

The `spath3` package: code

Andrew Stacey

loopspace@mathforge.org

v2.2 from 2021/02/05

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 was originally not really intended for use by end users but as a foundation on which other packages can be built. However, over the years I've found myself using it at ever higher levels and so a set of interfaces has been designed using TikZ keys.

It also provides the engine that drives a few other packages, such as the `calligraphy`, `knot`, and `penrose` packages. The first two of these are subpackages of this one. The `calligraphy` package simulates a calligraphic pen stroking a path. The `knots` package can be used to draw knot (and similar) diagrams.

For usage, see the documentation of the following packages (`\texdoc <package>`):

- `calligraphy`
- `knots`
- `penrose`
- `spath3` (*this* document is the code, there's another which focusses on usage)

2 Technical Details

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.

In the original implementation, I wrapped this token list in a `prop` to store useful information along with the path. Over time, this additional structure has proved a little unwieldy and I've pared it back to working primarily with the original soft path as a token list.

A frequent use of this package is to break a path into pieces and do something with each of those pieces. To that end, there are various words that I use to describe the levels of the structure of a path.

At the top level is the path itself. At the bottom level are the triples of the form `\macro{dim}{dim}`, as described above. In between these are the *segments* and *components*.

A *segment* is a minimal drawing piece. Thus it might be a straight line or a Bézier curve. When a path is broken into segments then each segment is a complete path so it isn't simply a selection of triples from the original path.

A *component* is a minimal connected section of the path. So every component starts with a move command and continues until the next move command. For ease of implementation (and to enable a copperplate pen in the calligraphy package!), an isolated move is considered as a component. Thus the following path consists of three components:

```
\path (0,0) -- (1,0) (2,0) (3,0) to[out=0,in=90] (4,0);
```

3 Implementation

3.1 Initialisation

```
1 <@=@spath>
```

Load the L^AT_EX3 foundation and register us as a L^AT_EX3 package.

```
2 \NeedsTeXFormat{LaTeX2e}
```

```
3 \RequirePackage{expl3}
```

```
4 \RequirePackage{pgf}
```

```
5 \ProvidesExplPackage {spath3} {2021/02/05} {2.2} {Functions for  
6 manipulating PGF soft paths}
```

```
7 \RequirePackage{xparse}
```

Utilities copied from <https://github.com/loopspace/LaTeX3-Utilities> for adding something in braces to a token list. I find I use this quite a lot in my packages.

```
8 \cs_new_protected:Nn \__spath_tl_put_right_braced:Nn
```

```
9 {
```

```
10 \tl_put_right:Nn #1 { { #2 } }
```

```
11 }
```

```
12 \cs_generate_variant:Nn \__spath_tl_put_right_braced:Nn { NV, cV, cv, Nx, cx }
```

```
13
```

```
14 \cs_new_protected:Nn \__spath_tl_gput_right_braced:Nn
```

```
15 {
```

```
16 \tl_gput_right:Nn #1 { { #2 } }
```

```
17 }
```

```
18 \cs_generate_variant:Nn \__spath_tl_gput_right_braced:Nn { NV, cV, cv, Nx, cx }
```

```
19 \cs_new_protected:Nn \__spath_tl_put_left_braced:Nn
```

```
20 {
```

```
21 \tl_put_left:Nn #1 { { #2 } }
```

```
22 }
```

```
23 \cs_generate_variant:Nn \__spath_tl_put_left_braced:Nn { NV, cV, cv, Nx, cx }
```

```
24
```

```
25 \cs_new_protected:Nn \__spath_tl_gput_left_braced:Nn
```

```
26 {
```

```
27 \tl_gput_left:Nn #1 { { #2 } }
```

```
28 }
```

```
29 \cs_generate_variant:Nn \__spath_tl_gput_left_braced:Nn { NV, cV, cv, Nx, cx }
```

I had to think a bit about how to get \TeX to work the way I wanted. I'm really defining *functions* but \TeX doesn't really have that concept, even with all the amazing \LaTeX 3 stuff. The main issue I had was with scoping and return values. By default, \TeX functions aren't scoped – they work on the same level as the calling functions. To protect the internals from being overwritten, each core function works inside a group. But then I have to work to get the answer out of it. So each of my core functions finishes by storing its return value in an appropriate `output` variable. The core functions are then wrapped in a more user friendly interface that will take that output and assign it to a variable. This also means that I can deal with local and global versions without duplicating code.

```

30 \tl_new:N \g__spath_output_tl
31 \int_new:N \g__spath_output_int
32 \seq_new:N \g__spath_output_seq
33 \bool_new:N \g__spath_output_bool

```

To avoid creating vast numbers of variables, we provide ourselves with a few that we reuse frequently. For that reason, most of them don't have very exciting names.

These are general purpose variables.

```

34 \tl_new:N \l__spath_tmpa_tl
35 \tl_new:N \l__spath_tmpb_tl
36 \tl_new:N \l__spath_tmpe_tl
37 \tl_new:N \l__spath_tmpe_tl
38 \tl_new:N \l__spath_tmpe_tl
39 \tl_new:N \l__spath_tmpe_tl
40 \tl_new:N \l__spath_tmpe_tl
41 \tl_new:N \l__spath_tmpe_tl
42 \tl_new:N \l__spath_tmpe_tl
43
44 \seq_new:N \l__spath_tmpe_seq
45 \seq_new:N \l__spath_tmpe_seq
46 \seq_new:N \l__spath_tmpe_seq
47
48 \dim_new:N \l__spath_tmpe_dim
49 \dim_new:N \l__spath_tmpe_dim
50
51 \fp_new:N \l__spath_tmpe_fp
52 \fp_new:N \l__spath_tmpe_fp
53 \fp_new:N \l__spath_tmpe_fp
54
55 \int_new:N \l__spath_tmpe_int
56 \int_new:N \l__spath_tmpe_int
57
58 \bool_new:N \l__spath_tmpe_bool

```

Whenever I need more than two `dim` variables it is because I need to remember the position of a move.

```

59 \dim_new:N \l__spath_move_x_dim
60 \dim_new:N \l__spath_move_y_dim

```

Closed paths often need special handling. When it's needed, this will say whether the path is closed or not.

```

61 \bool_new:N \l__spath_closed_bool

```

The intersection routine can't happen inside a group so we need two token lists to hold the paths that we'll intersect.

```

62 \tl_new:N \l__spath_intersecta_tl

```

```
63 \tl_new:N \l__spath_intersectb_tl
```

We need to be able to compare against the macros that can occur in a soft path so these token lists contain them. These are global constants so that they can be used in other packages.

```
64 \tl_const:Nn \c_spath_moveto_tl {\pgfsyssoftpath@movetotoken}
65 \tl_const:Nn \c_spath_lineto_tl {\pgfsyssoftpath@linetotoken}
66 \tl_const:Nn \c_spath_curveto_tl {\pgfsyssoftpath@curvetotoken}
67 \tl_const:Nn \c_spath_curvetoa_tl {\pgfsyssoftpath@curvetosupportatoken}
68 \tl_const:Nn \c_spath_curvetob_tl {\pgfsyssoftpath@curvetosupportbtoken}
69 \tl_const:Nn \c_spath_closepath_tl {\pgfsyssoftpath@closepathtoken}
```

We will want to be able to use anonymous spaths internally, so we create a global counter that we can use to refer to them.

```
70 \int_new:N \g__spath_anon_int
71 \int_gzero:N \g__spath_anon_int
```

And some error messages

```
72 \msg_new:nnn { spath3 } { unknown path construction }
73 { The~ path~ construction~ element~ #1~ is~ not~ currently~ supported.}
```

3.2 Functional Implementation

In the functional approach, we start with a token list containing a soft path and do something to it (either calculate some information or manipulate it in some fashion). We then store that information, or the manipulated path, in an appropriate macro. The macro to store it in is the first argument. These functions occur in two versions, the one with the `g` makes the assignment global.

`\spath_segments_to_seq:Nn` Splits a soft path into *segments*, storing the result in a sequence.

```
\spath_segments_gto_seq:Nn
74 \cs_new_protected_nopar:Npn \__spath_segments_to_seq:n #1
75 {
76   \group_begin:
77   \tl_set:Nn \l__spath_tmpa_tl {#1}
78   \tl_clear:N \l__spath_tmpb_tl
79   \seq_clear:N \l__spath_tmpa_seq
80   \dim_zero:N \l__spath_tmpa_dim
81   \dim_zero:N \l__spath_tmpb_dim
82
83   \bool_until_do:nn {
84     \tl_if_empty_p:N \l__spath_tmpa_tl
85   }
86   {
87     \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
88     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
89     \tl_case:NnF \l__spath_tmpc_tl
90     {
91       \c_spath_moveto_tl
92       {
93         \tl_set_eq:NN \l__spath_tmpb_tl \c_spath_moveto_tl
94         \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
95         \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
96         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
97
98         \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
99         \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
```

```

100      \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
101
102      \tl_set:Nx \l__spath_tmpd_tl {\tl_head:N \l__spath_tmpa_tl}
103      \tl_if_eq:NNF \l__spath_tmpd_tl \c_spath_moveto_tl
104      {
105          \tl_clear:N \l__spath_tmpb_tl
106      }
107
108  }
109
110  \c_spath_lineto_tl
111  {
112      \tl_set_eq:NN \l__spath_tmpb_tl \c_spath_moveto_tl
113      \tl_put_right:Nx \l__spath_tmpb_tl
114      {
115          {\dim_use:N \l__spath_tmpa_dim}
116          {\dim_use:N \l__spath_tmpb_dim}
117      }
118      \tl_put_right:NV \l__spath_tmpb_tl \c_spath_lineto_tl
119
120      \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
121      \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
122      \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
123
124      \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
125      \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
126      \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
127
128  }
129
130  \c_spath_curvetoa_tl
131  {
132      \tl_set_eq:NN \l__spath_tmpb_tl \c_spath_moveto_tl
133      \tl_put_right:Nx \l__spath_tmpb_tl
134      {
135          {\dim_use:N \l__spath_tmpa_dim}
136          {\dim_use:N \l__spath_tmpb_dim}
137      }
138      \tl_put_right:NV \l__spath_tmpb_tl \c_spath_curvetoa_tl
139
140      \prg_replicate:nn {2} {
141          \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
142          \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
143          \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
144          \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
145          \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
146          \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
147      }
148
149      \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
150      \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
151      \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
152
153      \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}

```

```

154         \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
155         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
156
157     }
158
159     \c_spath_closepath_tl
160     {
161         \tl_set_eq:NN \l__spath_tmpb_tl \c_spath_moveto_tl
162         \tl_put_right:Nx \l__spath_tmpb_tl
163         {
164             {\dim_use:N \l__spath_tmpa_dim}
165             {\dim_use:N \l__spath_tmpb_dim}
166         }
167         \tl_put_right:NV \l__spath_tmpb_tl \c_spath_lineto_tl
168
169         \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}}
170         \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
171         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
172
173         \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}}
174         \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
175         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
176
177     }
178
179 }
180 {
181
182     \tl_set_eq:NN \l__spath_tmpb_tl \l__spath_tmpc_tl
183     \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}}
184     \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
185     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
186
187     \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}}
188     \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
189     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
190
191 }
192
193 \tl_if_empty:NF \l__spath_tmpb_tl
194 {
195     \seq_put_right:NV \l__spath_tmpa_seq \l__spath_tmpb_tl
196 }
197 \tl_clear:N \l__spath_tmpb_tl
198 }
199
200 \seq_gclear:N \g__spath_output_seq
201 \seq_gset_eq:NN \g__spath_output_seq \l__spath_tmpa_seq
202 \group_end:
203 }
204 \cs_new_protected_nopar:Npn \spath_segments_to_seq:Nn #1#2
205 {
206     \__spath_segments_to_seq:n {#2}
207     \seq_clear_new:N #1

```

```

208 \seq_set_eq:NN #1 \g__spath_output_seq
209 \seq_gclear:N \g__spath_output_seq
210 }
211 \cs_generate_variant:Nn \spath_segments_to_seq:Nn {NV, cn, cV, Nv, cv}
212 \cs_new_protected_nopar:Npn \spath_segments_gto_seq:Nn #1#2
213 {
214   \__spath_segments_to_seq:n {#2}
215   \seq_clear_new:N #1
216   \seq_gset_eq:NN #1 \g__spath_output_seq
217   \seq_gclear:N \g__spath_output_seq
218 }
219 \cs_generate_variant:Nn \spath_segments_gto_seq:Nn {NV, cn, cV, Nv, cv}

```

(End definition for \spath_segments_to_seq:Nn and \spath_segments_gto_seq:Nn.)

\spath_components_to_seq:Nn Splits a soft path into *components*, storing the result in a sequence or a clist.

```

\spath_components_gto_seq:Nn
\spath_components_to_clist:Nn
\spath_components_gto_clist:Nn
220 \cs_new_protected_nopar:Npn \__spath_components_to_seq:n #1
221 {
222   \group_begin:
223   \tl_set:Nn \l__spath_tmpa_tl {#1}
224   \seq_clear:N \l__spath_tmpa_seq
225   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
226   \tl_put_right:NV \l__spath_tmpa_tl \c_spath_moveto_tl
227   \tl_set_eq:NN \l__spath_tmpb_tl \c_spath_moveto_tl
228   \bool_do_until:nn {
229     \tl_if_empty_p:N \l__spath_tmpa_tl
230   }
231   {
232     \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
233     \tl_if_eq:NNT \l__spath_tmpc_tl \c_spath_moveto_tl
234     {
235       \seq_put_right:NV \l__spath_tmpa_seq \l__spath_tmpb_tl
236       \tl_clear:N \l__spath_tmpb_tl
237     }
238     \tl_if_single:NTF \l__spath_tmpc_tl
239     {
240       \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpc_tl
241     }
242     {
243       \tl_put_right:Nx \l__spath_tmpb_tl {\l__spath_tmpc_tl}
244     }
245     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
246   }
247
248   \seq_gclear:N \g__spath_output_seq
249   \seq_gset_eq:NN \g__spath_output_seq \l__spath_tmpa_seq
250   \group_end:
251 }
252 \cs_new_protected_nopar:Npn \spath_components_to_seq:Nn #1#2
253 {
254   \__spath_components_to_seq:n {#2}
255   \seq_clear_new:N #1
256   \seq_set_eq:NN #1 \g__spath_output_seq
257   \seq_gclear:N \g__spath_output_seq

```

```

258 }
259 \cs_generate_variant:Nn \spath_components_to_seq:Nn {NV, cn, cV, cv, Nv}
260 \cs_new_protected_nopar:Npn \spath_components_gto_seq:Nn #1#2
261 {
262   \__spath_components_to_seq:n {#2}
263   \seq_clear_new:N #1
264   \seq_gset_eq:NN #1 \g__spath_output_seq
265   \seq_gclear:N \g__spath_output_seq
266 }
267 \cs_generate_variant:Nn \spath_components_gto_seq:Nn {NV, cn, cV, cv, Nv}
268 \cs_new_protected_nopar:Npn \spath_components_to_clist:Nn #1#2
269 {
270   \__spath_components_to_seq:n {#2}
271   \clist_clear_new:N #1
272   \clist_set_from_seq:NN #1 \g__spath_output_seq
273   \seq_gclear:N \g__spath_output_seq
274 }
275 \cs_generate_variant:Nn \spath_components_to_clist:Nn {NV, cn, cV, cv, Nv}
276 \cs_new_protected_nopar:Npn \spath_components_gto_clist:Nn #1#2
277 {
278   \__spath_components_to_seq:n {#2}
279   \clist_clear_new:N #1
280   \clist_gset_from_seq:NN #1 \g__spath_output_seq
281   \seq_gclear:N \g__spath_output_seq
282 }
283 \cs_generate_variant:Nn \spath_components_gto_clist:Nn {NV, cn, cV, cv, Nv}

```

(End definition for `\spath_components_to_seq:Nn` and others.)

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

```

284 \cs_new_protected_nopar:Npn \spath_length:n #1
285 {
286   \int_eval:n {\tl_count:n {#1} / 3}
287 }
288 \cs_generate_variant:Nn \spath_length:n {V}

```

(End definition for `\spath_length:n`.)

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

```

\spath_greallength:Nn
289 \cs_new_protected_nopar:Npn \__spath_reallength:n #1
290 {
291   \group_begin:
292   \int_set:Nn \l__spath_tmpa_int {0}
293   \tl_map_inline:nn {#1} {
294     \tl_set:Nn \l__spath_tmpa_tl {##1}
295     \tl_case:NnT \l__spath_tmpa_tl
296     {
297       \c_spath_lineto_tl {}
298       \c_spath_curveto_tl {}
299       \c_spath_closepath_tl {}
300     }
301     {
302       \int_incr:N \l__spath_tmpa_int

```



```

303     }
304   }
305   \int_gzero:N \g__spath_output_int
306   \int_gset_eq:NN \g__spath_output_int \l__spath_tmpa_int
307   \group_end:
308 }
309 \cs_new_protected_nopar:Npn \spath_reallength:Nn #1#2
310 {
311   \__spath_reallength:n {#2}
312   \int_set_eq:NN #1 \g__spath_output_int
313   \int_gzero:N \g__spath_output_int
314 }
315 \cs_generate_variant:Nn \spath_reallength:Nn {NV, cn, cV, Nv, cv}
316 \cs_new_protected_nopar:Npn \spath_greallength:Nn #1#2
317 {
318   \__spath_reallength:n {#2}
319   \int_gset_eq:NN #1 \g__spath_output_int
320   \int_gzero:N \g__spath_output_int
321 }
322 \cs_generate_variant:Nn \spath_greallength:Nn {NV, cn, cV}

```

(End definition for \spath_reallength:Nn and \spath_greallength:Nn.)

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

```

323 \cs_new_protected_nopar:Npn \__spath_numberofcomponents:n #1
324 {
325   \group_begin:
326   \int_set:Nn \l__spath_tmpa_int {0}
327   \tl_map_inline:nn {#1} {
328     \tl_set:Nn \l__spath_tmpa_tl {##1}
329     \tl_case:Nn \l__spath_tmpa_tl
330     {
331       \c_spath_moveto_tl
332       {
333         \int_incr:N \l__spath_tmpa_int
334       }
335     }
336   }
337   \int_gzero:N \g__spath_output_int
338   \int_gset_eq:NN \g__spath_output_int \l__spath_tmpa_int
339   \group_end:
340 }
341 \cs_new_protected_nopar:Npn \spath_numberofcomponents:Nn #1#2
342 {
343   \__spath_numberofcomponents:n {#2}
344   \int_set_eq:NN #1 \g__spath_output_int
345   \int_gzero:N \g__spath_output_int
346 }
347 \cs_generate_variant:Nn \spath_numberofcomponents:Nn {NV, cn, cV, Nv}
348 \cs_new_protected_nopar:Npn \spath_gnumberofcomponents:Nn #1#2
349 {
350   \__spath_numberofcomponents:n {#2}
351   \int_gset_eq:NN #1 \g__spath_output_int

```

```

352 \int_gzero:N \g__spath_output_int
353 }
354 \cs_generate_variant:Nn \spath_gnumberofcomponents:Nn {NV, cn, cV, Nv}

```

(End definition for \spath_numberofcomponents:Nn and \spath_gnumberofcomponents:Nn.)

\spath_initialpoint:Nn
\spath_ginitialpoint:Nn

The starting point of the path.

```

355 \cs_new_protected_nopar:Npn \__spath_initialpoint:n #1
356 {
357   \group_begin:
358   \tl_clear:N \l__spath_tmpa_tl
359   \tl_set:Nx \l__spath_tmpa_tl
360   {
361     { \tl_item:nn {#1} {2} }
362     { \tl_item:nn {#1} {3} }
363   }
364   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
365   \group_end:
366 }
367 \cs_new_protected_nopar:Npn \spath_initialpoint:Nn #1#2
368 {
369   \__spath_initialpoint:n {#2}
370   \tl_set_eq:NN #1 \g__spath_output_tl
371   \tl_gclear:N \g__spath_output_tl
372 }
373 \cs_generate_variant:Nn \spath_initialpoint:Nn {NV, cn, cV, Nv}
374 \cs_new_protected_nopar:Npn \spath_ginitialpoint:Nn #1#2
375 {
376   \__spath_initialpoint:n {#2}
377   \tl_gset_eq:NN #1 \g__spath_output_tl
378   \tl_gclear:N \g__spath_output_tl
379 }
380 \cs_generate_variant:Nn \spath_ginitialpoint:Nn {NV, cn, cV, Nv}

```

(End definition for \spath_initialpoint:Nn and \spath_ginitialpoint:Nn.)

