {
2775   \inheritsavedanchors[from=rectangle] % this is nearly a circle
2776   \inheritanchorborder[from=rectangle]
2777   \inheritanchor[from=rectangle]{north}
2778   \inheritanchor[from=rectangle]{north west}
2779   \inheritanchor[from=rectangle]{north east}
2780   \inheritanchor[from=rectangle]{center}
2781   \inheritanchor[from=rectangle]{west}
2782   \inheritanchor[from=rectangle]{east}
2783   \inheritanchor[from=rectangle]{mid}
2784   \inheritanchor[from=rectangle]{mid west}
2785   \inheritanchor[from=rectangle]{mid east}
2786   \inheritanchor[from=rectangle]{base}
2787   \inheritanchor[from=rectangle]{base west}
2788   \inheritanchor[from=rectangle]{base east}
2789   \inheritanchor[from=rectangle]{south}
2790   \inheritanchor[from=rectangle]{south west}
2791   \inheritanchor[from=rectangle]{south east}
2792   \inheritanchorborder[from=rectangle]
2793   \backgroundpath{
2794     \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
2795     \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
2796     \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
2797     \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
2798   }
2799   \foregroundpath{
2800 % store lower right in xa/ya and upper right in xb/yb

```



```

2801 \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
2802 \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
2803 \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
2804 \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
2805 }
2806 }

```

#### knotundercross

```

2807 \pgfdeclareshape{knot under cross}
2808 {
2809 \inheritsavedanchors[from=rectangle] % this is nearly a circle
2810 \inheritanchorborder[from=rectangle]
2811 \inheritanchor[from=rectangle]{north}
2812 \inheritanchor[from=rectangle]{north west}
2813 \inheritanchor[from=rectangle]{north east}
2814 \inheritanchor[from=rectangle]{center}
2815 \inheritanchor[from=rectangle]{west}
2816 \inheritanchor[from=rectangle]{east}
2817 \inheritanchor[from=rectangle]{mid}
2818 \inheritanchor[from=rectangle]{mid west}
2819 \inheritanchor[from=rectangle]{mid east}
2820 \inheritanchor[from=rectangle]{base}
2821 \inheritanchor[from=rectangle]{base west}
2822 \inheritanchor[from=rectangle]{base east}
2823 \inheritanchor[from=rectangle]{south}
2824 \inheritanchor[from=rectangle]{south west}
2825 \inheritanchor[from=rectangle]{south east}
2826 \inheritanchorborder[from=rectangle]
2827 \backgroundpath{
2828 \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
2829 \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
2830 \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
2831 \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
2832 }
2833 \foregroundpath{
2834 % store lower right in xa/ya and upper right in xb/yb
2835 \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
2836 \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
2837 \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
2838 \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
2839 }
2840 }

```

#### knotvert

```

2841 \pgfdeclareshape{knot vert}
2842 {
2843 \inheritsavedanchors[from=rectangle] % this is nearly a circle
2844 \inheritanchorborder[from=rectangle]
2845 \inheritanchor[from=rectangle]{north}

```

```

2846 \inheritanchor[from=rectangle]{north west}
2847 \inheritanchor[from=rectangle]{north east}
2848 \inheritanchor[from=rectangle]{center}
2849 \inheritanchor[from=rectangle]{west}
2850 \inheritanchor[from=rectangle]{east}
2851 \inheritanchor[from=rectangle]{mid}
2852 \inheritanchor[from=rectangle]{mid west}
2853 \inheritanchor[from=rectangle]{mid east}
2854 \inheritanchor[from=rectangle]{base}
2855 \inheritanchor[from=rectangle]{base west}
2856 \inheritanchor[from=rectangle]{base east}
2857 \inheritanchor[from=rectangle]{south}
2858 \inheritanchor[from=rectangle]{south west}
2859 \inheritanchor[from=rectangle]{south east}
2860 \inheritanchorborder[from=rectangle]
2861 \backgroundpath{
2862 % store lower right in xa/ya and upper right in xb/yb
2863   \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
2864   \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
2865   \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
2866   \pgfpathlineto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
2867   \pgfpathmoveto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
2868   \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
2869 }
2870 }

```

knothoriz

```

2871 \pgfdeclareshape{knot horiz}
2872 {
2873   \inheritsavedanchors[from=rectangle] % this is nearly a circle
2874   \inheritanchorborder[from=rectangle]
2875   \inheritanchor[from=rectangle]{north}
2876   \inheritanchor[from=rectangle]{north west}
2877   \inheritanchor[from=rectangle]{north east}
2878   \inheritanchor[from=rectangle]{center}
2879   \inheritanchor[from=rectangle]{west}
2880   \inheritanchor[from=rectangle]{east}
2881   \inheritanchor[from=rectangle]{mid}
2882   \inheritanchor[from=rectangle]{mid west}
2883   \inheritanchor[from=rectangle]{mid east}
2884   \inheritanchor[from=rectangle]{base}
2885   \inheritanchor[from=rectangle]{base west}
2886   \inheritanchor[from=rectangle]{base east}
2887   \inheritanchor[from=rectangle]{south}
2888   \inheritanchor[from=rectangle]{south west}
2889   \inheritanchor[from=rectangle]{south east}
2890   \inheritanchorborder[from=rectangle]
2891   \foregroundpath{
2892 % store lower right in xa/ya and upper right in xb/yb
2893   \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y

```

```
2894 \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
2895 \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
2896 \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
2897 \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
2898 \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
2899 }
2900 }
```