\spath_finalpoint:Nn
\spath_gfinalpoint:Nn

The final point of the path.

```

381 \cs_new_protected_nopar:Npn \__spath_finalpoint:n #1
382 {
383   \group_begin:
384   \tl_set:Nn \l__spath_tmpa_tl {#1}
385   \tl_reverse:N \l__spath_tmpa_tl
386   \tl_clear:N \l__spath_tmpb_tl
387   \tl_set:Nx \l__spath_tmpb_tl
388   {
389     { \tl_item:Nn \l__spath_tmpa_tl {2} }
390     { \tl_item:Nn \l__spath_tmpa_tl {1} }
391   }
392   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
393   \group_end:
394 }
395 \cs_new_protected_nopar:Npn \spath_finalpoint:Nn #1#2
396 {
397   \__spath_finalpoint:n {#2}
398   \tl_set_eq:NN #1 \g__spath_output_tl

```

```

399 \tl_gclear:N \g__spath_output_tl
400 }
401 \cs_generate_variant:Nn \spath_finalpoint:Nn {NV, cn, cV, Nv}
402 \cs_new_protected_nopar:Npn \spath_gfinalpoint:Nn #1#2
403 {
404   \__spath_finalpoint:n {#2}
405   \tl_gset_eq:NN #1 \g__spath_output_tl
406   \tl_gclear:N \g__spath_output_tl
407 }
408 \cs_generate_variant:Nn \spath_gfinalpoint:Nn {NV, cn, cV, Nv}

```

(End definition for `\spath_finalpoint:Nn` and `\spath_gfinalpoint:Nn`.)

`\spath_finalmovepoint:Nn` Get the last move on the path.

```

\spath_gfinalmovepoint:Nn
409 \cs_new_protected_nopar:Npn \__spath_finalmovepoint:n #1
410 {
411   \group_begin:
412   \tl_set:Nn \l__spath_tmpc_tl { {0pt} {0pt} }
413   \tl_set:Nn \l__spath_tmpa_tl {#1}
414   \bool_do_until:nn
415   {
416     \tl_if_empty_p:N \l__spath_tmpa_tl
417   }
418   {
419     \tl_set:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
420     \tl_case:Nn \l__spath_tmpb_tl
421     {
422       \c_spath_moveto_tl
423       {
424         \tl_set:Nx \l__spath_tmpc_tl
425         {
426           { \tl_item:Nn \l__spath_tmpa_tl {2} }
427           { \tl_item:Nn \l__spath_tmpa_tl {3} }
428         }
429       }
430     }
431     \prg_replicate:nn {3}
432     {
433       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
434     }
435   }
436   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpc_tl
437   \group_end:
438 }
439 \cs_new_protected_nopar:Npn \spath_finalmovepoint:Nn #1#2
440 {
441   \__spath_finalmovepoint:n {#2}
442   \tl_set_eq:NN #1 \g__spath_output_tl
443   \tl_gclear:N \g__spath_output_tl
444 }
445 \cs_generate_variant:Nn \spath_finalmovepoint:Nn {NV, cn, cV}
446 \cs_new_protected_nopar:Npn \spath_gfinalmovepoint:Nn #1#2
447 {
448   \__spath_finalmovepoint:n {#2}

```

```

449 \tl_gset_eq:NN #1 \g__spath_output_tl
450 \tl_gclear:N \g__spath_output_tl
451 }
452 \cs_generate_variant:Nn \spath_gfinalmovepoint:Nn {NV, cn, cV}

```

(End definition for \spath_finalmovepoint:Nn and \spath_gfinalmovepoint:Nn.)

\spath_reverse:Nn
\spath_greverse:Nn

This computes the reverse of the path.

```

453 \cs_new_protected_nopar:Npn \__spath_reverse:n #1
454 {
455   \group_begin:
456   \tl_set:Nn \l__spath_tmpa_tl {#1}
457
458   \tl_clear:N \l__spath_tmpb_tl
459   \tl_clear:N \l__spath_tmpd_tl
460   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
461   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
462   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
463   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
464   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
465
466   \tl_put_left:Nx \l__spath_tmpd_tl
467   {
468     {\dim_use:N \l__spath_tmpa_dim}
469     {\dim_use:N \l__spath_tmpb_dim}
470   }
471
472   \bool_set_false:N \l__spath_closed_bool
473
474   \bool_until_do:nn {
475     \tl_if_empty_p:N \l__spath_tmpa_tl
476   }
477   {
478     \tl_set:Nx \l__spath_tmpe_tl {\tl_head:N \l__spath_tmpa_tl}
479
480     \tl_case:NnTF \l__spath_tmpe_tl
481     {
482       \c_spath_moveto_tl {
483
484         \bool_if:NT \l__spath_closed_bool
485         {
486           \tl_put_right:NV \l__spath_tmpd_tl \c_spath_closepath_tl
487           \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpd_tl}
488           \tl_put_right:Nx \l__spath_tmpd_tl
489           {
490             {\tl_head:N \l__spath_tmpd_tl }
491             {\tl_head:N \l__spath_tmpe_tl }
492           }
493         }
494         \bool_set_false:N \l__spath_closed_bool
495         \tl_put_left:NV \l__spath_tmpd_tl \c_spath_moveto_tl
496         \tl_put_left:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
497         \tl_clear:N \l__spath_tmpd_tl
498       }

```

```

499     \c_spath_lineto_tl {
500         \tl_put_left:NV \l__spath_tmpd_tl \c_spath_lineto_tl
501     }
502     \c_spath_curveto_tl {
503         \tl_put_left:NV \l__spath_tmpd_tl \c_spath_curvetoa_tl
504     }
505     \c_spath_curvetoa_tl {
506         \tl_put_left:NV \l__spath_tmpd_tl \c_spath_curveto_tl
507     }
508     \c_spath_curvetob_tl {
509         \tl_put_left:NV \l__spath_tmpd_tl \c_spath_curvetob_tl
510     }
511 }
512 {
513     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
514
515     \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
516     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
517     \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
518     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
519
520     \tl_put_left:Nx \l__spath_tmpd_tl
521     {
522         {\dim_use:N \l__spath_tmpa_dim}
523         {\dim_use:N \l__spath_tmpb_dim}
524     }
525
526 }
527 {
528     \tl_if_eq:NNTF \l__spath_tmpc_tl \c_spath_closepath_tl
529     {
530         \bool_set_true:N \l__spath_closed_bool
531     }
532     {
533         \msg_warning:nxx
534         { spath3 }
535         { unknown path construction }
536         {\l__spath_tmpc_tl }
537     }
538
539     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
540     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
541     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
542
543 }
544 }
545
546 \bool_if:NT \l__spath_closed_bool
547 {
548     \tl_put_right:NV \l__spath_tmpd_tl \c_spath_closepath_tl
549     \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpd_tl}
550     \tl_put_right:Nx \l__spath_tmpd_tl
551     {
552         { \tl_head:N \l__spath_tmpd_tl }

```

```

553     { \tl_head:N \l__spath_tmpe_tl }
554   }
555 }
556
557 \bool_set_false:N \l__spath_closed_bool
558 \tl_put_left:NV \l__spath_tmpd_tl \c_spath_moveto_tl
559 \tl_put_left:NV \l__spath_tmpe_tl \l__spath_tmpd_tl
560
561 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpe_tl
562 \group_end:
563 }
564 \cs_new_protected_nopar:Npn \spath_reverse:Nn #1#2
565 {
566   \__spath_reverse:n {#2}
567   \tl_set_eq:NN #1 \g__spath_output_tl
568   \tl_gclear:N \g__spath_output_tl
569 }
570 \cs_generate_variant:Nn \spath_reverse:Nn {NV, cn, cV, Nv}
571 \cs_new_protected_nopar:Npn \spath_reverse:N #1
572 {
573   \spath_reverse:NV #1#1
574 }
575 \cs_generate_variant:Nn \spath_reverse:N {c}
576 \cs_new_protected_nopar:Npn \spath_greverse:Nn #1#2
577 {
578   \__spath_reverse:n {#2}
579   \tl_gset_eq:NN #1 \g__spath_output_tl
580   \tl_gclear:N \g__spath_output_tl
581 }
582 \cs_generate_variant:Nn \spath_greverse:Nn {NV, cn, cV, Nv}
583 \cs_new_protected_nopar:Npn \spath_greverse:N #1
584 {
585   \spath_greverse:NV #1#1
586 }
587 \cs_generate_variant:Nn \spath_greverse:N {c}

```

(End definition for \spath_reverse:Nn and \spath_greverse:Nn.)

\spath_initialaction:Nn This is the first thing that the path does (after the initial move).

```

\spath_ginitialaction:Nn
588 \cs_new_protected_nopar:Npn \__spath_initialaction:n #1
589 {
590   \group_begin:
591   \tl_clear:N \l__spath_tmpe_tl
592   \int_compare:nT
593   {
594     \tl_count:n {#1} > 3
595   }
596   {
597     \tl_set:Nx \l__spath_tmpe_tl
598     {
599       \tl_item:Nn {#1} {4}
600     }
601   }
602   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpe_tl

```

```

603 \group_end:
604 }
605 \cs_new_protected_nopar:Npn \spath_initialaction:Nn #1#2
606 {
607   \__spath_initialaction:n {#2}
608   \tl_set_eq:NN #1 \g__spath_output_tl
609   \tl_gclear:N \g__spath_output_tl
610 }
611 \cs_generate_variant:Nn \spath_initialaction:Nn {NV}
612 \cs_new_protected_nopar:Npn \spath_ginitialaction:Nn #1#2
613 {
614   \__spath_initialaction:n {#2}
615   \tl_gset_eq:NN #1 \g__spath_output_tl
616   \tl_gclear:N \g__spath_output_tl
617 }
618 \cs_generate_variant:Nn \spath_ginitialaction:Nn {NV}

(End definition for \spath_initialaction:Nn and \spath_ginitialaction:Nn.)

```

\spath_finalaction:Nn
\spath_gfinalaction:Nn

This is the last thing that the path does.

```

619 \cs_new_protected_nopar:Npn \__spath_finalaction:n #1
620 {
621   \group_begin:
622   \tl_clear:N \l__spath_tmpb_tl
623   \int_compare:nT
624   {
625     \tl_count:n {#1} > 3
626   }
627   {
628     \tl_set:Nn \l__spath_tmpa_tl {#1}
629     \tl_reverse:N \l__spath_tmpa_tl
630     \tl_set:Nx \l__spath_tmpb_tl
631     {
632       \tl_item:Nn \l__spath_tmpa_tl {3}
633     }
634     \tl_if_eq:NNT \l__spath_tmpb_tl \c_spath_curvetoa_tl
635     {
636       \tl_set_eq:NN \l__spath_tmpb_tl \c_spath_curveto_tl
637     }
638   }
639   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
640   \group_end:
641 }
642 \cs_new_protected_nopar:Npn \spath_finalaction:Nn #1#2
643 {
644   \__spath_finalaction:n {#2}
645   \tl_set_eq:NN #1 \g__spath_output_tl
646   \tl_gclear:N \g__spath_output_tl
647 }
648 \cs_generate_variant:Nn \spath_finalaction:Nn {NV}
649 \cs_new_protected_nopar:Npn \spath_gfinalaction:Nn #1#2
650 {
651   \__spath_finalaction:n {#2}
652   \tl_gset_eq:NN #1 \g__spath_output_tl

```

```

653 \tl_gclear:N \g__spath_output_tl
654 }
655 \cs_generate_variant:Nn \spath_gfinalaction:Nn {NV}

(End definition for \spath_finalaction:Nn and \spath_gfinalaction:Nn.)

```

```

\spath_minbb:Nn This computes the minimum (bottom left) of the bounding box of the path.
\spath_gminbb:Nn
656 \cs_new_protected_nopar:Npn \__spath_minbb:n #1
657 {
658   \group_begin:
659   \tl_set:Nn \l__spath_tmpa_tl {#1}
660   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
661   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
662   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
663   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
664   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
665   \bool_until_do:nn {
666     \tl_if_empty_p:N \l__spath_tmpa_tl
667   }
668   {
669     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
670     \dim_set:Nn \l__spath_tmpa_dim
671     {\dim_min:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tmpa_dim}}
672     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
673     \dim_set:Nn \l__spath_tmpb_dim
674     {\dim_min:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tmpb_dim}}
675     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
676   }
677   \tl_clear:N \l__spath_tmpb_tl
678   \tl_put_right:Nx \l__spath_tmpb_tl
679   {
680     {\dim_use:N \l__spath_tmpa_dim}
681     {\dim_use:N \l__spath_tmpb_dim}
682   }
683   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
684   \group_end:
685 }
686 \cs_new_protected_nopar:Npn \spath_minbb:Nn #1#2
687 {
688   \__spath_minbb:n {#2}
689   \tl_set_eq:NN #1 \g__spath_output_tl
690   \tl_gclear:N \g__spath_output_tl
691 }
692 \cs_generate_variant:Nn \spath_minbb:Nn {NV, cn, cV}
693 \cs_new_protected_nopar:Npn \spath_gminbb:Nn #1#2
694 {
695   \__spath_minbb:n {#2}
696   \tl_gset_eq:NN #1 \g__spath_output_tl
697   \tl_gclear:N \g__spath_output_tl
698 }
699 \cs_generate_variant:Nn \spath_gminbb:Nn {NV, cn, cV}

(End definition for \spath_minbb:Nn and \spath_gminbb:Nn.)

```


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

```

\spath_gmaxbb:Nn
700 \cs_new_protected_nopar:Npn \__spath_maxbb:n #1
701 {
702   \group_begin:
703   \tl_set:Nn \l__spath_tmpa_tl {#1}
704   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
705   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
706   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
707   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
708   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
709   \bool_until_do:nn {
710     \tl_if_empty_p:N \l__spath_tmpa_tl
711   }
712   {
713     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
714     \dim_set:Nn \l__spath_tmpa_dim
715     {\dim_max:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tmpa_dim}}
716     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
717     \dim_set:Nn \l__spath_tmpb_dim
718     {\dim_max:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tmpb_dim}}
719     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
720   }
721   \tl_clear:N \l__spath_tmpb_tl
722   \tl_put_right:Nx \l__spath_tmpb_tl
723   {
724     {\dim_use:N \l__spath_tmpa_dim}
725     {\dim_use:N \l__spath_tmpb_dim}
726   }
727   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
728   \group_end:
729 }
730 \cs_new_protected_nopar:Npn \spath_maxbb:Nn #1#2
731 {
732   \__spath_maxbb:n {#2}
733   \tl_set_eq:NN #1 \g__spath_output_tl
734   \tl_gclear:N \g__spath_output_tl
735 }
736 \cs_generate_variant:Nn \spath_maxbb:Nn {NV, cn, cV}
737 \cs_new_protected_nopar:Npn \spath_gmaxbb:Nn #1#2
738 {
739   \__spath_maxbb:n {#2}
740   \tl_gset_eq:NN #1 \g__spath_output_tl
741   \tl_gclear:N \g__spath_output_tl
742 }
743 \cs_generate_variant:Nn \spath_gmaxbb:Nn {NV, cn, cV}

```

(End definition for `\spath_maxbb:Nn` and `\spath_gmaxbb:Nn`.)

`\spath_save_to_aux:Nn` This saves a soft path to the auxfile. The first argument is the macro that will be assigned
`\spath_save_to_aux:N` to the soft path when the aux file is read back in.

```

744 \int_set:Nn \l__spath_tmpa_int {\char_value_catcode:n {'@}}
745 \char_set_catcode_letter:N @
746 \cs_new_protected_nopar:Npn \spath_save_to_aux:Nn #1#2 {
747   \tl_if_empty:nF {#2}

```

```

748 {
749   \tl_clear:N \l__spath_tmpa_tl
750   \tl_put_right:Nn \l__spath_tmpa_tl {
751     \ExplSyntaxOn
752     \tl_clear:N #1
753     \tl_set:Nn #1 {#2}
754     \ExplSyntaxOff
755   }
756   \protected@write\@auxout{}\{
757     \tl_to_str:N \l__spath_tmpa_tl
758   }
759 }
760 }
761 \char_set_catcode:nn {'@} {\l__spath_tmpa_int}
762 \cs_generate_variant:Nn \spath_save_to_aux:Nn {cn, cV, NV}
763 \cs_new_protected_nopar:Npn \spath_save_to_aux:N #1
764 {
765   \tl_if_exist:NT #1
766   {
767     \spath_save_to_aux:NV #1#1
768   }
769 }

```

(End definition for `\spath_save_to_aux:Nn` and `\spath_save_to_aux:N`.)

3.3 Path Manipulation

These functions all manipulate a soft path. They come with a variety of different argument specifications. As a general rule, the first argument is the macro in which to store the modified path, the second is the path to manipulate, and the rest are the information about what to do. There is always a variant in which the path is specified by a macro and restored back in that same macro.

<code>\spath_translate:Nnnn</code> <code>\spath_translate:Nnn</code> <code>\spath_gtranslate:Nnnn</code> <code>\spath_gtranslate:Nnn</code>	Translates a path by an amount. <pre> 770 \cs_new_protected_nopar:Npn __spath_translate:nnn #1#2#3 771 { 772 \group_begin: 773 \tl_set:Nn \l__spath_tmpa_tl {#1} 774 \tl_clear:N \l__spath_tmpb_tl 775 \bool_until_do:nn { 776 \tl_if_empty_p:N \l__spath_tmpa_tl 777 } 778 { 779 \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl} 780 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl} 781 782 \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl + #2} 783 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl} 784 785 \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl + #3} 786 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl} 787 788 \tl_put_right:Nx \l__spath_tmpb_tl 789 { </pre>
--	---

```

790     {\dim_use:N \l__spath_tmpa_dim}
791     {\dim_use:N \l__spath_tmpb_dim}
792   }
793 }
794 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
795 \group_end:
796 }
797 \cs_generate_variant:Nn \__spath_translate:nnn {nVV}
798 \cs_new_protected_nopar:Npn \spath_translate:Nnnn #1#2#3#4
799 {
800   \__spath_translate:nnn {#2}{#3}{#4}
801   \tl_set_eq:NN #1 \g__spath_output_tl
802   \tl_gclear:N \g__spath_output_tl
803 }
804 \cs_generate_variant:Nn \spath_translate:Nnnn {NVxx, NVVV, NVnn}
805 \cs_new_protected_nopar:Npn \spath_translate:Nnn #1#2#3
806 {
807   \spath_translate:NVnn #1#1{#2}{#3}
808 }
809 \cs_generate_variant:Nn \spath_translate:Nnn {NVV, cnn, cVV}
810 \cs_new_protected_nopar:Npn \spath_gtranslate:Nnnn #1#2#3#4
811 {
812   \__spath_translate:nnn {#2}{#3}{#4}
813   \tl_gset_eq:NN #1 \g__spath_output_tl
814   \tl_gclear:N \g__spath_output_tl
815 }
816 \cs_generate_variant:Nn \spath_gtranslate:Nnnn {NVxx, NVVV, NVnn}
817 \cs_new_protected_nopar:Npn \spath_gtranslate:Nnn #1#2#3
818 {
819   \spath_gtranslate:NVnn #1#1{#2}{#3}
820 }
821 \cs_generate_variant:Nn \spath_gtranslate:Nnn {NVV, cnn, cVV}

```

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

```

822 \cs_new_protected_nopar:Npn \spath_translate:Nn #1#2
823 {
824   \spath_translate:Nnn #1 #2
825 }
826 \cs_generate_variant:Nn \spath_translate:Nn {NV}
827 \cs_new_protected_nopar:Npn \spath_gtranslate:Nn #1#2
828 {
829   \spath_gtranslate:Nnn #1 #2
830 }
831 \cs_generate_variant:Nn \spath_gtranslate:Nn {NV}

```

(End definition for `\spath_translate:Nnnn` and others.)

<pre> \spath_translate_to:Nnnn \spath_translate_to:Nnn \spath_gtranslate_to:Nnnn \spath_gtranslate_to:Nnn </pre>	<p>Translates a path so that it starts at a point.</p> <pre> 832 \cs_new_protected_nopar:Npn __spath_translate_to:nnn #1#2#3 833 { 834 \group_begin: 835 \spath_initialpoint:Nn \l__spath_tmpa_tl {#1} 836 837 \dim_set:Nn \l__spath_tmpa_dim </pre>
--	---

```

838 {
839   #2
840   -
841   \tl_item:Nn \l__spath_tmpa_tl {1}
842 }
843 \dim_set:Nn \l__spath_tmpb_dim
844 {
845   #3
846   -
847   \tl_item:Nn \l__spath_tmpa_tl {2}
848 }
849
850 \__spath_translate:nVV {#1} \l__spath_tmpa_dim \l__spath_tmpb_dim
851 \group_end:
852 }
853 \cs_new_protected_nopar:Npn \spath_translate_to:Nnnn #1#2#3#4
854 {
855   \__spath_translate_to:nnn {#2}{#3}{#4}
856   \tl_set_eq:NN #1 \g__spath_output_tl
857   \tl_gclear:N \g__spath_output_tl
858 }
859 \cs_generate_variant:Nn \spath_translate_to:Nnnn {NVxx, NVVV, NVnn}
860 \cs_new_protected_nopar:Npn \spath_translate_to:Nnn #1#2#3
861 {
862   \spath_translate_to:NVnn #1#1{#2}{#3}
863 }
864 \cs_generate_variant:Nn \spath_translate_to:Nnn {NVV, cnn, cVV}
865 \cs_new_protected_nopar:Npn \spath_gtranslate_to:Nnnn #1#2#3#4
866 {
867   \__spath_translate_to:nnn {#2}{#3}{#4}
868   \tl_gset_eq:NN #1 \g__spath_output_tl
869   \tl_gclear:N \g__spath_output_tl
870 }
871 \cs_generate_variant:Nn \spath_gtranslate_to:Nnnn {NVxx, NVVV, NVnn}
872 \cs_new_protected_nopar:Npn \spath_gtranslate_to:Nnn #1#2#3
873 {
874   \spath_gtranslate_to:NVnn #1#1{#2}{#3}
875 }
876 \cs_generate_variant:Nn \spath_gtranslate_to:Nnn {NVV, cnn, cVV}

```

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

```

877 \cs_new_protected_nopar:Npn \spath_translate_to:Nn #1#2
878 {
879   \spath_translate_to:Nnn #1 #2
880 }
881 \cs_generate_variant:Nn \spath_translate_to:Nn {NV}
882 \cs_new_protected_nopar:Npn \spath_gtranslate_to:Nn #1#2
883 {
884   \spath_gtranslate_to:Nnn #1 #2
885 }
886 \cs_generate_variant:Nn \spath_gtranslate_to:Nn {NV}

```

(End definition for `\spath_translate_to:Nnnn` and others.)

```

\spath_scale:Nnnn Scale a path.
\spath_scale:Nnn 887 \cs_new_protected_nopar:Npn \__spath_scale:nnn #1#2#3
\spath_gscale:Nnnn 888 {
\spath_gscale:Nnn 889 \group_begin:
890 \tl_set:Nn \l__spath_tmpa_tl {#1}
891 \tl_clear:N \l__spath_tmpb_tl
892 \bool_until_do:nn {
893 \tl_if_empty_p:N \l__spath_tmpa_tl
894 }
895 {
896 \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
897 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
898
899 \fp_set:Nn \l__spath_tmpa_fp {\tl_head:N \l__spath_tmpa_tl * #2}
900 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
901
902 \fp_set:Nn \l__spath_tmpb_fp {\tl_head:N \l__spath_tmpa_tl * #3}
903 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
904
905 \tl_put_right:Nx \l__spath_tmpb_tl
906 {
907 {\fp_to_dim:N \l__spath_tmpa_fp}
908 {\fp_to_dim:N \l__spath_tmpb_fp}
909 }
910 }
911 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
912 \group_end:
913 }
914 \cs_new_protected_nopar:Npn \spath_scale:Nnnn #1#2#3#4
915 {
916 \__spath_scale:nnn {#2}{#3}{#4}
917 \tl_set_eq:NN #1 \g__spath_output_tl
918 \tl_gclear:N \g__spath_output_tl
919 }
920 \cs_generate_variant:Nn \spath_scale:Nnnn {NVnn, Nnxx}
921 \cs_new_protected_nopar:Npn \spath_scale:Nnn #1#2#3
922 {
923 \spath_scale:NVnn #1#1{#2}{#3}
924 }
925 \cs_generate_variant:Nn \spath_scale:Nnn {cnn, cVV, NVV}
926 \cs_new_protected_nopar:Npn \spath_gscale:Nnnn #1#2#3#4
927 {
928 \__spath_scale:nnn {#2}{#3}{#4}
929 \tl_gset_eq:NN #1 \g__spath_output_tl
930 \tl_gclear:N \g__spath_output_tl
931 }
932 \cs_generate_variant:Nn \spath_gscale:Nnnn {NVnn, Nnxx}
933 \cs_new_protected_nopar:Npn \spath_gscale:Nnn #1#2#3
934 {
935 \spath_gscale:NVnn #1#1{#2}{#3}
936 }
937 \cs_generate_variant:Nn \spath_gscale:Nnn {cnn, cVV, NVV}

```

This variant allows for passing the coordinates as a single braced group as it strips

off the outer braces of the second argument.

```

938 \cs_new_protected_nopar:Npn \spath_scale:Nn #1#2
939 {
940   \spath_scale:Nnn #1 #2
941 }
942
943 \cs_generate_variant:Nn \spath_scale:Nn {NV}
944 \cs_new_protected_nopar:Npn \spath_gscale:Nn #1#2
945 {
946   \spath_gscale:Nnn #1 #2
947 }
948
949 \cs_generate_variant:Nn \spath_gscale:Nn {NV}

```

(End definition for `\spath_scale:Nnnn` and others.)

```

\spath_transform:Nnnnnnnn
\spath_transform:Nnnnnnnn
\spath_gtransform:Nnnnnnnn
\spath_gtransform:Nnnnnnnn

```

Applies an affine (matrix and vector) transformation to path. The matrix is specified in rows first.

```

950 \cs_new_protected_nopar:Npn \__spath_transform:nnnnnnn #1#2#3#4#5#6#7
951 {
952   \group_begin:
953   \tl_set:Nn \l__spath_tmpa_tl {#1}
954   \tl_clear:N \l__spath_tmpb_tl
955   \bool_until_do:nn {
956     \tl_if_empty_p:N \l__spath_tmpa_tl
957   }
958   {
959     \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
960     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
961     \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
962     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
963     \tl_set:Nx \l__spath_tmpd_tl {\tl_head:N \l__spath_tmpa_tl}
964     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
965
966     \fp_set:Nn \l__spath_tmpa_fp
967     {\l__spath_tmpc_tl * #2 + \l__spath_tmpd_tl * #4 + #6}
968     \fp_set:Nn \l__spath_tmpb_fp
969     {\l__spath_tmpc_tl * #3 + \l__spath_tmpd_tl * #5 + #7}
970     \tl_put_right:Nx \l__spath_tmpb_tl
971     {
972       {\fp_to_dim:N \l__spath_tmpa_fp}
973       {\fp_to_dim:N \l__spath_tmpb_fp}
974     }
975   }
976
977   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
978   \group_end:
979 }
980 \cs_new_protected_nopar:Npn \spath_transform:Nnnnnnnn #1#2#3#4#5#6#7#8
981 {
982   \__spath_transform:nnnnnnn {#2}{#3}{#4}{#5}{#6}{#7}{#8}
983   \tl_set_eq:NN #1 \g__spath_output_tl
984   \tl_gclear:N \g__spath_output_tl
985 }

```

```

986 \cs_generate_variant:Nn \spath_transform:Nnnnnnnn
987 {NVnnnnnn, Nxxxxxxx, cnnnnnnn}
988 \cs_new_protected_nopar:Npn \spath_transform:Nnnnnnn #1#2#3#4#5#6#7
989 {
990   \spath_transform:NVnnnnnn #1#1{#2}{#3}{#4}{#5}{#6}{#7}
991 }
992 \cs_generate_variant:Nn \spath_transform:Nnnnnnn {cnnnnnn}
993 \cs_new_protected_nopar:Npn \spath_transform:Nnn #1#2#3
994 {
995   \spath_transform:Nnnnnnn #1{#2}#3
996 }
997 \cs_generate_variant:Nn \spath_transform:Nnn {cnn, cVn, NVn, NnV}
998 \cs_new_protected_nopar:Npn \spath_transform:Nn #1#2
999 {
1000   \spath_transform:NVnnnnnn #1#1#2
1001 }
1002 \cs_generate_variant:Nn \spath_transform:Nn {cn, cV, NV}
1003
1004 \cs_new_protected_nopar:Npn \spath_gtransform:Nnnnnnnn #1#2#3#4#5#6#7#8
1005 {
1006   \__spath_transform:nnnnnn {#2}{#3}{#4}{#5}{#6}{#7}{#8}
1007   \tl_gset_eq:NN #1 \g__spath_output_tl
1008   \tl_gclear:N \g__spath_output_tl
1009 }
1010 \cs_generate_variant:Nn \spath_gtransform:Nnnnnnnn {NVnnnnnn, Nxxxxxxx, cnnnnnnn}
1011 \cs_new_protected_nopar:Npn \spath_gtransform:Nnnnnnn #1#2#3#4#5#6#7
1012 {
1013   \spath_gtransform:NVnnnnnn #1#1{#2}{#3}{#4}{#5}{#6}{#7}
1014 }
1015 \cs_generate_variant:Nn \spath_gtransform:Nnnnnnn {cnnnnnn}
1016 \cs_new_protected_nopar:Npn \spath_gtransform:Nnn #1#2#3
1017 {
1018   \spath_gtransform:Nnnnnnn #1{#2}#3
1019 }
1020 \cs_generate_variant:Nn \spath_gtransform:Nnn {cnn, cVn, NVn, NnV}
1021 \cs_new_protected_nopar:Npn \spath_gtransform:Nn #1#2
1022 {
1023   \spath_gtransform:NVnnnnnn #1#1#2
1024 }
1025 \cs_generate_variant:Nn \spath_gtransform:Nn {cn, cV, NV}

```

(End definition for `\spath_transform:Nnnnnnnn` and others.)

```

\spath_span:Nnnn The span functions transform a path to start and end at specified points. The normalise
\spath_span:Nnn functions transform it to start at the origin and end at (1pt,0pt).
\spath_gspan:Nnnn
\spath_gspan:Nnn If the path starts and ends at the same point then it is translated to the specified
\spath_gspan:Nnn point (or origin) but not otherwise changed.
\spath_normalise:Nn
\spath_normalise:N 1026 \cs_new_protected_nopar:Npn \__spath_span:nnn #1#2#3
\spath_gnormalise:Nn 1027 {
\spath_gnormalise:N 1028   \group_begin:
1029   \spath_initialpoint:Nn \l__spath_tmpa_tl {#1}
1030   \spath_finalpoint:Nn \l__spath_tmpb_tl {#1}
1031
1032   \fp_set:Nn \l__spath_tmpa_fp

```

```

1033 {
1034     (\tl_item:Nn \l__spath_tmpb_tl {1}) -
1035     (\tl_item:Nn \l__spath_tmpa_tl {1})
1036 }
1037 \fp_set:Nn \l__spath_tmpb_fp
1038 {
1039     (\tl_item:Nn \l__spath_tmpb_tl {2}) -
1040     (\tl_item:Nn \l__spath_tmpa_tl {2})
1041 }
1042 \fp_set:Nn \l__spath_tmpc_fp
1043 {
1044     (\l__spath_tmpa_fp) * (\l__spath_tmpa_fp)
1045     +
1046     (\l__spath_tmpb_fp * \l__spath_tmpb_fp)
1047 }
1048
1049 \fp_compare:nTF
1050 {
1051     \l__spath_tmpc_fp < 0.001
1052 }
1053 {
1054     \spath_translate_to:Nnnn \l__spath_tmpd_tl {#1} #2
1055 }
1056 {
1057     \fp_set:Nn \l__spath_tmpa_fp
1058     {
1059         (
1060             ((\tl_item:nn {#3} {1})
1061             -
1062             (\tl_item:nn {#2} {1}))
1063         *
1064         ((\tl_item:Nn \l__spath_tmpb_tl {1})
1065         -
1066         (\tl_item:Nn \l__spath_tmpa_tl {1}))
1067         +
1068         ((\tl_item:nn {#3} {2})
1069         -
1070         (\tl_item:nn {#2} {2}))
1071         *
1072         ((\tl_item:Nn \l__spath_tmpb_tl {2})
1073         -
1074         (\tl_item:Nn \l__spath_tmpa_tl {2}))
1075         )
1076         /
1077         \l__spath_tmpc_fp
1078     }
1079     \fp_set:Nn \l__spath_tmpb_fp
1080     {
1081         (
1082             ((\tl_item:nn {#3} {2})
1083             -
1084             (\tl_item:nn {#2} {2}))
1085         *
1086         ((\tl_item:Nn \l__spath_tmpb_tl {1})

```



```

1087 -
1088 (\tl_item:Nn \l__spath_tmpa_tl {1}))
1089 -
1090 ((\tl_item:nn {#3} {1})
1091 -
1092 (\tl_item:nn {#2} {1}))
1093 *
1094 ((\tl_item:Nn \l__spath_tmpb_tl {2})
1095 -
1096 (\tl_item:Nn \l__spath_tmpa_tl {2}))
1097 )
1098 /
1099 \l__spath_tmpc_fp
1100 }
1101
1102 \tl_set:Nx \l__spath_tmpc_tl
1103 {
1104 {
1105 \fp_to_decimal:N \l__spath_tmpa_fp
1106 }
1107 {
1108 \fp_to_decimal:N \l__spath_tmpb_fp
1109 }
1110 {
1111 \fp_eval:n { - \l__spath_tmpb_fp }
1112 }
1113 {
1114 \fp_to_decimal:N \l__spath_tmpa_fp
1115 }
1116 {
1117 \fp_to_dim:n
1118 {
1119 \tl_item:nn {#2} {1}
1120 -
1121 \l__spath_tmpa_fp * (\tl_item:Nn \l__spath_tmpa_tl {1})
1122 +
1123 \l__spath_tmpb_fp * (\tl_item:Nn \l__spath_tmpa_tl {2})
1124 }
1125 }
1126 {
1127 \fp_to_dim:n
1128 {
1129 \tl_item:nn {#2} {2}
1130 -
1131 \l__spath_tmpb_fp * (\tl_item:Nn \l__spath_tmpa_tl {1})
1132 -
1133 \l__spath_tmpa_fp * (\tl_item:Nn \l__spath_tmpa_tl {2})
1134 }
1135 }
1136 }
1137 \spath_transform:NnV \l__spath_tmpd_tl {#1} \l__spath_tmpc_tl
1138 }
1139 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpd_tl
1140 \group_end:

```

```

1141 }
1142 \cs_new_protected_nopar:Npn \spath_span:Nnnn #1#2#3#4
1143 {
1144   \__spath_span:nnn {#2}{#3}{#4}
1145   \tl_set_eq:NN #1 \g__spath_output_tl
1146   \tl_gclear:N \g__spath_output_tl
1147 }
1148 \cs_generate_variant:Nn \spath_span:Nnnn {NVnn, NVVV, NnVV}
1149 \cs_new_protected_nopar:Npn \spath_span:Nnn #1#2#3
1150 {
1151   \spath_span:NVnn #1#1{#2}{#3}
1152 }
1153 \cs_generate_variant:Nn \spath_span:Nnn {NVV, cnn, cvv, cVV}
1154 \cs_new_protected_nopar:Npn \spath_gspan:Nnnn #1#2#3#4
1155 {
1156   \__spath_span:nnn {#2}{#3}{#4}
1157   \tl_gset_eq:NN #1 \g__spath_output_tl
1158   \tl_gclear:N \g__spath_output_tl
1159 }
1160 \cs_generate_variant:Nn \spath_gspan:Nnnn {NVnn, NVVV}
1161 \cs_new_protected_nopar:Npn \spath_gspan:Nnn #1#2#3
1162 {
1163   \spath_gspan:NVnn #1#1{#2}{#3}
1164 }
1165 \cs_generate_variant:Nn \spath_gspan:Nnn {NVV, cnn, cvv, cVV}
1166 \cs_new_protected_nopar:Npn \__spath_normalise:n #1
1167 {
1168   \__spath_span:nnn {#1}{0pt}{0pt}}{1pt}{0pt}}
1169 }
1170 \cs_new_protected_nopar:Npn \spath_normalise:Nn #1#2
1171 {
1172   \__spath_normalise:n {#2}
1173   \tl_set_eq:NN #1 \g__spath_output_tl
1174   \tl_gclear:N \g__spath_output_tl
1175 }
1176 \cs_generate_variant:Nn \spath_normalise:Nn {cn,NV, cV, cv}
1177 \cs_new_protected_nopar:Npn \spath_normalise:N #1
1178 {
1179   \spath_normalise:NV #1#1
1180 }
1181 \cs_generate_variant:Nn \spath_normalise:N {c}
1182 \cs_new_protected_nopar:Npn \spath_gnormalise:Nn #1#2
1183 {
1184   \__spath_normalise:n {#2}
1185   \tl_gset_eq:NN #1 \g__spath_output_tl
1186   \tl_gclear:N \g__spath_output_tl
1187 }
1188 \cs_generate_variant:Nn \spath_gnormalise:Nn {cn,NV, cV, cv}
1189 \cs_new_protected_nopar:Npn \spath_gnormalise:N #1
1190 {
1191   \spath_gnormalise:NV #1#1
1192 }
1193 \cs_generate_variant:Nn \spath_gnormalise:N {c}

```

(End definition for \spath_span:Nnnn and others.)

`\spath_splice_between:Nnnn`
`\spath_splice_between:Nnn`
`\spath_gsplice_between:Nnnn`
`\spath_gsplice_between:Nnn`

This takes three paths and returns a single path in which the middle one is adjusted (and welded) so that it joins the first path to the third.

```

1194 \cs_new_protected_nopar:Npn \__spath_splice_between:nnn #1#2#3
1195 {
1196   \group_begin:
1197   \spath_finalpoint:NV \l__spath_tmpd_tl {#1}
1198   \spath_initialpoint:NV \l__spath_tmpe_tl {#3}
1199   \spath_span:NnVV \l__spath_tmpe_tl {#2} \l__spath_tmpd_tl \l__spath_tmpe_tl
1200   \spath_append_no_move:NnV \l__spath_tmpe_tl {#1} \l__spath_tmpe_tl
1201   \spath_append_no_move:Nn \l__spath_tmpe_tl {#3}
1202   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpe_tl
1203   \group_end:
1204 }
1205 \cs_new_protected_nopar:Npn \spath_splice_between:Nnnn #1#2#3#4
1206 {
1207   \__spath_splice_between:nnn {#2}{#3}{#4}
1208   \tl_set_eq:NN #1 \g__spath_output_tl
1209   \tl_gclear:N \g__spath_output_tl
1210 }
1211 \cs_generate_variant:Nn \spath_splice_between:Nnnn {NVnn, NVVV}
1212 \cs_new_protected_nopar:Npn \spath_splice_between:Nnn #1#2#3
1213 {
1214   \spath_splice_between:NVnn #1#1{#2}{#3}
1215 }
1216 \cs_generate_variant:Nn \spath_splice_between:Nnn {NVV, cnn, cvv, Nvn}
1217 \cs_new_protected_nopar:Npn \spath_gsplice_between:Nnnn #1#2#3#4
1218 {
1219   \__spath_splice_between:nnn {#2}{#3}{#4}
1220   \tl_gset_eq:NN #1 \g__spath_output_tl
1221   \tl_gclear:N \g__spath_output_tl
1222 }
1223 \cs_generate_variant:Nn \spath_gsplice_between:Nnnn {NVnn, NVVV}
1224 \cs_new_protected_nopar:Npn \spath_gsplice_between:Nnn #1#2#3
1225 {
1226   \spath_gsplice_between:NVnn #1#1{#2}{#3}
1227 }
1228 \cs_generate_variant:Nn \spath_gsplice_between:Nnn {NVV, cnn, cvv}

```

(End definition for `\spath_splice_between:Nnnn` and others.)

`\spath_close_with:Nn`
`\spath_gclose_with:Nn`

Closes the first path by splicing in the second.

```

1229 \cs_new_protected_nopar:Npn \__spath_close_with:nn #1#2
1230 {
1231   \group_begin:
1232   \spath_finalmovepoint:Nn \l__spath_tmpe_tl {#1}
1233   \spath_finalpoint:Nn \l__spath_tmpe_tl {#1}
1234   \dim_compare:nTF
1235   {
1236     \dim_abs:n
1237     {
1238       \tl_item:Nn \l__spath_tmpe_tl {1}
1239       -
1240       \tl_item:Nn \l__spath_tmpe_tl {1}
1241     }

```

```

1242     +
1243     \dim_abs:n
1244     {
1245       \tl_item:Nn \l__spath_tmpa_tl {2}
1246       -
1247       \tl_item:Nn \l__spath_tmpb_tl {2}
1248     }
1249     < 0.01pt
1250   }
1251   {
1252     \__spath_close:n {#1}
1253   }
1254   {
1255     \spath_span:NnVV \l__spath_tmpc_tl {#2} \l__spath_tmpb_tl \l__spath_tmpa_tl
1256     \spath_append_no_move:NnV \l__spath_tmpd_tl {#1} \l__spath_tmpc_tl
1257     \__spath_close:V \l__spath_tmpd_tl
1258   }
1259   \group_end:
1260 }
1261 \cs_new_protected_nopar:Npn \spath_close_with:Nnn #1#2#3
1262 {
1263   \__spath_close_with:nn {#2}{#3}
1264   \tl_set_eq:NN #1 \g__spath_output_tl
1265   \tl_gclear:N \g__spath_output_tl
1266 }
1267 \cs_generate_variant:Nn \spath_close_with:Nnn {cnn, cVV, cvv, NVn}
1268 \cs_new_protected_nopar:Npn \spath_close_with:Nn #1#2
1269 {
1270   \spath_close_with:NVn #1#1{#2}
1271 }
1272 \cs_generate_variant:Nn \spath_close_with:Nn {cn, cV, cv, NV}
1273 \cs_new_protected_nopar:Npn \spath_gclose_with:Nnn #1#2#3
1274 {
1275   \__spath_close_with:nn {#2}{#3}
1276   \tl_gset_eq:NN #1 \g__spath_output_tl
1277   \tl_gclear:N \g__spath_output_tl
1278 }
1279 \cs_generate_variant:Nn \spath_gclose_with:Nnn {cnn, cVV, cvv, NVn}
1280 \cs_new_protected_nopar:Npn \spath_gclose_with:Nn #1#2
1281 {
1282   \spath_gclose_with:NVn #1#1{#2}
1283 }
1284 \cs_generate_variant:Nn \spath_gclose_with:Nn {cn, cV, cv, NV}

```

(End definition for `\spath_close_with:Nn` and `\spath_gclose_with:Nn`.)

`\spath_weld:Nnn` 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.

`\spath_weld:Nn`
`\spath_gweld:Nnn`
`\spath_gweld:Nn`

```

1285 \cs_new_protected_nopar:Npn \__spath_weld:nn #1#2
1286 {
1287   \group_begin:
1288   \tl_set:Nn \l__spath_tmpa_tl {#1}
1289   \tl_set:Nn \l__spath_tmpb_tl {#2}

```

```

1290 \spath_finalpoint:Nn \l__spath_tmpc_tl {#1}
1291 \spath_translate_to:NV \l__spath_tmpb_tl \l__spath_tmpc_tl
1292
1293 \__spath_append_no_move:VV \l__spath_tmpa_tl \l__spath_tmpb_tl
1294 \group_end:
1295 }
1296 \cs_new_protected_nopar:Npn \spath_weld:Nnn #1#2#3
1297 {
1298   \__spath_weld:nn {#2}{#3}
1299   \tl_set_eq:NN #1 \g__spath_output_tl
1300   \tl_gclear:N \g__spath_output_tl
1301 }
1302 \cs_generate_variant:Nn \spath_weld:Nnn {NVV,NVn}
1303 \cs_new_protected_nopar:Npn \spath_weld:Nn #1#2
1304 {
1305   \spath_weld:NVn #1#1{#2}
1306 }
1307 \cs_generate_variant:Nn \spath_weld:Nn {NV, Nv, cV, cv}
1308 \cs_new_protected_nopar:Npn \spath_gweld:Nnn #1#2#3
1309 {
1310   \__spath_weld:nn {#2}{#3}
1311   \tl_gset_eq:NN #1 \g__spath_output_tl
1312   \tl_gclear:N \g__spath_output_tl
1313 }
1314 \cs_generate_variant:Nn \spath_gweld:Nnn {NVV, NVn}
1315 \cs_new_protected_nopar:Npn \spath_gweld:Nn #1#2
1316 {
1317   \spath_gweld:NVn #1#1{#2}
1318 }
1319 \cs_generate_variant:Nn \spath_gweld:Nn {NV, Nv, cV, cv}

```

(End definition for `\spath_weld:Nnn` and others.)

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

```

\spath_append_no_move:Nn
\spath_gappend_no_move:Nnn
\spath_gappend_no_move:Nn
1320 \cs_new_protected_nopar:Npn \__spath_append_no_move:nn #1#2
1321 {
1322   \group_begin:
1323   \tl_set:Nn \l__spath_tmpa_tl {#1}
1324   \tl_set:Nn \l__spath_tmpb_tl {#2}
1325   \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
1326   \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
1327   \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
1328
1329   \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
1330   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
1331   \group_end:
1332 }
1333 \cs_generate_variant:Nn \__spath_append_no_move:nn {VV}
1334 \cs_new_protected_nopar:Npn \spath_append_no_move:Nnn #1#2#3
1335 {
1336   \__spath_append_no_move:nn {#2}{#3}
1337   \tl_set_eq:NN #1 \g__spath_output_tl
1338   \tl_gclear:N \g__spath_output_tl
1339 }

```

```

1340 \cs_generate_variant:Nn \spath_append_no_move:Nnn {NVV, NVn, NnV}
1341 \cs_new_protected_nopar:Npn \spath_append_no_move:Nn #1#2
1342 {
1343   \spath_append_no_move:NVn #1#1{#2}
1344 }
1345 \cs_generate_variant:Nn \spath_append_no_move:Nn {NV, cv, Nv}
1346 \cs_new_protected_nopar:Npn \spath_gappend_no_move:Nnn #1#2#3
1347 {
1348   \__spath_append_no_move:nn {#2}{#3}
1349   \tl_gset_eq:NN #1 \g__spath_output_tl
1350   \tl_gclear:N \g__spath_output_tl
1351 }
1352 \cs_generate_variant:Nn \spath_gappend_no_move:Nnn {NVV, NVn}
1353 \cs_new_protected_nopar:Npn \spath_gappend_no_move:Nn #1#2
1354 {
1355   \spath_gappend_no_move:NVn #1#1{#2}
1356 }
1357 \cs_generate_variant:Nn \spath_gappend_no_move:Nn {NV, cv, Nv}

```

(End definition for `\spath_append_no_move:Nnn` and others.)

<pre> \spath_append:Nnn \spath_append:Nn \spath_gappend:Nnn \spath_gappend:Nn </pre>	<p>Prepend the path from the second <code>spath</code> to the first.</p> <pre> 1358 \cs_new_protected_nopar:Npn \spath_append:Nnn #1#2#3 1359 { 1360 \tl_set:Nn #1 {#2} 1361 \tl_put_right:Nn #1 {#3} 1362 } 1363 \cs_generate_variant:Nn \spath_append:Nnn {NVV, NVn} 1364 \cs_new_protected_nopar:Npn \spath_append:Nn #1#2 1365 { 1366 \spath_append:NVn #1#1{#2} 1367 } 1368 \cs_generate_variant:Nn \spath_append:Nn {NV, Nv, cv} 1369 \cs_new_protected_nopar:Npn \spath_gappend:Nnn #1#2#3 1370 { 1371 \tl_gset:Nn #1 {#2} 1372 \tl_gput_right:Nn #1 {#3} 1373 } 1374 \cs_generate_variant:Nn \spath_gappend:Nnn {NVV, NVn} 1375 \cs_new_protected_nopar:Npn \spath_gappend:Nn #1#2 1376 { 1377 \spath_gappend:NVn #1#1{#2} 1378 } 1379 \cs_generate_variant:Nn \spath_gappend:Nn {NV, Nv, cv} </pre>
--	--

(End definition for `\spath_append:Nnn` and others.)

<pre> \spath_prepend_no_move:Nnn \spath_prepend_no_move:Nn \spath_gprepend_no_move:Nnn \spath_gprepend_no_move:Nn </pre>	<p>Prepend the path from the second <code>spath</code> to the first, removing the adjoining move.</p> <pre> 1380 \cs_new_protected_nopar:Npn \spath_prepend_no_move:Nnn #1#2#3 1381 { 1382 \spath_append_no_move:Nnn #1{#3}{#2} 1383 } 1384 \cs_generate_variant:Nn \spath_prepend_no_move:Nnn {NVV, NVn} 1385 \cs_new_protected_nopar:Npn \spath_prepend_no_move:Nn #1#2 1386 { </pre>
--	---

```

1387 \spath_prepend_no_move:NVn #1#1{#2}
1388 }
1389 \cs_generate_variant:Nn \spath_prepend_no_move:Nn {NV, cv}
1390 \cs_new_protected_nopar:Npn \spath_gprepend_no_move:Nnn #1#2#3
1391 {
1392   \spath_gappend_no_move:Nnn #1{#3}{#2}
1393 }
1394 \cs_generate_variant:Nn \spath_gprepend_no_move:Nnn {NVV, NVn}
1395 \cs_new_protected_nopar:Npn \spath_gprepend_no_move:Nn #1#2
1396 {
1397   \spath_gprepend_no_move:NVn #1#1{#2}
1398 }
1399 \cs_generate_variant:Nn \spath_gprepend_no_move:Nn {NV, cv}

```

(End definition for `\spath_prepend_no_move:Nnn` and others.)

```

\spath_prepend:Nnn      Prepend the path from the second spath to the first.
\spath_prepend:Nn      1400 \cs_new_protected_nopar:Npn \spath_prepend:Nnn #1#2#3
\spath_gprepend:Nnn      1401 {
\spath_gprepend:Nn      1402   \spath_append:Nnn #1{#3}{#2}
                        1403 }
                        1404 \cs_generate_variant:Nn \spath_prepend:Nnn {NVV, NVn}
                        1405 \cs_new_protected_nopar:Npn \spath_prepend:Nn #1#2
                        1406 {
                        1407   \spath_prepend:NVn #1#1{#2}
                        1408 }
                        1409 \cs_generate_variant:Nn \spath_prepend:Nn {NV}
                        1410 \cs_new_protected_nopar:Npn \spath_gprepend:Nnn #1#2#3
                        1411 {
                        1412   \spath_gappend:Nnn #1{#3}{#2}
                        1413 }
                        1414 \cs_generate_variant:Nn \spath_gprepend:Nnn {NVV, NVn}
                        1415 \cs_new_protected_nopar:Npn \spath_gprepend:Nn #1#2
                        1416 {
                        1417   \spath_gprepend:NVn #1#1{#2}
                        1418 }
                        1419 \cs_generate_variant:Nn \spath_gprepend:Nn {NV}

```

(End definition for `\spath_prepend:Nnn` and others.)

```

\spath_bake_round:Nn      The corner rounding routine is applied quite late in the process of building a soft path
\spath_bake_round:N      so this ensures that it is done.
\spath_gbake_round:Nn      1420 \cs_new_protected_nopar:Npn \__spath_bake_round:n #1
\spath_gbake_round:N      1421 {
                        1422   \group_begin:
                        1423   \tl_set:Nn \l__spath_tmpa_tl {#1}
                        1424   \pgf@processround \l__spath_tmpa_tl\l__spath_tmpb_tl
                        1425   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
                        1426   \group_end:
                        1427 }
                        1428 \cs_new_protected_nopar:Npn \spath_bake_round:Nn #1#2
                        1429 {
                        1430   \__spath_bake_round:n {#2}
                        1431   \tl_set_eq:NN #1 \g__spath_output_tl

```

```

1432 \tl_gclear:N \g__spath_output_tl
1433 }
1434 \cs_generate_variant:Nn \spath_bake_round:Nn {NV}
1435 \cs_new_protected_nopar:Npn \spath_bake_round:N #1
1436 {
1437   \spath_bake_round:NV #1#1
1438 }
1439 \cs_generate_variant:Nn \spath_bake_round:N {c}
1440 \cs_new_protected_nopar:Npn \spath_gbake_round:Nn #1#2
1441 {
1442   \__spath_bake_round:n {#2}
1443   \tl_gset_eq:NN #1 \g__spath_output_tl
1444   \tl_gclear:N \g__spath_output_tl
1445 }
1446 \cs_generate_variant:Nn \spath_gbake_round:Nn {NV}
1447 \cs_new_protected_nopar:Npn \spath_gbake_round:N #1
1448 {
1449   \spath_gbake_round:NV #1#1
1450 }
1451 \cs_generate_variant:Nn \spath_gbake_round:N {c}

```

(End definition for `\spath_bake_round:Nn` and others.)

```

\spath_close:Nn Appends a close path to the end of the path.
\spath_close:N
\spath_gclose:Nn
\spath_gclose:N
1452 \cs_new_protected_nopar:Npn \__spath_close:n #1
1453 {
1454   \group_begin:
1455   \tl_set:Nn \l__spath_tmpa_tl {#1}
1456   \spath_finalmovepoint:NV \l__spath_tmpb_tl \l__spath_tmpa_tl
1457   \tl_put_right:NV \l__spath_tmpa_tl \c_spath_closepath_tl
1458   \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
1459   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
1460   \group_end:
1461 }
1462 \cs_generate_variant:Nn \__spath_close:n {V}
1463 \cs_new_protected_nopar:Npn \spath_close:Nn #1#2
1464 {
1465   \__spath_close:n {#2}
1466   \tl_set_eq:NN #1 \g__spath_output_tl
1467   \tl_gclear:N \g__spath_output_tl
1468 }
1469 \cs_generate_variant:Nn \spath_close:Nn {NV}
1470 \cs_new_protected_nopar:Npn \spath_close:N #1
1471 {
1472   \spath_close:NV #1#1
1473 }
1474 \cs_generate_variant:Nn \spath_close:N {c}
1475 \cs_new_protected_nopar:Npn \spath_gclose:Nn #1#2
1476 {
1477   \__spath_close:n {#2}
1478   \tl_gset_eq:NN #1 \g__spath_output_tl
1479   \tl_gclear:N \g__spath_output_tl
1480 }
1481 \cs_generate_variant:Nn \spath_gclose:Nn {NV}

```



```

1482 \cs_new_protected_nopar:Npn \spath_gclose:N #1
1483 {
1484   \spath_gclose:NV #1#1
1485 }
1486 \cs_generate_variant:Nn \spath_gclose:N {c}

```

(End definition for \spath_close:Nn and others.)

\spath_open:Nn Removes all close paths from the path, replacing them by lineto if they move any distance.

```

\spath_open:N
\spath_gopen:Nn
\spath_gopen:N
1487 \cs_new_protected_nopar:Npn \__spath_open:n #1
1488 {
1489   \group_begin:
1490   \tl_set:Nn \l__spath_tmpa_tl {#1}
1491   \tl_clear:N \l__spath_tmpb_tl
1492   \bool_until_do:nn {
1493     \tl_if_empty_p:N \l__spath_tmpa_tl
1494   }
1495   {
1496     \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
1497     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1498
1499     \tl_case:NnF \l__spath_tmpc_tl
1500     {
1501       \c_spath_closepath_tl {
1502
1503         \bool_if:nF
1504         {
1505           \dim_compare_p:n
1506           {
1507             \l__spath_move_x_dim == \l__spath_tmpa_dim
1508           }
1509           &&
1510           \dim_compare_p:n
1511           {
1512             \l__spath_move_y_dim == \l__spath_tmpb_dim
1513           }
1514         }
1515         {
1516           \tl_put_right:NV \l__spath_tmpb_tl \c_spath_lineto_tl
1517
1518           \tl_put_right:Nx \l__spath_tmpb_tl {
1519             { \dim_use:N \l__spath_move_x_dim }
1520             { \dim_use:N \l__spath_move_y_dim }
1521           }
1522         }
1523
1524         \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
1525         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1526         \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
1527         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1528       }
1529
1530       \c_spath_moveto_tl {

```

```

1531 \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpc_tl
1532
1533 \dim_set:Nn \l__spath_move_x_dim {\tl_head:N \l__spath_tmpa_tl}
1534 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1535 \dim_set:Nn \l__spath_move_y_dim {\tl_head:N \l__spath_tmpa_tl}
1536 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1537
1538 \tl_put_right:Nx \l__spath_tmpb_tl {
1539   { \dim_use:N \l__spath_move_x_dim }
1540   { \dim_use:N \l__spath_move_y_dim }
1541 }
1542
1543 \dim_set_eq:NN \l__spath_tmpa_dim \l__spath_move_x_dim
1544 \dim_set_eq:NN \l__spath_tmpb_dim \l__spath_move_y_dim
1545 }
1546 }
1547 {
1548   \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpc_tl
1549
1550   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
1551   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1552   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
1553   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1554
1555   \tl_put_right:Nx \l__spath_tmpb_tl {
1556     { \dim_use:N \l__spath_tmpa_dim }
1557     { \dim_use:N \l__spath_tmpb_dim }
1558   }
1559 }
1560 }
1561 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
1562 \group_end:
1563 }
1564 \cs_new_protected_nopar:Npn \spath_open:Nn #1#2
1565 {
1566   \__spath_open:n {#2}
1567   \tl_set_eq:NN #1 \g__spath_output_tl
1568   \tl_gclear:N \g__spath_output_tl
1569 }
1570 \cs_generate_variant:Nn \spath_open:Nn {NV}
1571 \cs_new_protected_nopar:Npn \spath_open:N #1
1572 {
1573   \spath_open:NV #1#1
1574 }
1575 \cs_new_protected_nopar:Npn \spath_gopen:Nn #1#2
1576 {
1577   \__spath_open:n {#2}
1578   \tl_gset_eq:NN #1 \g__spath_output_tl
1579   \tl_gclear:N \g__spath_output_tl
1580 }
1581 \cs_generate_variant:Nn \spath_gopen:Nn {NV}
1582 \cs_new_protected_nopar:Npn \spath_gopen:N #1
1583 {
1584   \spath_gopen:NV #1#1

```

1585 }

(End definition for \spath_open:Nn and others.)

\spath_remove_empty_components:Nn
\spath_remove_empty_components:N
\spath_gremove_empty_components:Nn
\spath_gremove_empty_components:N

Remove any component that is simply a moveto.

```

1586 \cs_new_protected_nopar:Npn \__spath_remove_empty_components:n #1
1587 {
1588   \group_begin:
1589   \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
1590   \tl_clear:N \l__spath_tmpa_tl
1591   \seq_map_inline:Nn \l__spath_tmpa_seq
1592   {
1593     \int_compare:nF
1594     {
1595       \tl_count:n {##1} == 3
1596     }
1597     {
1598       \tl_put_right:Nn \l__spath_tmpa_tl {##1}
1599     }
1600   }
1601   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
1602   \group_end:
1603 }
1604 \cs_new_protected_nopar:Npn \spath_remove_empty_components:Nn #1#2
1605 {
1606   \__spath_remove_empty_components:n {#2}
1607   \tl_set_eq:NN #1 \g__spath_output_tl
1608   \tl_gclear:N \g__spath_output_tl
1609 }
1610 \cs_generate_variant:Nn \spath_remove_empty_components:Nn {NV}
1611 \cs_new_protected_nopar:Npn \spath_remove_empty_components:N #1
1612 {
1613   \spath_remove_empty_components:NV #1#1
1614 }
1615 \cs_generate_variant:Nn \spath_remove_empty_components:N {c}
1616 \cs_new_protected_nopar:Npn \spath_gremove_empty_components:Nn #1#2
1617 {
1618   \__spath_remove_empty_components:n {#2}
1619   \tl_gset_eq:NN #1 \g__spath_output_tl
1620   \tl_gclear:N \g__spath_output_tl
1621 }
1622 \cs_generate_variant:Nn \spath_gremove_empty_components:Nn {NV}
1623 \cs_new_protected_nopar:Npn \spath_gremove_empty_components:N #1
1624 {
1625   \spath_gremove_empty_components:NV #1#1
1626 }
1627 \cs_generate_variant:Nn \spath_gremove_empty_components:N {c}

```

(End definition for \spath_remove_empty_components:Nn and others.)

\spath_if_eq:nn Test if two soft paths are equal, we allow a little tolerance on the calculations.

```

1628 \prg_new_protected_conditional:Npnn \spath_if_eq:nn #1#2 { T, F, TF }
1629 {
1630   \group_begin:
1631   \tl_set:Nn \l__spath_tmpa_tl {#1}

```

```

1632 \tl_set:Nn \l__spath_tmpb_tl {#2}
1633 \bool_gset_true:N \g__spath_tmpa_bool
1634 \int_compare:nNnTF
1635 {\tl_count:N \l__spath_tmpa_tl}
1636 =
1637 {\tl_count:N \l__spath_tmpb_tl}
1638 {
1639   \int_step_inline:nnnn {1} {3} {\tl_count:N \l__spath_tmpa_tl}
1640   {
1641     \tl_set:Nx \l__spath_tmpc_tl {\tl_item:Nn \l__spath_tmpa_tl {##1}}
1642     \tl_set:Nx \l__spath_tmpd_tl {\tl_item:Nn \l__spath_tmpb_tl {##1}}
1643     \tl_if_eq:NnF \l__spath_tmpc_tl \l__spath_tmpd_tl
1644     {
1645       \bool_gset_false:N \g__spath_tmpa_bool
1646     }
1647     \dim_set:Nn \l__spath_tmpa_dim {\tl_item:Nn \l__spath_tmpa_tl {##1+1}}
1648     \dim_set:Nn \l__spath_tmpb_dim {\tl_item:Nn \l__spath_tmpb_tl {##1+1}}
1649     \dim_compare:nF
1650     {
1651       \dim_abs:n
1652       {
1653         \l__spath_tmpa_dim - \l__spath_tmpb_dim
1654       }
1655       < 0.001pt
1656     }
1657     {
1658       \bool_gset_false:N \g__spath_tmpa_bool
1659     }
1660     \dim_set:Nn \l__spath_tmpa_dim {\tl_item:Nn \l__spath_tmpa_tl {##1+2}}
1661     \dim_set:Nn \l__spath_tmpb_dim {\tl_item:Nn \l__spath_tmpb_tl {##1+2}}
1662     \dim_compare:nF
1663     {
1664       \dim_abs:n
1665       {
1666         \l__spath_tmpa_dim - \l__spath_tmpb_dim
1667       }
1668       < 0.001pt
1669     }
1670     {
1671       \bool_gset_false:N \g__spath_tmpa_bool
1672     }
1673   }
1674 }
1675 {
1676   \bool_gset_false:N \g__spath_tmpa_bool
1677 }
1678 \group_end:
1679 \bool_if:NNTF \g__spath_tmpa_bool
1680 {
1681   \prg_return_true:
1682 }
1683 {
1684   \prg_return_false:
1685 }

```

```

1686 }
1687 \prg_generate_conditional_variant:Nnn \spath_if_eq:nn {VV, Vn, nV, vv} {TF, T, F}

(End definition for \spath_if_eq:nn.)

```

3.4 Splitting Commands

`\spath_split_curve:NNnn` Splits a Bezier cubic into pieces, storing the pieces in the first two arguments.

```

\spath_gsplit_curve:NNnn
1688 \cs_new_protected_nopar:Npn \__spath_split_curve:nn #1#2
1689 {
1690   \group_begin:
1691   \tl_set_eq:NN \l__spath_tmpa_tl \c_spath_moveto_tl
1692   \tl_put_right:Nx \l__spath_tmpa_tl {
1693     {\tl_item:nn {#1} {2}}
1694     {\tl_item:nn {#1} {3}}
1695   }
1696   \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curvetoa_tl
1697   \tl_put_right:Nx \l__spath_tmpa_tl
1698   {
1699     {\fp_to_dim:n
1700      {
1701        (1 - #2) * \tl_item:nn {#1} {2} + (#2) * \tl_item:nn {#1} {5}
1702      }}
1703     {\fp_to_dim:n
1704      {
1705        (1 - #2) * \tl_item:nn {#1} {3} + (#2) * \tl_item:nn {#1} {6}
1706      }}
1707   }
1708
1709   \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curvetob_tl
1710   \tl_put_right:Nx \l__spath_tmpa_tl
1711   {
1712     {\fp_to_dim:n
1713      {
1714        (1 - #2)^2 * \tl_item:nn {#1} {2}
1715        + 2 * (1 - #2) * (#2) * \tl_item:nn {#1} {5}
1716        + (#2)^2 * \tl_item:nn {#1} {8}
1717      }}
1718     {\fp_to_dim:n
1719      {
1720        (1 - #2)^2 * \tl_item:nn {#1} {3}
1721        + 2 * (1 - #2) * (#2) * \tl_item:nn {#1} {6}
1722        + (#2)^2 * \tl_item:nn {#1} {9}
1723      }}
1724   }
1725
1726   \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curveto_tl
1727   \tl_put_right:Nx \l__spath_tmpa_tl
1728   {
1729     {\fp_to_dim:n
1730      {
1731        (1 - #2)^3 * \tl_item:nn {#1} {2}
1732        + 3 * (1 - #2)^2 * (#2) * \tl_item:nn {#1} {5}
1733        + 3 * (1 - #2) * (#2)^2 * \tl_item:nn {#1} {8}

```

```

1734         + (#2)^3 * \tl_item:nn {#1} {11}
1735     }}
1736     {\fp_to_dim:n
1737     {
1738         (1 - #2)^3 * \tl_item:nn {#1} {3}
1739         + 3 * (1 - #2)^2 * (#2) * \tl_item:nn {#1} {6}
1740         + 3 * (1 - #2) * (#2)^2 * \tl_item:nn {#1} {9}
1741         + (#2)^3 * \tl_item:nn {#1} {12}
1742     }}
1743 }
1744
1745 \tl_gclear:N \g__spath_output_tl
1746 \__spath_tl_gput_right_braced:N \g__spath_output_tl \l__spath_tmpa_tl
1747
1748 \tl_clear:N \l__spath_tmpa_tl
1749 \tl_set_eq:NN \l__spath_tmpa_tl \c_spath_moveto_tl
1750 \tl_put_right:Nx \l__spath_tmpa_tl
1751 {
1752     {\fp_to_dim:n
1753     {
1754         (1 - #2)^3 * \tl_item:nn {#1} {2}
1755         + 3 * (1 - #2)^2 * (#2) * \tl_item:nn {#1} {5}
1756         + 3 * (1 - #2) * (#2)^2 * \tl_item:nn {#1} {8}
1757         + (#2)^3 * \tl_item:nn {#1} {11}
1758     }}
1759     {\fp_to_dim:n
1760     {
1761         (1 - #2)^3 * \tl_item:nn {#1} {3}
1762         + 3 * (1 - #2)^2 * (#2) * \tl_item:nn {#1} {6}
1763         + 3 * (1 - #2) * (#2)^2 * \tl_item:nn {#1} {9}
1764         + (#2)^3 * \tl_item:nn {#1} {12}
1765     }}
1766 }
1767
1768 \tl_put_right:N \l__spath_tmpa_tl \c_spath_curvetoa_tl
1769 \tl_put_right:Nx \l__spath_tmpa_tl
1770 {
1771     {\fp_to_dim:n
1772     {
1773         (1 - #2)^2 * \tl_item:nn {#1} {5}
1774         + 2 * (1 - #2) * (#2) * \tl_item:nn {#1} {8}
1775         + (#2)^2 * \tl_item:nn {#1} {11}
1776     }}
1777     {\fp_to_dim:n
1778     {
1779         (1 - #2)^2 * \tl_item:nn {#1} {6}
1780         + 2 * (1 - #2) * (#2) * \tl_item:nn {#1} {9}
1781         + (#2)^2 * \tl_item:nn {#1} {12}
1782     }}
1783 }
1784 \tl_put_right:N \l__spath_tmpa_tl \c_spath_curvetob_tl
1785 \tl_put_right:Nx \l__spath_tmpa_tl
1786 {
1787     {\fp_to_dim:n

```

```

1788     {
1789       (1 - #2) * \tl_item:nn {#1} {8} + (#2) * \tl_item:nn {#1} {11}
1790     }
1791     {\fp_to_dim:n
1792     {
1793       (1 - #2) * \tl_item:nn {#1} {9} + (#2) * \tl_item:nn {#1} {12}
1794     }
1795   }
1796   \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curveto_tl
1797   \tl_put_right:Nx \l__spath_tmpa_tl {
1798     {\tl_item:nn {#1} {11}}
1799     {\tl_item:nn {#1} {12}}
1800   }
1801
1802   \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpa_tl
1803   \group_end:
1804 }
1805 \cs_generate_variant:Nn \__spath_split_curve:nn {nv, nV}
1806 \cs_new_protected_nopar:Npn \spath_split_curve:NNnn #1#2#3#4
1807 {
1808   \__spath_split_curve:nn {#3}{#4}
1809   \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
1810   \tl_set:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
1811   \tl_gclear:N \g__spath_output_tl
1812 }
1813 \cs_generate_variant:Nn \spath_split_curve:NNnn {NNnV, NNnV, NNnV}
1814 \cs_new_protected_nopar:Npn \spath_gsplit_curve:NNnn #1#2#3#4
1815 {
1816   \__spath_split_curve:nn {#3}{#4}
1817   \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
1818   \tl_gset:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
1819   \tl_gclear:N \g__spath_output_tl
1820 }
1821 \cs_generate_variant:Nn \spath_gsplit_curve:NNnn {NNnV, NNnV, NNnV}

```

(End definition for \spath_split_curve:NNnn and \spath_gsplit_curve:NNnn.)

\spath_maybe_split_curve:Nn Possibly splits a bezier curve to ensure that the pieces don't self-intersect. 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!

```

1822 \cs_new_protected_nopar:Npn \__spath_maybe_split_curve:n #1
1823 {
1824   \group_begin:
1825   \fp_set:Nn \l__spath_tmpa_fp
1826   {
1827     (
1828       \tl_item:nn {#1} {3}
1829       - 3 * \tl_item:nn {#1} {6}
1830       + 3 * \tl_item:nn {#1} {9}
1831       - \tl_item:nn {#1} {12}
1832     )
1833     *
1834     (3 * \tl_item:nn {#1} {8} - 3 * \tl_item:nn {#1} {11})

```

```

1835 -
1836 (
1837   \tl_item:nn {#1} {2}
1838   - 3 * \tl_item:nn {#1} {5}
1839   + 3 * \tl_item:nn {#1} {8}
1840   - \tl_item:nn {#1} {11}
1841 )
1842 *
1843 (3 * \tl_item:nn {#1} {9} - 3 * \tl_item:nn {#1} {12})
1844 }
1845 \fp_set:Nn \l__spath_tmpb_fp
1846 {
1847   (
1848     \tl_item:nn {#1} {2}
1849     - 3 * \tl_item:nn {#1} {5}
1850     + 3 * \tl_item:nn {#1} {8}
1851     - \tl_item:nn {#1} {11}
1852   )
1853   *
1854   (
1855     3 * \tl_item:nn {#1} {6}
1856     - 6 * \tl_item:nn {#1} {9}
1857     + 3 * \tl_item:nn {#1} {12}
1858   )
1859   -
1860   (
1861     \tl_item:nn {#1} {3}
1862     - 3 * \tl_item:nn {#1} {6}
1863     + 3 * \tl_item:nn {#1} {9}
1864     - \tl_item:nn {#1} {12}
1865   )
1866   *
1867   (
1868     3 * \tl_item:nn {#1} {5}
1869     - 6 * \tl_item:nn {#1} {8}
1870     + 3 * \tl_item:nn {#1} {11}
1871   )
1872 }
1873 \fp_compare:nTF
1874 {
1875   \l__spath_tmpb_fp != 0
1876 }
1877 {
1878   \fp_set:Nn \l__spath_tmpa_fp {.5 * \l__spath_tmpa_fp / \l__spath_tmpb_fp}
1879   \fp_compare:nTF
1880   {
1881     0 < \l__spath_tmpa_fp && \l__spath_tmpa_fp < 1
1882   }
1883   {
1884     \__spath_split_curve:nV {#1} \l__spath_tmpa_fp
1885   }
1886   {
1887     \tl_gset:Nn \g__spath_output_tl { {#1} {} }
1888   }

```



```

1889 }
1890 {
1891   \tl_gset:Nn \g__spath_output_tl { {#1} {} }
1892 }
1893 \group_end:
1894 }
1895 \cs_new_protected_nopar:Npn \spath_maybe_split_curve:NNn #1#2#3
1896 {
1897   \__spath_maybe_split_curve:n {#3}
1898   \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
1899   \tl_set:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
1900   \tl_gclear:N \g__spath_output_tl
1901 }
1902 \cs_generate_variant:Nn \spath_maybe_split_curve:NNn {NNn, NNV }
1903 \cs_new_protected_nopar:Npn \spath_maybe_gsplit_curve:NNn #1#2#3
1904 {
1905   \__spath_maybe_split_curve:n {#3}
1906   \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
1907   \tl_gset:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
1908   \tl_gclear:N \g__spath_output_tl
1909 }
1910 \cs_generate_variant:Nn \spath_maybe_gsplit_curve:NNn {NNn, NNV}

```

(End definition for \spath_maybe_split_curve:Nn and \spath_maybe_gsplit_curve:Nn.)

\spath_split_curves:Nn Slurp through the path ensuring that beziers don't self-intersect.

\spath_gsplit_curves:Nn

```

1911 \cs_new_protected_nopar:Npn \__spath_split_curves:n #1
1912 {
1913   \group_begin:
1914   \tl_set:Nn \l__spath_tmpa_tl {#1}
1915   \tl_clear:N \l__spath_tmpb_tl
1916   \tl_clear:N \l__spath_tmpc_tl
1917   \bool_do_until:nn
1918   {
1919     \tl_if_empty_p:N \l__spath_tmpa_tl
1920   }
1921   {
1922     \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
1923     \tl_case:NnF \l__spath_tmpc_tl
1924     {
1925       \c_spath_curvetoa_tl
1926       {
1927         \tl_clear:N \l__spath_tmpd_tl
1928         \tl_set_eq:NN \l__spath_tmpd_tl \c_spath_moveto_tl
1929         \tl_put_right:Nx \l__spath_tmpd_tl
1930         {
1931           { \dim_use:N \l__spath_tmpa_dim }
1932           { \dim_use:N \l__spath_tmpb_dim }
1933         }
1934         \dim_set:Nn \l__spath_tmpa_dim
1935         {
1936           \tl_item:Nn \l__spath_tmpa_tl {8}
1937         }
1938         \dim_set:Nn \l__spath_tmpb_dim

```

```

1939     {
1940         \tl_item:Nn \l__spath_tmpa_tl {9}
1941     }
1942     \prg_replicate:nn {3}
1943     {
1944         \tl_put_right:Nx \l__spath_tmpd_tl
1945         {
1946             \tl_item:Nn \l__spath_tmpa_tl {1}
1947             {\tl_item:Nn \l__spath_tmpa_tl {2}}
1948             {\tl_item:Nn \l__spath_tmpa_tl {3}}
1949         }
1950         \prg_replicate:nn {3}
1951         {
1952             \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1953         }
1954     }
1955
1956     \spath_maybe_split_curve:NNV
1957     \l__spath_tmpd_tl
1958     \l__spath_tmpe_tl
1959     \l__spath_tmpd_tl
1960     \prg_replicate:nn {3}
1961     {
1962         \tl_set:Nx \l__spath_tmpd_tl {\tl_tail:N \l__spath_tmpd_tl}
1963         \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl}
1964     }
1965     \tl_put_right:NV \l__spath_tmpe_tl \l__spath_tmpd_tl
1966     \tl_put_right:NV \l__spath_tmpe_tl \l__spath_tmpe_tl
1967 }
1968 }
1969 {
1970     \dim_set:Nn \l__spath_tmpa_dim
1971     {
1972         \tl_item:Nn \l__spath_tmpa_tl {2}
1973     }
1974     \dim_set:Nn \l__spath_tmpe_dim
1975     {
1976         \tl_item:Nn \l__spath_tmpa_tl {3}
1977     }
1978     \tl_put_right:Nx \l__spath_tmpe_tl
1979     {
1980         \tl_item:Nn \l__spath_tmpa_tl {1}
1981         {\tl_item:Nn \l__spath_tmpa_tl {2}}
1982         {\tl_item:Nn \l__spath_tmpa_tl {3}}
1983     }
1984     \prg_replicate:nn {3}
1985     {
1986         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1987     }
1988 }
1989 }
1990 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpe_tl
1991 \group_end:
1992 }

```

```

1993 \cs_new_protected_nopar:Npn \spath_split_curves:Nn #1#2
1994 {
1995   \__spath_split_curves:n {#2}
1996   \tl_set_eq:NN #1 \g__spath_output_tl
1997   \tl_gclear:N \g__spath_output_tl
1998 }
1999 \cs_generate_variant:Nn \spath_split_curves:Nn {NV, cV, cn, cv }
2000 \cs_new_protected_nopar:Npn \spath_split_curves:N #1
2001 {
2002   \spath_split_curves:NV #1#1
2003 }
2004 \cs_generate_variant:Nn \spath_split_curves:N {c}
2005 \cs_new_protected_nopar:Npn \spath_gsplit_curves:Nn #1#2
2006 {
2007   \__spath_split_curves:n {#2}
2008   \tl_gset_eq:NN #1 \g__spath_output_tl
2009   \tl_gclear:N \g__spath_output_tl
2010 }
2011 \cs_generate_variant:Nn \spath_gsplit_curves:Nn {NV, cV, cn, cv }
2012 \cs_new_protected_nopar:Npn \spath_gsplit_curves:N #1
2013 {
2014   \spath_gsplit_curves:NV #1#1
2015 }
2016 \cs_generate_variant:Nn \spath_gsplit_curves:N {c}

```

(End definition for \spath_split_curves:Nn and \spath_gsplit_curves:Nn.)

\spath_split_line:NNnn
\spath_gsplit_line:NNnn

Splits a line segment.

```

2017 \cs_new_protected_nopar:Npn \__spath_split_line:nn #1#2
2018 {
2019   \group_begin:
2020   \tl_set_eq:NN \l__spath_tmpa_tl \c_spath_moveto_tl
2021   \tl_put_right:Nx \l__spath_tmpa_tl {
2022     {\tl_item:nn {#1} {2}}
2023     {\tl_item:nn {#1} {3}}
2024   }
2025   \tl_put_right:NV \l__spath_tmpa_tl \c_spath_lineto_tl
2026   \tl_put_right:Nx \l__spath_tmpa_tl
2027   {
2028     {\fp_to_dim:n
2029     {
2030       (1 - #2) * \tl_item:nn {#1} {2} + (#2) * \tl_item:nn {#1} {5}
2031     }}
2032     {\fp_to_dim:n
2033     {
2034       (1 - #2) * \tl_item:nn {#1} {3} + (#2) * \tl_item:nn {#1} {6}
2035     }}
2036   }
2037   \tl_gclear:N \g__spath_output_tl
2038   \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpa_tl
2039
2040   \tl_clear:N \l__spath_tmpa_tl
2041   \tl_set_eq:NN \l__spath_tmpa_tl \c_spath_moveto_tl
2042   \tl_put_right:Nx \l__spath_tmpa_tl

```

```

2043 {
2044   {\fp_to_dim:n
2045   {
2046     (1 - #2) * \tl_item:nn {#1} {2} + (#2) * \tl_item:nn {#1} {5}
2047   }}
2048   {\fp_to_dim:n
2049   {
2050     (1 - #2) * \tl_item:nn {#1} {3} + (#2) * \tl_item:nn {#1} {6}
2051   }}
2052 }
2053 \tl_put_right:NV \l__spath_tmpa_tl \c_spath_lineto_tl
2054 \tl_put_right:Nx \l__spath_tmpa_tl {
2055   {\tl_item:nn {#1} {5}}
2056   {\tl_item:nn {#1} {6}}
2057 }
2058
2059 \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpa_tl
2060 \group_end:
2061 }
2062 \cs_new_protected_nopar:Npn \spath_split_line:NNnn #1#2#3#4
2063 {
2064   \__spath_split_line:nn {#3}{#4}
2065   \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
2066   \tl_set:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
2067   \tl_gclear:N \g__spath_output_tl
2068 }
2069 \cs_generate_variant:Nn \spath_split_line:NNnn {NNnV, NNVn, NNVV}
2070 \cs_new_protected_nopar:Npn \spath_gsplit_line:NNnn #1#2#3#4
2071 {
2072   \__spath_split_line:nn {#3}{#4}
2073   \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
2074   \tl_gset:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
2075   \tl_gclear:N \g__spath_output_tl
2076 }
2077 \cs_generate_variant:Nn \spath_gsplit_line:NNnn {NNnV, NNVn, NNVV}

```

(End definition for \spath_split_line:NNnn and \spath_gsplit_line:NNnn.)

\spath_split_at:NNnn Split a path according to the parameter generated by the intersection routine. The versions with two N arguments stores the two parts in two macros, the version with a single N joins them back into a single path (as separate components).

```

\spath_split_at:Nnn
\spath_split_at:Nn
\spath_split_at:Nn
\spath_gsplit_at:NNnn\spath_gsplit_at:Nnn
\spath_gsplit_at:Nn
2078 \cs_new_protected_nopar:Npn \__spath_split_at:nn #1#2
2079 {
2080   \group_begin:
2081   \int_set:Nn \l__spath_tmpa_int {\fp_to_int:n {floor(#2) + 1}}
2082   \fp_set:Nn \l__spath_tmpa_fp {#2 - floor(#2)}
2083
2084   % Is split point near one end or other of a component?
2085   \fp_compare:nT
2086   {
2087     \l__spath_tmpa_fp < 0.01
2088   }
2089   {
2090     % Near the start, so we'll place it at the start

```

```

2091     \fp_set:Nn \l__spath_tmpa_fp {0}
2092   }
2093   \fp_compare:nT
2094   {
2095     \l__spath_tmpa_fp > 0.99
2096   }
2097   {
2098     % Near the end, so we'll place it at the end
2099     \fp_set:Nn \l__spath_tmpa_fp {0}
2100     \int_incr:N \l__spath_tmpa_int
2101   }
2102
2103   \int_zero:N \l__spath_tmpe_int
2104   \bool_set_true:N \l__spath_tmpe_bool
2105
2106   \tl_set:Nn \l__spath_tmpe_tl {#1}
2107   \tl_clear:N \l__spath_tmpe_tl
2108
2109   \dim_zero:N \l__spath_tmpe_dim
2110   \dim_zero:N \l__spath_tmpe_dim
2111
2112   \bool_until_do:nn {
2113     \tl_if_empty_p:N \l__spath_tmpe_tl
2114     ||
2115     \int_compare_p:n { \l__spath_tmpe_int == \l__spath_tmpe_int }
2116   }
2117   {
2118     \tl_set:Nx \l__spath_tmpe_tl {\tl_head:N \l__spath_tmpe_tl}
2119     \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2120     \tl_case:Nn \l__spath_tmpe_tl
2121     {
2122       \c_spath_lineto_tl
2123       {
2124         \int_incr:N \l__spath_tmpe_int
2125       }
2126       \c_spath_curvetoa_tl
2127       {
2128         \int_incr:N \l__spath_tmpe_int
2129       }
2130     }
2131     \int_compare:nT { \l__spath_tmpe_int < \l__spath_tmpe_int }
2132     {
2133       \tl_put_right:NV \l__spath_tmpe_tl \l__spath_tmpe_tl
2134
2135       \tl_put_right:Nx \l__spath_tmpe_tl
2136       {{ \tl_head:N \l__spath_tmpe_tl }}
2137       \dim_set:Nn \l__spath_tmpe_dim {\tl_head:N \l__spath_tmpe_tl}
2138       \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2139
2140       \tl_put_right:Nx \l__spath_tmpe_tl
2141       {{ \tl_head:N \l__spath_tmpe_tl }}
2142       \dim_set:Nn \l__spath_tmpe_dim {\tl_head:N \l__spath_tmpe_tl}
2143       \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2144

```

```

2145     }
2146 }
2147
2148 \tl_clear:N \l__spath_tmpd_tl
2149 \tl_put_right:NV \l__spath_tmpd_tl \c_spath_moveto_tl
2150 \tl_put_right:Nx \l__spath_tmpd_tl
2151 {
2152   {\dim_use:N \l__spath_tmpa_dim}
2153   {\dim_use:N \l__spath_tmpb_dim}
2154 }
2155
2156 \fp_compare:nTF
2157 {
2158   \l__spath_tmpa_fp == 0
2159 }
2160 {
2161   \tl_set_eq:NN \l__spath_tmpb_tl \l__spath_tmpd_tl
2162   \tl_if_empty:NF \l__spath_tmpe_tl
2163   {
2164     \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpf_tl
2165     \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpe_tl
2166   }
2167 }
2168 {
2169
2170   \tl_case:Nn \l__spath_tmpf_tl
2171   {
2172     \c_spath_lineto_tl
2173     {
2174       \tl_put_right:NV \l__spath_tmpd_tl \l__spath_tmpf_tl
2175       \tl_put_right:Nx \l__spath_tmpd_tl
2176       {{ \tl_head:N \l__spath_tmpe_tl }}
2177       \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2178
2179       \tl_put_right:Nx \l__spath_tmpd_tl
2180       {{ \tl_head:N \l__spath_tmpe_tl }}
2181       \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2182
2183       \spath_split_line:NNVV
2184       \l__spath_tmpa_tl
2185       \l__spath_tmpb_tl
2186       \l__spath_tmpd_tl
2187       \l__spath_tmpa_fp
2188
2189       \prg_replicate:nn {3} {
2190         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
2191       }
2192
2193       \tl_put_right:NV \l__spath_tmpe_tl \l__spath_tmpa_tl
2194       \tl_put_right:NV \l__spath_tmpe_tl \l__spath_tmpe_tl
2195     }
2196     \c_spath_curvetoa_tl
2197     {
2198       \tl_put_right:NV \l__spath_tmpd_tl \l__spath_tmpf_tl

```

```

2199 \tl_put_right:Nx \l__spath_tmpd_tl
2200 {{ \tl_head:N \l__spath_tmpe_tl }}
2201 \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2202
2203 \tl_put_right:Nx \l__spath_tmpd_tl
2204 {{ \tl_head:N \l__spath_tmpe_tl }}
2205 \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2206
2207 \prg_replicate:nn {2} {
2208
2209 \tl_put_right:Nx \l__spath_tmpd_tl
2210 { \tl_head:N \l__spath_tmpe_tl }
2211 \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2212
2213 \tl_put_right:Nx \l__spath_tmpd_tl
2214 {{ \tl_head:N \l__spath_tmpe_tl }}
2215 \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2216
2217 \tl_put_right:Nx \l__spath_tmpd_tl
2218 {{ \tl_head:N \l__spath_tmpe_tl }}
2219 \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2220 }
2221
2222 \spath_split_curve:NNVV
2223 \l__spath_tmpa_tl
2224 \l__spath_tmpe_tl
2225 \l__spath_tmpe_tl \l__spath_tmpe_tl
2226
2227 \prg_replicate:nn {3} {
2228 \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl}
2229 }
2230
2231 \tl_put_right:NV \l__spath_tmpe_tl \l__spath_tmpe_tl
2232 \tl_put_right:NV \l__spath_tmpe_tl \l__spath_tmpe_tl
2233 }
2234 }
2235 }
2236
2237 \tl_gclear:N \g__spath_output_tl
2238 \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpe_tl
2239 \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpe_tl
2240 \group_end:
2241 }
2242 \cs_new_protected_nopar:Npn \spath_split_at:NNnn #1#2#3#4
2243 {
2244 \__spath_split_at:nn {#3}{#4}
2245 \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
2246 \tl_set:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
2247 \tl_gclear:N \g__spath_output_tl
2248 }
2249 \cs_generate_variant:Nn \spath_split_at:NNnn {NNVn, NNVV, NNnV}
2250 \cs_new_protected_nopar:Npn \spath_gsplit_at:NNnn #1#2#3#4
2251 {
2252 \__spath_split_at:nn {#3}{#4}

```

```

2253 \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
2254 \tl_gset:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
2255 \tl_gclear:N \g__spath_output_tl
2256 }
2257 \cs_generate_variant:Nn \spath_gsplit_at:NNnn {NNVn, NNVV, NNnV}
2258 \cs_new_protected_nopar:Npn \spath_split_at:Nnn #1#2#3
2259 {
2260   \__spath_split_at:nn {#2}{#3}
2261   \tl_set:Nx #1
2262   {
2263     \tl_item:Nn \g__spath_output_tl {1}
2264     \tl_item:Nn \g__spath_output_tl {2}
2265   }
2266   \tl_gclear:N \g__spath_output_tl
2267 }
2268 \cs_generate_variant:Nn \spath_split_at:Nnn {NVn, NVV}
2269 \cs_new_protected_nopar:Npn \spath_split_at:Nn #1#2
2270 {
2271   \spath_split_at:NVn #1#1{#2}
2272 }
2273 \cs_new_protected_nopar:Npn \spath_gsplit_at:Nnn #1#2#3
2274 {
2275   \__spath_split_at:nn {#2}{#3}
2276   \tl_gset:Nx #1
2277   {
2278     \tl_item:Nn \g__spath_output_tl {1}
2279     \tl_item:Nn \g__spath_output_tl {2}
2280   }
2281   \tl_gclear:N \g__spath_output_tl
2282 }
2283 \cs_generate_variant:Nn \spath_gsplit_at:Nnn {NVn, NVV}
2284 \cs_new_protected_nopar:Npn \spath_gsplit_at:Nn #1#2
2285 {
2286   \spath_gsplit_at:NVn #1#1{#2}
2287 }

```

(End definition for `\spath_split_at:NNnn` and others.)

3.5 Shortening Paths

This code relates to shortening paths. For curved paths, the routine uses the derivative at the end to figure out how far back to shorten. This means that the actual length that it shortens by is approximate, but it is guaranteed to be along its length.

As in the previous section, there are various versions. In particular, there are versions where the path can be specified by a macro and is saved back into that macro.

`\spath_shorten_at_end:Nnn` This macro shortens a path from the end by a dimension.

```

2288 \cs_new_protected_nopar:Npn \__spath_shorten_at_end:nn #1#2
2289 {
2290   \int_compare:nTF
2291   {
2292     \tl_count:n {#1} > 3
2293   }

```



```

2294 {
2295   \group_begin:
2296   \tl_set:Nn \l__spath_tmpa_tl {#1}
2297   \tl_reverse:N \l__spath_tmpa_tl
2298
2299   \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {3}}
2300
2301   \tl_clear:N \l__spath_tmpe_tl
2302   \tl_if_eq:NNTF \l__spath_tmpb_tl \c_spath_curveto_tl
2303   {
2304     \int_set:Nn \l__spath_tmpa_int {3}
2305   }
2306   {
2307     \int_set:Nn \l__spath_tmpa_int {1}
2308   }
2309
2310   \prg_replicate:nn { \l__spath_tmpa_int }
2311   {
2312     \tl_put_right:Nx \l__spath_tmpe_tl
2313     {
2314       {\tl_head:N \l__spath_tmpa_tl}
2315     }
2316     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
2317     \tl_put_right:Nx \l__spath_tmpe_tl
2318     {
2319       {\tl_head:N \l__spath_tmpa_tl}
2320     }
2321     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
2322     \tl_put_right:Nx \l__spath_tmpe_tl
2323     {
2324       \tl_head:N \l__spath_tmpa_tl
2325     }
2326     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
2327   }
2328
2329   \tl_put_right:Nx \l__spath_tmpe_tl
2330   {
2331     {\tl_item:Nn \l__spath_tmpa_tl {1}}
2332     {\tl_item:Nn \l__spath_tmpa_tl {2}}
2333   }
2334   \tl_put_right:NV \l__spath_tmpe_tl \c_spath_moveto_tl
2335
2336   \tl_reverse:N \l__spath_tmpa_tl
2337
2338   \fp_set:Nn \l__spath_tmpa_fp
2339   {
2340     \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {4}}
2341     -
2342     \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {1}}
2343   }
2344
2345   \fp_set:Nn \l__spath_tmpb_fp
2346   {
2347     \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {5}}

```

```

2348 -
2349 \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {2}}
2350 }
2351
2352 \fp_set:Nn \l__spath_tmpc_fp
2353 {
2354   sqrt(
2355     \l__spath_tmpe_fp * \l__spath_tmpe_fp
2356     +
2357     \l__spath_tmpe_fp * \l__spath_tmpe_fp
2358   ) * \l__spath_tmpe_int
2359 }
2360
2361 \fp_compare:nTF
2362 {
2363   \l__spath_tmpc_fp > #2
2364 }
2365 {
2366
2367   \fp_set:Nn \l__spath_tmpe_fp
2368   {
2369     (\l__spath_tmpe_fp - #2)/ \l__spath_tmpe_fp
2370   }
2371
2372   \tl_reverse:N \l__spath_tmpe_tl
2373
2374   \tl_if_eq:NNTF \l__spath_tmpe_tl \c_spath_curveto_tl
2375   {
2376     \spath_split_curve:NNVV
2377     \l__spath_tmpe_tl
2378     \l__spath_tmpe_tl
2379     \l__spath_tmpe_tl
2380     \l__spath_tmpe_fp
2381   }
2382   {
2383     \spath_split_line:NNVV
2384     \l__spath_tmpe_tl
2385     \l__spath_tmpe_tl
2386     \l__spath_tmpe_tl
2387     \l__spath_tmpe_fp
2388   }
2389
2390   \prg_replicate:nn {3}
2391   {
2392     \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl}
2393   }
2394
2395   \tl_put_right:NV \l__spath_tmpe_tl \l__spath_tmpe_tl
2396
2397 }
2398 {
2399
2400   \int_compare:nT
2401   {

```

```

2402         \tl_count:N \l__spath_tmpa_tl > 3
2403     }
2404     {
2405         \dim_set:Nn \l__spath_tmpa_dim {\fp_to_dim:n {#2 - \l__spath_tmpe_fp } }
2406         \spath_shorten_at_end:NV \l__spath_tmpa_tl \l__spath_tmpa_dim
2407     }
2408 }
2409
2410 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
2411 \group_end:
2412 }
2413 {
2414     \tl_gset:Nn \g__spath_output_tl {#1}
2415 }
2416 }
2417 \cs_new_protected_nopar:Npn \spath_shorten_at_end:Nnn #1#2#3
2418 {
2419     \__spath_shorten_at_end:nn {#2}{#3}
2420     \tl_set_eq:NN #1 \g__spath_output_tl
2421     \tl_gclear:N \g__spath_output_tl
2422 }
2423 \cs_generate_variant:Nn \spath_shorten_at_end:Nnn {NVV, cnn, cVV, NVn}
2424 \cs_new_protected_nopar:Npn \spath_shorten_at_end:Nn #1#2
2425 {
2426     \spath_shorten_at_end:NVn #1#1{#2}
2427 }
2428 \cs_generate_variant:Nn \spath_shorten_at_end:Nn {cn, cV, NV}
2429 \cs_new_protected_nopar:Npn \spath_gshorten_at_end:Nnn #1#2#3
2430 {
2431     \__spath_shorten_at_end:nn {#2}{#3}
2432     \tl_gset_eq:NN #1 \g__spath_output_tl
2433     \tl_gclear:N \g__spath_output_tl
2434 }
2435 \cs_generate_variant:Nn \spath_gshorten_at_end:Nnn {NVV, cnn, cVV, NVn}
2436 \cs_new_protected_nopar:Npn \spath_gshorten_at_end:Nn #1#2
2437 {
2438     \spath_gshorten_at_end:NVn #1#1{#2}
2439 }
2440 \cs_generate_variant:Nn \spath_gshorten_at_end:Nn {cn, cV, NV}

```

(End definition for \spath_shorten_at_end:Nnn.)

\spath_shorten_at_start:Nnn
\spath_shorten_at_start:Nn
\spath_gshorten_at_start:Nnn
\spath_gshorten_at_start:Nn

This macro shortens a path from the start by a dimension.

```

2441 \cs_new_protected_nopar:Npn \__spath_shorten_at_start:nn #1#2
2442 {
2443     \int_compare:nTF
2444     {
2445         \tl_count:n {#1} > 3
2446     }
2447     {
2448         \group_begin:
2449         \tl_set:Nn \l__spath_tmpa_tl {#1}
2450
2451         \tl_set:Nx \l__spath_tmpe_tl {\tl_item:Nn \l__spath_tmpa_tl {4}}

```

```

2452
2453     \tl_clear:N \l__spath_tmpe_tl
2454
2455 \tl_if_eq:NNTF \l__spath_tmpe_tl \c_spath_curvetoa_tl
2456 {
2457     \int_set:Nn \l__spath_tmpe_int {3}
2458 }
2459 {
2460     \int_set:Nn \l__spath_tmpe_int {1}
2461 }
2462
2463 \tl_set_eq:NN \l__spath_tmpe_tl \c_spath_moveto_tl
2464 \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2465
2466 \prg_replicate:nn { \l__spath_tmpe_int }
2467 {
2468     \__spath_tl_put_right_braced:Nx
2469     \l__spath_tmpe_tl
2470     {\tl_item:Nn \l__spath_tmpe_tl {1}}
2471     \__spath_tl_put_right_braced:Nx
2472     \l__spath_tmpe_tl
2473     {\tl_item:Nn \l__spath_tmpe_tl {2}}
2474     \tl_put_right:Nx \l__spath_tmpe_tl {\tl_item:Nn \l__spath_tmpe_tl {3}}
2475
2476     \prg_replicate:nn {3}
2477     {
2478         \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2479     }
2480 }
2481 \__spath_tl_put_right_braced:Nx
2482 \l__spath_tmpe_tl
2483 {\tl_item:Nn \l__spath_tmpe_tl {1}}
2484 \__spath_tl_put_right_braced:Nx
2485 \l__spath_tmpe_tl
2486 {\tl_item:Nn \l__spath_tmpe_tl {2}}
2487
2488 \fp_set:Nn \l__spath_tmpe_fp
2489 {
2490     \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {5}}
2491     -
2492     \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {2}}
2493 }
2494
2495 \fp_set:Nn \l__spath_tmpe_fp
2496 {
2497     \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {6}}
2498     -
2499     \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {3}}
2500 }
2501
2502 \fp_set:Nn \l__spath_tmpe_fp
2503 {
2504     sqrt(
2505     \l__spath_tmpe_fp * \l__spath_tmpe_fp

```

```

2506     +
2507     \l__spath_tmpb_fp * \l__spath_tmpb_fp
2508   )
2509   *
2510   \l__spath_tmpa_int
2511 }
2512
2513 \fp_compare:nTF
2514 {
2515   \l__spath_tmpe_fp > #2
2516 }
2517 {
2518
2519   \fp_set:Nn \l__spath_tmpe_fp
2520   {
2521     #2/ \l__spath_tmpe_fp
2522   }
2523
2524   \tl_if_eq:NNTF \l__spath_tmpe_tl \c_spath_curvetoa_tl
2525   {
2526     \spath_split_curve:NNVV
2527     \l__spath_tmpe_tl
2528     \l__spath_tmpe_tl
2529     \l__spath_tmpe_tl
2530     \l__spath_tmpe_fp
2531   }
2532   {
2533     \spath_split_line:NNVV
2534     \l__spath_tmpe_tl
2535     \l__spath_tmpe_tl
2536     \l__spath_tmpe_tl
2537     \l__spath_tmpe_fp
2538   }
2539
2540   \prg_replicate:nn {2}
2541   {
2542     \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl}
2543   }
2544
2545   \tl_put_left:NV \l__spath_tmpe_tl \l__spath_tmpe_tl
2546 }
2547 {
2548 {
2549
2550   \tl_put_left:NV \l__spath_tmpe_tl \c_spath_movetoa_tl
2551
2552   \int_compare:nT
2553   {
2554     \tl_count:N \l__spath_tmpe_tl > 3
2555   }
2556   {
2557     \dim_set:Nn \l__spath_tmpe_dim {\fp_to_dim:n {#2 - \l__spath_tmpe_fp } }
2558     \spath_shorten_at_start:NV \l__spath_tmpe_tl \l__spath_tmpe_dim
2559   }

```

```

2560 }
2561
2562 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
2563 \group_end:
2564 }
2565 {
2566   \tl_gset:Nn \g__spath_output_tl {#1}
2567 }
2568 }
2569 \cs_new_protected_nopar:Npn \spath_shorten_at_start:Nnn #1#2#3
2570 {
2571   \__spath_shorten_at_start:nn {#2}{#3}
2572   \tl_set_eq:NN #1 \g__spath_output_tl
2573   \tl_gclear:N \g__spath_output_tl
2574 }
2575 \cs_generate_variant:Nn \spath_shorten_at_start:Nnn {NVV, cnn, cVV, NVn}
2576 \cs_new_protected_nopar:Npn \spath_shorten_at_start:Nn #1#2
2577 {
2578   \spath_shorten_at_start:NVn #1#1{#2}
2579 }
2580 \cs_generate_variant:Nn \spath_shorten_at_start:Nn {cn, cV, NV}
2581 \cs_new_protected_nopar:Npn \spath_gshorten_at_start:Nnn #1#2#3
2582 {
2583   \__spath_shorten_at_start:nn {#2}{#3}
2584   \tl_gset_eq:NN #1 \g__spath_output_tl
2585   \tl_gclear:N \g__spath_output_tl
2586 }
2587 \cs_generate_variant:Nn \spath_gshorten_at_start:Nnn {NVV, cnn, cVV, NVn}
2588 \cs_new_protected_nopar:Npn \spath_gshorten_at_start:Nn #1#2
2589 {
2590   \spath_gshorten_at_start:NVn #1#1{#2}
2591 }
2592 \cs_generate_variant:Nn \spath_gshorten_at_start:Nn {cn, cV, NV}

```

(End definition for `\spath_shorten_at_start:Nnn` and others.)

3.6 Points on a Path

`\spath_point_at:Nnn` Get the location of a point on a path, using the same location specification as the intersection library.
`\spath_gpoint_at:Nnn`

```

2593 \cs_new_protected_nopar:Npn \__spath_point_at:nn #1#2
2594 {
2595   \group_begin:
2596   \int_set:Nn \l__spath_tmpa_int {\fp_to_int:n {floor(#2) + 1}}
2597   \fp_set:Nn \l__spath_tmpa_fp {#2 - floor(#2)}
2598
2599   \spath_segments_to_seq:Nn \l__spath_tmpa_seq {#1}
2600
2601   \int_compare:nTF
2602   {
2603     \l__spath_tmpa_int < 1
2604   }
2605   {
2606     \spath_initialpoint:Nn \l__spath_tmpc_tl {#1}

```

```

2607 }
2608 {
2609   \int_compare:nTF
2610   {
2611     \l__spath_tmpa_int > \seq_count:N \l__spath_tmpa_seq
2612   }
2613   {
2614     \spath_finalpoint:Nn \l__spath_tmpc_tl {#1}
2615   }
2616   {
2617
2618     \tl_set:Nx
2619     \l__spath_tmpa_tl
2620     {\seq_item:Nn \l__spath_tmpa_seq { \l__spath_tmpa_int} }
2621
2622     \int_compare:nTF
2623     {
2624       \tl_count:N \l__spath_tmpa_tl > 3
2625     }
2626     {
2627       \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {4}}
2628     }
2629     {
2630       \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {1}}
2631     }
2632
2633     \tl_clear:N \l__spath_tmpc_tl
2634
2635     \tl_case:Nn \l__spath_tmpb_tl
2636     {
2637       \c_spath_moveto_tl
2638       {
2639         \tl_set:Nx \l__spath_tmpc_tl
2640         {
2641           {
2642             \tl_item:Nn \l__spath_tmpa_tl {2}
2643           }
2644           {
2645             \tl_item:Nn \l__spath_tmpa_tl {3}
2646           }
2647         }
2648       }
2649
2650       \c_spath_lineto_tl
2651       {
2652         \tl_set:Nx \l__spath_tmpc_tl
2653         {
2654           {\fp_to_dim:n
2655           {
2656             (1 - \l__spath_tmpa_fp) * ( \tl_item:Nn \l__spath_tmpa_tl {2} )
2657             +
2658             \l__spath_tmpa_fp * ( \tl_item:Nn \l__spath_tmpa_tl {5} )
2659           }
2660         }

```

```

2661         {\fp_to_dim:n
2662         {
2663             (1 - \l__spath_tmpa_fp) * ( \tl_item:Nn \l__spath_tmpa_tl {3} )
2664             +
2665             \l__spath_tmpa_fp * ( \tl_item:Nn \l__spath_tmpa_tl {6} )
2666         }
2667     }
2668 }
2669 }
2670
2671 \c_spath_closepath_tl
2672 {
2673     \tl_set:Nx \l__spath_tmpe_tl
2674     {
2675         {\fp_to_dim:n
2676         {
2677             (1 - \l__spath_tmpa_fp) * ( \tl_item:Nn \l__spath_tmpa_tl {2} )
2678             +
2679             \l__spath_tmpa_fp * ( \tl_item:Nn \l__spath_tmpa_tl {5} )
2680         }
2681     }
2682     {\fp_to_dim:n
2683     {
2684         (1 - \l__spath_tmpa_fp) * ( \tl_item:Nn \l__spath_tmpa_tl {3} )
2685         +
2686         \l__spath_tmpa_fp * ( \tl_item:Nn \l__spath_tmpa_tl {6} )
2687     }
2688 }
2689 }
2690 }
2691
2692 \c_spath_curvetoa_tl
2693 {
2694     \tl_set:Nx \l__spath_tmpe_tl
2695     {
2696         {\fp_to_dim:n
2697         {
2698             (1 - \l__spath_tmpa_fp)^3 * \tl_item:Nn \l__spath_tmpa_tl {2}
2699             + 3 * (1 - \l__spath_tmpa_fp)^2 * (\l__spath_tmpa_fp)
2700             * \tl_item:Nn \l__spath_tmpa_tl {5}
2701             + 3 * (1 - \l__spath_tmpa_fp) * (\l__spath_tmpa_fp)^2
2702             * \tl_item:Nn \l__spath_tmpa_tl {8}
2703             + (\l__spath_tmpa_fp)^3 * \tl_item:Nn \l__spath_tmpa_tl {11}
2704         }}
2705         {\fp_to_dim:n
2706         {
2707             (1 - \l__spath_tmpa_fp)^3 * \tl_item:Nn \l__spath_tmpa_tl {3}
2708             + 3 * (1 - \l__spath_tmpa_fp)^2 * (\l__spath_tmpa_fp)
2709             * \tl_item:Nn \l__spath_tmpa_tl {6}
2710             + 3 * (1 - \l__spath_tmpa_fp) * (\l__spath_tmpa_fp)^2
2711             * \tl_item:Nn \l__spath_tmpa_tl {9}
2712             + (\l__spath_tmpa_fp)^3 * \tl_item:Nn \l__spath_tmpa_tl {12}
2713         }}
2714     }

```



```

2715     }
2716   }
2717 }
2718 }
2719
2720 \tl_gclear:N \g__spath_output_tl
2721 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpc_tl
2722 \group_end:
2723 }
2724 \cs_new_protected_nopar:Npn \spath_point_at:Nnn #1#2#3
2725 {
2726   \__spath_point_at:nn {#2}{#3}
2727   \tl_set_eq:NN #1 \g__spath_output_tl
2728   \tl_gclear:N \g__spath_output_tl
2729 }
2730 \cs_generate_variant:Nn \spath_point_at:Nnn {NVn, NVV, NnV}
2731 \cs_new_protected_nopar:Npn \spath_gpoint_at:Nnn #1#2#3
2732 {
2733   \__spath_point_at:nn {#2}{#3}
2734   \tl_gset_eq:NN #1 \g__spath_output_tl
2735   \tl_gclear:N \g__spath_output_tl
2736 }
2737 \cs_generate_variant:Nn \spath_gpoint_at:Nnn {NVn, NVV, NnV}

```

(End definition for \spath_point_at:Nnn and \spath_gpoint_at:Nnn.)

\spath_tangent_at:Nnn Get the tangent at a point on a path, using the same location specification as the intersection library.

```

2738 \cs_new_protected_nopar:Npn \__spath_tangent_at:nn #1#2
2739 {
2740   \group_begin:
2741   \int_set:Nn \l__spath_tmpa_int {\fp_to_int:n {floor(#2) + 1}}
2742   \fp_set:Nn \l__spath_tmpa_fp {#2 - floor(#2)}
2743
2744   \spath_segments_to_seq:Nn \l__spath_tmpa_seq {#1}
2745
2746   \int_compare:nTF
2747   {
2748     \l__spath_tmpa_int < 1
2749   }
2750   {
2751     \spath_initialpoint:Nn \l__spath_tmpc_tl {#1}
2752   }
2753   {
2754     \int_compare:nTF
2755     {
2756       \l__spath_tmpa_int > \seq_count:N \l__spath_tmpa_seq
2757     }
2758     {
2759       \spath_finalpoint:Nn \l__spath_tmpc_tl {#1}
2760     }
2761     {
2762
2763       \tl_set:Nx

```

```

2764 \l__spath_tmpa_tl
2765 {\seq_item:Nn \l__spath_tmpa_seq { \l__spath_tmpa_int} }
2766
2767 \int_compare:nTF
2768 {
2769   \tl_count:N \l__spath_tmpa_tl > 3
2770 }
2771 {
2772   \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {4}}
2773 }
2774 {
2775   \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {1}}
2776 }
2777
2778 \tl_clear:N \l__spath_tmpc_tl
2779
2780 \tl_case:Nn \l__spath_tmpb_tl
2781 {
2782   \c_spath_moveto_tl
2783   {
2784     \tl_set:Nx \l__spath_tmpc_tl
2785     {
2786       {
2787         \tl_item:Nn \l__spath_tmpa_tl {2}
2788       }
2789       {
2790         \tl_item:Nn \l__spath_tmpa_tl {3}
2791       }
2792     }
2793   }
2794
2795   \c_spath_lineto_tl
2796   {
2797     \tl_set:Nx \l__spath_tmpc_tl
2798     {
2799       {\fp_to_dim:n
2800         {
2801           ( \tl_item:Nn \l__spath_tmpa_tl {5} )
2802           -
2803           ( \tl_item:Nn \l__spath_tmpa_tl {2} )
2804         }
2805       }
2806       {\fp_to_dim:n
2807         {
2808           ( \tl_item:Nn \l__spath_tmpa_tl {6} )
2809           -
2810           ( \tl_item:Nn \l__spath_tmpa_tl {3} )
2811         }
2812       }
2813     }
2814
2815   \c_spath_closepath_tl
2816   {
2817

```

```

2818 \tl_set:Nx \l__spath_tmpc_tl
2819 {
2820   {\fp_to_dim:n
2821     {
2822       ( \tl_item:Nn \l__spath_tmpa_tl {5} )
2823       -
2824       ( \tl_item:Nn \l__spath_tmpa_tl {2} )
2825     }
2826   }
2827   {\fp_to_dim:n
2828     {
2829       ( \tl_item:Nn \l__spath_tmpa_tl {6} )
2830       -
2831       ( \tl_item:Nn \l__spath_tmpa_tl {3} )
2832     }
2833   }
2834 }
2835
2836
2837 \c_spath_curvetoa_tl
2838 {
2839   \tl_set:Nx \l__spath_tmpc_tl
2840   {
2841     {\fp_to_dim:n
2842       {
2843         3*(1 - \l__spath_tmpa_fp)^2 * (\tl_item:Nn \l__spath_tmpa_tl {5}
2844         - \tl_item:Nn \l__spath_tmpa_tl {2})
2845         + 6 * (1 - \l__spath_tmpa_fp) * (\l__spath_tmpa_fp) *
2846         (\tl_item:Nn \l__spath_tmpa_tl {8}
2847         - \tl_item:Nn \l__spath_tmpa_tl {5})
2848         + 3*(\l__spath_tmpa_fp)^2 * (\tl_item:Nn \l__spath_tmpa_tl {11}
2849         - \tl_item:Nn \l__spath_tmpa_tl {8})
2850       }
2851     }
2852     {\fp_to_dim:n
2853       {
2854         3*(1 - \l__spath_tmpa_fp)^2 * (\tl_item:Nn \l__spath_tmpa_tl {6}
2855         - \tl_item:Nn \l__spath_tmpa_tl {3})
2856         + 6 * (1 - \l__spath_tmpa_fp) * (\l__spath_tmpa_fp) *
2857         (\tl_item:Nn \l__spath_tmpa_tl {9}
2858         - \tl_item:Nn \l__spath_tmpa_tl {6})
2859         + 3*(\l__spath_tmpa_fp)^2 * (\tl_item:Nn \l__spath_tmpa_tl {12}
2860         - \tl_item:Nn \l__spath_tmpa_tl {9})
2861       }}
2862     }
2863   }
2864 }
2865
2866 }
2867
2868 \tl_gclear:N \g__spath_output_tl
2869 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpc_tl
2870 \group_end:
2871 }

```

```

2872 \cs_new_protected_nopar:Npn \spath_tangent_at:Nnn #1#2#3
2873 {
2874   \__spath_tangent_at:nn {#2}{#3}
2875   \tl_set_eq:NN #1 \g__spath_output_tl
2876   \tl_gclear:N \g__spath_output_tl
2877 }
2878 \cs_generate_variant:Nn \spath_tangent_at:Nnn {NVn, NVV, NnV}
2879 \cs_new_protected_nopar:Npn \spath_gtangent_at:Nnn #1#2#3
2880 {
2881   \__spath_tangent_at:nn {#2}{#3}
2882   \tl_gset_eq:NN #1 \g__spath_output_tl
2883   \tl_gclear:N \g__spath_output_tl
2884 }
2885 \cs_generate_variant:Nn \spath_gtangent_at:Nnn {NVn, NVV, NnV}

```

(End definition for \spath_tangent_at:Nnn and \spath_gtangent_at:Nnn.)

\spath_transformation_at:Nnn Gets a transformation that will align to a point on the path with the x-axis along the path.

```

2886 \cs_new_protected_nopar:Npn \__spath_transformation_at:nn #1#2
2887 {
2888   \group_begin:
2889   \tl_clear:N \l__spath_tmpa_tl
2890   \__spath_tangent_at:nn {#1}{#2}
2891   \tl_set_eq:NN \l__spath_tmpb_tl \g__spath_output_tl
2892   \fp_set:Nn \l__spath_tmpa_fp
2893   {
2894     sqrt(
2895       (\tl_item:Nn \l__spath_tmpb_tl {1})^2
2896       +
2897       (\tl_item:Nn \l__spath_tmpb_tl {2})^2
2898     )
2899   }
2900   \fp_compare:nTF {\l__spath_tmpa_fp = 0}
2901   {
2902     \fp_set:Nn \l__spath_tmpa_fp {1}
2903     \fp_set:Nn \l__spath_tmpb_fp {0}
2904   }
2905   {
2906     \fp_set:Nn \l__spath_tmpb_fp
2907     { (\tl_item:Nn \l__spath_tmpb_tl {2}) / \l__spath_tmpa_fp }
2908     \fp_set:Nn \l__spath_tmpa_fp
2909     { (\tl_item:Nn \l__spath_tmpb_tl {1}) / \l__spath_tmpa_fp }
2910   }
2911   \tl_set:Nx \l__spath_tmpa_tl
2912   {
2913     { \fp_to_decimal:n { \l__spath_tmpa_fp } }
2914     { \fp_to_decimal:n { \l__spath_tmpb_fp } }
2915     { \fp_to_decimal:n {- \l__spath_tmpb_fp } }
2916     { \fp_to_decimal:n { \l__spath_tmpa_fp } }
2917   }
2918   \__spath_point_at:nn {#1}{#2}
2919   \tl_put_right:NV \l__spath_tmpa_tl \g__spath_output_tl
2920   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl

```

```

2921 \group_end:
2922 }
2923 \cs_new_protected_nopar:Npn \spath_transformation_at:Nnn #1#2#3
2924 {
2925   \__spath_transformation_at:nn {#2}{#3}
2926   \tl_set_eq:NN #1 \g__spath_output_tl
2927   \tl_gclear:N \g__spath_output_tl
2928 }
2929 \cs_generate_variant:Nn \spath_transformation_at:Nnn {NVn, NVV, NnV, NvV}
2930 \cs_new_protected_nopar:Npn \spath_gtransformation_at:Nnn #1#2#3
2931 {
2932   \__spath_transformation_at:nn {#2}{#3}
2933   \tl_gset_eq:NN #1 \g__spath_output_tl
2934   \tl_gclear:N \g__spath_output_tl
2935 }
2936 \cs_generate_variant:Nn \spath_gtransformation_at:Nnn {NVn, NVV, NnV}

```

(End definition for `\spath_transformation_at:Nnn` and `\spath_gtransformation_at:Nnn`.)

3.7 Intersection Routines

Note: I'm not consistent with number schemes. The intersection library is 0-based, but the user interface is 1-based (since if we “count” in a `\foreach` then it starts at 1). This should be more consistent.

```

\spath_intersect:NN Pass two spaths to pgf's intersection routine.
\spath_intersect:nn
2937 \cs_new_protected_nopar:Npn \spath_intersect:NN #1#2
2938 {
2939   \pgfintersectionofpaths%
2940   {%
2941     \pgfsetpath #1
2942   }{%
2943     \pgfsetpath #2
2944   }
2945 }
2946 \cs_new_protected_nopar:Npn \spath_intersect:nn #1#2
2947 {
2948   \tl_set:Nn \l__spath_intersecta_tl {#1}
2949   \tl_set:Nn \l__spath_intersectb_tl {#2}
2950   \spath_intersect:NN \l__spath_intersecta_tl \l__spath_intersectb_tl
2951 }

```

(End definition for `\spath_intersect:NN` and `\spath_intersect:nn`.)

`\spath_split_component_at_intersections:Nnn` Split a component where it intersects a path. Key assumption is that the first path is a single component, so if it is closed then the end joins up to the beginning. The component is modified but the path is not.

```

2952 \cs_new_protected_nopar:Npn \__spath_split_component_at_intersections:nn #1#2
2953 {
2954   \group_begin:
2955   \tl_clear:N \l__spath_tmpe_tl
2956   \seq_clear:N \l__spath_tmpb_seq
2957
2958   % Find the intersections of these segments

```

```

2959 \tl_set:Nn \l__spath_tmpb_tl {#1}
2960 \tl_set:Nn \l__spath_tmpc_tl {#2}
2961
2962 \spath_reallength:Nn \l__spath_tmpa_int {#1}
2963
2964 % Remember if the component is closed
2965 \spath_finalaction:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
2966
2967 \bool_set:Nn \l__spath_closed_bool
2968 {
2969   \tl_if_eq_p:NN \l__spath_tmpa_tl \c_spath_closepath_tl
2970 }
2971
2972 % Open it
2973 \spath_open:N \l__spath_tmpb_tl
2974
2975 % Sort intersections along the component
2976 \pgfintersectionsorthbyfirstpath
2977 \spath_intersect:NN \l__spath_tmpb_tl \l__spath_tmpc_tl
2978
2979 % If we get intersections
2980 \int_compare:nT {\pgfintersectionsolutions > 0}
2981 {
2982   % Find the times of the intersections on the component
2983   \int_step_inline:nnnn {1} {1} {\pgfintersectionsolutions}
2984   {
2985     \pgfintersectiongetsolutiontimes{##1}{\l__spath_tmph_tl}{\l__spath_tmpl_tl}
2986     \seq_put_left:NV \l__spath_tmpb_seq \l__spath_tmph_tl
2987   }
2988
2989   \seq_get_left:NN \l__spath_tmpb_seq \l__spath_tmpa_tl
2990   \fp_compare:nT
2991   {
2992     \l__spath_tmpa_tl > \l__spath_tmpa_int - .01
2993   }
2994   {
2995     \bool_set_false:N \l__spath_closed_bool
2996   }
2997   \seq_get_right:NN \l__spath_tmpb_seq \l__spath_tmpa_tl
2998   \fp_compare:nT
2999   {
3000     \l__spath_tmpa_tl < .01
3001   }
3002   {
3003     \bool_set_false:N \l__spath_closed_bool
3004   }
3005
3006
3007 \tl_set:Nn \l__spath_tmpg_tl {-1}
3008
3009 \seq_map_inline:Nn \l__spath_tmpb_seq
3010 {
3011   \tl_set:Nn \l__spath_tmph_tl {##1}
3012

```

```

3013 \tl_set_eq:NN \l__spath_tmpe_tl \l__spath_tmph_tl
3014 \int_compare:nT
3015 {
3016   \fp_to_int:n {floor( \l__spath_tmph_tl ) }
3017   =
3018   \fp_to_int:n {floor( \l__spath_tmpg_tl ) }
3019 }
3020 {
3021   \tl_set:Nx \l__spath_tmph_tl
3022   {
3023     \fp_eval:n {
3024       floor( \l__spath_tmph_tl )
3025       +
3026       ( \l__spath_tmph_tl - floor( \l__spath_tmph_tl ) )
3027       /
3028       ( \l__spath_tmpg_tl - floor( \l__spath_tmpg_tl ) )
3029     }
3030   }
3031 }
3032 \tl_set_eq:NN \l__spath_tmpg_tl \l__spath_tmpe_tl
3033
3034 \spath_split_at:NNVV
3035 \l__spath_tmpd_tl
3036 \l__spath_tmpe_tl
3037 \l__spath_tmph_tl
3038 \l__spath_tmpe_tl
3039
3040 \tl_put_left:NV \l__spath_tmpe_tl \l__spath_tmpe_tl
3041 \tl_set_eq:NN \l__spath_tmpe_tl \l__spath_tmpe_tl
3042
3043 }
3044
3045
3046 \tl_put_left:NV \l__spath_tmpe_tl \l__spath_tmpe_tl
3047
3048 \spath_remove_empty_components:N \l__spath_tmpe_tl
3049
3050 \tl_set_eq:NN \l__spath_tmpe_tl \l__spath_tmpe_tl
3051 }
3052
3053 \bool_if:NT \l__spath_closed_bool
3054 {
3055   \spath_join_component:Nn \l__spath_tmpe_tl {1}
3056 }
3057
3058 \tl_gclear:N \g__spath_output_tl
3059 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpe_tl
3060
3061 \group_end:
3062 }
3063 \cs_new_protected_nopar:Npn \spath_split_component_at_intersections:Nnn #1#2#3
3064 {
3065   \__spath_split_component_at_intersections:nn {#2}{#3}
3066   \tl_set_eq:NN #1 \g__spath_output_tl

```

```

3067 \tl_gclear:N \g__spath_output_tl
3068 }
3069 \cs_generate_variant:Nn \spath_split_component_at_intersections:Nnn {NVn, NVV}
3070 \cs_new_protected_nopar:Npn \spath_split_component_at_intersections:Nn #1#2
3071 {
3072   \spath_split_component_at_intersections:NVn #1#1{#2}
3073 }
3074 \cs_generate_variant:Nn \spath_split_component_at_intersections:Nn {cn, cv}
3075 \cs_new_protected_nopar:Npn \spath_gsplit_component_at_intersections:Nnn #1#2#3
3076 {
3077   \__spath_split_component_at_intersections:nn {#2}{#3}
3078   \tl_gset_eq:NN #1 \g__spath_output_tl
3079   \tl_gclear:N \g__spath_output_tl
3080 }
3081 \cs_generate_variant:Nn \spath_gsplit_component_at_intersections:Nnn {NVn, NVV}
3082 \cs_new_protected_nopar:Npn \spath_gsplit_component_at_intersections:Nn #1#2
3083 {
3084   \spath_gsplit_component_at_intersections:NVn #1#1{#2}
3085 }
3086 \cs_generate_variant:Nn \spath_gsplit_component_at_intersections:Nn {cn, cv}

```

(End definition for \spath_split_component_at_intersections:Nnn.)

Split paths at their intersections. The path versions only split the first path. The others split both paths.

```

\spath_split_path_at_intersections:Nnn
\spath_split_path_at_intersections:Nn
\spath_gsplit_path_at_intersections:Nnn
\spath_gsplit_path_at_intersections:Nn
\spath_split_at_intersections:NNnn
\spath_split_at_intersections:NN
\spath_gsplit_at_intersections:NNnn
\spath_gsplit_at_intersections:NN

```

```

3087 \cs_new_protected_nopar:Npn \__spath_split_path_at_intersections:nn #1#2
3088 {
3089   \group_begin:
3090
3091   \seq_clear:N \l__spath_tmpa_seq
3092   \seq_clear:N \l__spath_tmpb_seq
3093
3094   \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
3095   \seq_map_inline:Nn \l__spath_tmpa_seq
3096   {
3097     \spath_split_component_at_intersections:Nnn \l__spath_tmpa_tl {##1} {#2}
3098     \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
3099   }
3100
3101   \tl_gclear:N \g__spath_output_tl
3102   \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpb_seq {} }
3103   \group_end:
3104 }
3105 \cs_new_protected_nopar:Npn \spath_split_path_at_intersections:Nnn #1#2#3
3106 {
3107   \__spath_split_path_at_intersections:nn {#2}{#3}
3108   \tl_set_eq:NN #1 \g__spath_output_tl
3109   \tl_gclear:N \g__spath_output_tl
3110 }
3111 \cs_generate_variant:Nn \spath_split_path_at_intersections:Nnn
3112 {NVn, NVV, cVn, cVV, cvn, cvv}
3113 \cs_new_protected_nopar:Npn \spath_split_path_at_intersections:Nn #1#2
3114 {
3115   \spath_split_path_at_intersections:NVn #1#1{#2}

```



```

3116 }
3117 \cs_generate_variant:Nn \spath_split_path_at_intersections:Nn {cv, NV}
3118 \cs_new_protected_nopar:Npn \spath_gsplit_path_at_intersections:Nnn #1#2#3
3119 {
3120   \__spath_split_path_at_intersections:nn {#2}{#3}
3121   \tl_gset_eq:NN #1 \g__spath_output_tl
3122   \tl_gclear:N \g__spath_output_tl
3123 }
3124 \cs_generate_variant:Nn \spath_gsplit_path_at_intersections:Nnn
3125 {NNVn, NVV, cVn, cVV, cvn, cvv}
3126 \cs_new_protected_nopar:Npn \spath_gsplit_path_at_intersections:Nn #1#2
3127 {
3128   \spath_gsplit_path_at_intersections:NVn #1#1{#2}
3129 }
3130 \cs_generate_variant:Nn \spath_gsplit_path_at_intersections:Nn {cv, NV}
3131 \cs_new_protected_nopar:Npn \spath_split_at_intersections:NNnn #1#2#3#4
3132 {
3133   \__spath_split_path_at_intersections:nn {#3}{#4}
3134   \tl_set_eq:NN #1 \g__spath_output_tl
3135   \__spath_split_path_at_intersections:nn {#4}{#3}
3136   \tl_set_eq:NN #2 \g__spath_output_tl
3137   \tl_gclear:N \g__spath_output_tl
3138 }
3139 \cs_generate_variant:Nn \spath_split_at_intersections:NNnn
3140 {NNVn, NNVV, ccVn, ccVV, ccvn, ccvv}
3141 \cs_new_protected_nopar:Npn \spath_split_at_intersections:NN #1#2
3142 {
3143   \spath_split_at_intersections:NNVV #1#2#1#2
3144 }
3145 \cs_generate_variant:Nn \spath_split_at_intersections:NN {cc}
3146 \cs_new_protected_nopar:Npn \spath_gsplit_at_intersections:NNnn #1#2#3#4
3147 {
3148   \__spath_split_path_at_intersections:nn {#3}{#4}
3149   \tl_gset_eq:NN #1 \g__spath_output_tl
3150   \__spath_split_path_at_intersections:nn {#4}{#3}
3151   \tl_gset_eq:NN #2 \g__spath_output_tl
3152   \tl_gclear:N \g__spath_output_tl
3153 }
3154 \cs_generate_variant:Nn \spath_gsplit_at_intersections:NNnn
3155 {NNVn, NNVV, ccVn, ccVV, ccvn, ccvv}
3156 \cs_new_protected_nopar:Npn \spath_gsplit_at_intersections:NN #1#2
3157 {
3158   \spath_gsplit_at_intersections:NNVV #1#2#1#2
3159 }
3160 \cs_generate_variant:Nn \spath_gsplit_at_intersections:NN {cc}

```

(End definition for \spath_split_path_at_intersections:Nnn and others.)

th_split_component_at_self_intersections:Nn
ath_split_component_at_self_intersections:N
h_gsplit_component_at_self_intersections:Nn
th_gsplit_component_at_self_intersections:N

Given a component of a path, split it at points where it self-intersects.

```

3161 \cs_new_protected_nopar:Npn \__spath_split_component_at_self_intersections:n #1
3162 {
3163   \group_begin:
3164   \tl_set:Nn \l__spath_tmpe_tl {#1}
3165

```

```

3166 % Remember if the component is closed
3167 \spath_finalaction:NV \l__spath_tmpa_tl \l__spath_tmpe_tl
3168
3169 \bool_set:Nn \l__spath_closed_bool
3170 {
3171   \tl_if_eq_p:NN \l__spath_tmpa_tl \c_spath_closepath_tl
3172 }
3173
3174 % Copy the path
3175 \tl_set:Nn \l__spath_tmpe_tl {#1}
3176
3177 % Open the path
3178 \spath_open:N \l__spath_tmpe_tl
3179 % Ensure beziers don't self-intersect
3180 \spath_split_curves:N \l__spath_tmpe_tl
3181
3182 % Make a copy for later
3183 \tl_set_eq:NN \l__spath_tmpg_tl \l__spath_tmpe_tl
3184
3185 % Clear some token lists and sequences
3186 \tl_clear:N \l__spath_tmpd_tl
3187 \seq_clear:N \l__spath_tmpb_seq
3188 \int_zero:N \l__spath_tmpa_int
3189
3190 \pgfintersectionsorthbyfirstpath
3191
3192 % Split the path into a sequence of segments
3193 \spath_segments_to_seq:NV \l__spath_tmpa_seq \l__spath_tmpe_tl
3194
3195 \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
3196 {
3197   \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
3198   {
3199     % Don't intersect a segment with itself
3200     \int_compare:nF
3201     {
3202       ##1 == ####1
3203     }
3204     {
3205       \spath_intersect:nn {##2} {####2}
3206
3207       \int_compare:nT {\pgfintersectionsolutions > 0}
3208       {
3209         % Find the times of the intersections on each path
3210         \int_step_inline:nnnn {1} {1} {\pgfintersectionsolutions}
3211         {
3212           \pgfintersectiongetsolutiontimes
3213           {#####1}{\l__spath_tmpb_tl}{\l__spath_tmpe_tl}
3214
3215           \bool_if:nT
3216           {
3217             !(
3218               \fp_compare_p:n { \l__spath_tmpb_tl > .99 }
3219               &&

```

```

3220         \int_compare_p:n {##1 + 1 == ###1}
3221     )
3222     &&
3223     !(
3224     \fp_compare_p:n { \l__spath_tmpb_tl < .01 }
3225     &&
3226     \int_compare_p:n {##1 - 1 == ###1}
3227     )
3228     &&
3229     !(
3230     \l__spath_closed_bool
3231     &&
3232     \fp_compare_p:n { \l__spath_tmpb_tl < .01 }
3233     &&
3234     \int_compare_p:n {##1 == 1}
3235     &&
3236     \int_compare_p:n {\seq_count:N \l__spath_tmpa_seq == ###1}
3237     )
3238     &&
3239     !(
3240     \l__spath_closed_bool
3241     &&
3242     \fp_compare_p:n { \l__spath_tmpb_tl > .99 }
3243     &&
3244     \int_compare_p:n {####1 == 1}
3245     &&
3246     \int_compare_p:n {\seq_count:N \l__spath_tmpa_seq == ##1}
3247     )
3248 }
3249 {
3250     \tl_set:Nx \l__spath_tmpa_tl
3251     {\fp_to_decimal:n {\l__spath_tmpb_tl + ##1 - 1}}
3252     \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
3253 }
3254 }
3255 }
3256 }
3257 }
3258 }
3259
3260 % Sort the sequence by reverse order along the path
3261 \seq_sort:Nn \l__spath_tmpb_seq
3262 {
3263     \fp_compare:nNnTF { ##1 } < { ##2 }
3264     { \sort_return_swapped: }
3265     { \sort_return_same: }
3266 }
3267
3268 \seq_get_left:NN \l__spath_tmpb_seq \l__spath_tmpa_tl
3269 \fp_compare:nT
3270 {
3271     \l__spath_tmpa_tl > \seq_count:N \l__spath_tmpa_seq - .01
3272 }
3273 {

```

```

3274     \bool_set_false:N \l__spath_closed_bool
3275 }
3276 \seq_get_right:NN \l__spath_tmpb_seq \l__spath_tmpa_tl
3277 \fp_compare:nT
3278 {
3279     \l__spath_tmpa_tl < .01
3280 }
3281 {
3282     \bool_set_false:N \l__spath_closed_bool
3283 }
3284
3285 % Restore the original copy of the path
3286 \tl_set_eq:NN \l__spath_tmpe_tl \l__spath_tmpg_tl
3287
3288 % Clear the token lists
3289 \tl_clear:N \l__spath_tmpf_tl
3290 \tl_clear:N \l__spath_tmph_tl
3291 \tl_clear:N \l__spath_tmpg_tl
3292
3293 \tl_set:Nn \l__spath_tmpl_tl {-1}
3294
3295 \seq_map_inline:Nn \l__spath_tmpb_seq
3296 {
3297     \tl_set:Nn \l__spath_tmpb_tl {##1}
3298     \tl_set_eq:NN \l__spath_tmpa_tl \l__spath_tmpb_tl
3299     \int_compare:nT
3300     {
3301         \fp_to_int:n {floor( \l__spath_tmpb_tl ) }
3302         =
3303         \fp_to_int:n {floor( \l__spath_tmpl_tl ) }
3304     }
3305     {
3306         \tl_set:Nx \l__spath_tmpb_tl
3307         {
3308             \fp_eval:n {
3309                 floor( \l__spath_tmpb_tl )
3310                 +
3311                 ( \l__spath_tmpb_tl - floor( \l__spath_tmpb_tl ) )
3312                 /
3313                 ( \l__spath_tmpl_tl - floor( \l__spath_tmpl_tl ) )
3314             }
3315         }
3316     }
3317     \tl_set_eq:NN \l__spath_tmpl_tl \l__spath_tmpa_tl
3318
3319     \spath_split_at:NNVV
3320     \l__spath_tmpf_tl
3321     \l__spath_tmph_tl
3322     \l__spath_tmpe_tl
3323     \l__spath_tmpb_tl
3324
3325     \tl_put_left:NV \l__spath_tmpg_tl \l__spath_tmph_tl
3326     \tl_set_eq:NN \l__spath_tmpe_tl \l__spath_tmpf_tl
3327

```

```

3328 }
3329
3330 \tl_put_left:NV \l__spath_tmpg_tl \l__spath_tmpe_tl
3331
3332 \tl_if_empty:NT \l__spath_tmpg_tl
3333 {
3334   \tl_set_eq:NN \l__spath_tmpg_tl \l__spath_tmpe_tl
3335 }
3336
3337 \spath_remove_empty_components:N \l__spath_tmpg_tl
3338
3339 % Do something with closed
3340 \bool_if:NT \l__spath_closed_bool
3341 {
3342   \spath_join_component:Nn \l__spath_tmpg_tl {1}
3343 }
3344
3345 \tl_gclear:N \g__spath_output_tl
3346 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpg_tl
3347 \group_end:
3348 }
3349 \cs_new_protected_nopar:Npn \spath_split_component_at_self_intersections:Nn #1#2
3350 {
3351   \__spath_split_component_at_self_intersections:n {#2}
3352   \tl_set_eq:NN #1 \g__spath_output_tl
3353   \tl_gclear:N \g__spath_output_tl
3354 }
3355 \cs_generate_variant:Nn \spath_split_component_at_self_intersections:Nn {NV}
3356 \cs_new_protected_nopar:Npn \spath_split_component_at_self_intersections:N #1
3357 {
3358   \spath_split_component_at_self_intersections:NV #1#1
3359 }
3360 \cs_generate_variant:Nn \spath_split_component_at_self_intersections:N {c}
3361 \cs_new_protected_nopar:Npn \spath_gsplit_component_at_self_intersections:Nn #1#2
3362 {
3363   \__spath_split_component_at_self_intersections:n {#2}
3364   \tl_gset_eq:NN #1 \g__spath_output_tl
3365   \tl_gclear:N \g__spath_output_tl
3366 }
3367 \cs_generate_variant:Nn \spath_gsplit_component_at_self_intersections:Nn {NV}
3368 \cs_new_protected_nopar:Npn \spath_gsplit_component_at_self_intersections:N #1
3369 {
3370   \spath_gsplit_component_at_self_intersections:NV #1#1
3371 }
3372 \cs_generate_variant:Nn \spath_gsplit_component_at_self_intersections:N {c}

```

(End definition for `\spath_split_component_at_self_intersections:Nn` and others.)

`\spath_split_at_self_intersections:Nn` Split a path at its self intersections. We iterate over the components, splitting each where it meets all the others and itself. To make this more efficient, we split against the components of the original path rather than updating each time.

```

\spath_split_at_self_intersections:N
\spath_split_at_self_intersections:N
\spath_gsplit_at_self_intersections:Nn
\spath_gsplit_at_self_intersections:N
3373 \cs_new_protected_nopar:Npn \__spath_split_at_self_intersections:n #1
3374 {
3375   \group_begin:

```

```

3376 \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
3377 \seq_clear:N \l__spath_tmpb_seq
3378 \seq_clear:N \l__spath_tmpc_seq
3379
3380 % Iterate over the components of the original path.
3381 \bool_do_until:nn
3382 {
3383   \seq_if_empty_p:N \l__spath_tmpa_seq
3384 }
3385 {
3386   % Get the next component
3387   \seq_pop_left:NN \l__spath_tmpa_seq \l__spath_tmpa_tl
3388   % Copy for later
3389   \tl_set_eq:NN \l__spath_tmpc_tl \l__spath_tmpa_tl
3390   \int_compare:nT
3391   {
3392     \tl_count:N \l__spath_tmpa_tl > 3
3393   }
3394   {
3395
3396     % Split against itself
3397     \spath_split_component_at_self_intersections:N \l__spath_tmpa_tl
3398     % Grab the rest of the path
3399     \tl_set:Nx \l__spath_tmpb_tl
3400     {
3401       \seq_use:Nn \l__spath_tmpb_seq {}
3402       \seq_use:Nn \l__spath_tmpa_seq {}
3403     }
3404     % Split against the rest of the path
3405     \spath_split_path_at_intersections:N \l__spath_tmpa_tl \l__spath_tmpb_tl
3406   }
3407   % Save the split path
3408   \seq_put_right:N \l__spath_tmpc_seq \l__spath_tmpa_tl
3409   % Add the original copy to the sequence of processed components
3410   \seq_put_right:N \l__spath_tmpb_seq \l__spath_tmpc_tl
3411 }
3412
3413 \tl_gclear:N \g__spath_output_tl
3414 \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpc_seq {} }
3415 \group_end:
3416 }
3417 \cs_generate_variant:Nn \__spath_split_at_self_intersections:n {V, v}
3418 \cs_new_protected_nopar:Npn \spath_split_at_self_intersections:Nn #1#2
3419 {
3420   \__spath_split_at_self_intersections:n {#2}
3421   \tl_set_eq:NN #1 \g__spath_output_tl
3422   \tl_gclear:N \g__spath_output_tl
3423 }
3424 \cs_generate_variant:Nn \spath_split_at_self_intersections:Nn {NV, cn, cV, cv}
3425 \cs_new_protected_nopar:Npn \spath_split_at_self_intersections:N #1
3426 {
3427   \spath_split_at_self_intersections:NV #1#1
3428 }
3429 \cs_generate_variant:Nn \spath_split_at_self_intersections:N {c}

```

```

3430 \cs_new_protected_nopar:Npn \spath_gsplit_at_self_intersections:Nn #1#2
3431 {
3432   \__spath_split_at_self_intersections:n {#2}
3433   \tl_gset_eq:NN #1 \g__spath_output_tl
3434   \tl_gclear:N \g__spath_output_tl
3435 }
3436 \cs_generate_variant:Nn \spath_gsplit_at_self_intersections:Nn {NV, cn, cV, cv}
3437 \cs_new_protected_nopar:Npn \spath_gsplit_at_self_intersections:N #1
3438 {
3439   \spath_gsplit_at_self_intersections:NV #1#1
3440 }
3441 \cs_generate_variant:Nn \spath_gsplit_at_self_intersections:N {c}

```

(End definition for \spath_split_at_self_intersections:Nn and others.)

\spath_join_component:Nnn Join the specified component of the spath to its predecessor.

```

\spath_join_component:Nn 3442 \cs_new_protected_nopar:Npn \__spath_join_component:nn #1#2
\spath_gjoin_component:Nnn 3443 {
\spath_gjoin_component:Nn 3444   \group_begin:
3445   \spath_numberofcomponents:Nn \l__spath_tmpa_int {#1}
3446
3447   \bool_if:nTF
3448   {
3449     \int_compare_p:n { #2 >= 1 }
3450     &&
3451     \int_compare_p:n { #2 <= \l__spath_tmpa_int }
3452   }
3453   {
3454     \int_compare:nTF
3455     {
3456       #2 == 1
3457     }
3458     {
3459       \int_compare:nTF
3460       {
3461         \l__spath_tmpa_int == 1
3462       }
3463       {
3464         \tl_set:Nn \l__spath_tmpa_tl {#1}
3465         \spath_initialpoint:Nn \l__spath_tmpb_tl {#1}
3466         \tl_put_right:NV \l__spath_tmpa_tl \c_spath_closepath_tl
3467         \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
3468         \tl_gclear:N \g__spath_output_tl
3469         \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
3470       }
3471       {
3472         \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
3473         \seq_pop_left:NN \l__spath_tmpa_seq \l__spath_tmpa_tl
3474
3475         \prg_replicate:nn {3}
3476         {
3477           \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
3478         }
3479

```

```

3480         \seq_put_right:NV \l__spath_tmpa_seq \l__spath_tmpa_tl
3481
3482         \tl_gclear:N \g__spath_output_tl
3483         \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpa_seq {}}
3484     }
3485 }
3486 {
3487     \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
3488
3489     \seq_clear:N \l__spath_tmpb_seq
3490     \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
3491     {
3492         \tl_set:Nn \l__spath_tmpa_tl {##2}
3493         \int_compare:nT {##1 = #2}
3494         {
3495             \prg_replicate:nn {3}
3496             {
3497                 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
3498             }
3499         }
3500         \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
3501     }
3502
3503     \tl_gclear:N \g__spath_output_tl
3504     \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpb_seq {}}
3505 }
3506 }
3507 {
3508     \tl_gclear:N \g__spath_output_tl
3509     \tl_gset:Nn \g__spath_output_tl {#1}
3510 }
3511
3512 \group_end:
3513 }
3514 \cs_new_protected_nopar:Npn \spath_join_component:Nnn #1#2#3
3515 {
3516     \__spath_join_component:nn {#2}{#3}
3517     \tl_set_eq:NN #1 \g__spath_output_tl
3518     \tl_gclear:N \g__spath_output_tl
3519 }
3520 \cs_generate_variant:Nn \spath_join_component:Nnn {NVn, NVV}
3521 \cs_new_protected_nopar:Npn \spath_join_component:Nn #1#2
3522 {
3523     \spath_join_component:NVn #1#1{#2}
3524 }
3525 \cs_generate_variant:Nn \spath_join_component:Nn {cn, NV, cV}
3526 \cs_new_protected_nopar:Npn \spath_gjoin_component:Nnn #1#2#3
3527 {
3528     \__spath_join_component:nn {#2}{#3}
3529     \tl_gset_eq:NN #1 \g__spath_output_tl
3530     \tl_gclear:N \g__spath_output_tl
3531 }
3532 \cs_generate_variant:Nn \spath_gjoin_component:Nnn {NVn, NVV}
3533 \cs_new_protected_nopar:Npn \spath_gjoin_component:Nn #1#2

```



```

3534 {
3535   \spath_gjoin_component:N\n #1#1{#2}
3536 }
3537 \cs_generate_variant:Nn \spath_gjoin_component:Nn {cn, NV, cV}

```

(End definition for \spath_join_component:Nnn and others.)

\spath_spot_weld_components:Nn Weld together any components where the last point of one is at the start point of the next (within a tolerance).

```

\spath_spot_weld_components:N
\spath_spot_gweld_components:Nn
\spath_spot_gweld_components:N
3538 \cs_new_protected_nopar:Npn \__spath_spot_weld_components:n #1
3539 {
3540   \group_begin:
3541   \dim_zero:N \l__spath_move_x_dim
3542   \dim_zero:N \l__spath_move_y_dim
3543
3544   \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
3545   \seq_clear:N \l__spath_tmpb_seq
3546   \dim_set:Nn \l__spath_move_x_dim {\tl_item:nn {#1} {2} + 10 pt}
3547   \dim_set:Nn \l__spath_move_y_dim {\tl_item:nn {#1} {3} + 10 pt}
3548
3549   \int_set:Nn \l__spath_tmpa_int {\seq_count:N \l__spath_tmpa_seq}
3550
3551   \seq_map_inline:Nn \l__spath_tmpa_seq
3552   {
3553     \tl_set:Nn \l__spath_tmpa_tl {##1}
3554     \bool_if:nT
3555     {
3556       \dim_compare_p:n
3557       {
3558         \dim_abs:n
3559         {\l__spath_move_x_dim - \tl_item:Nn \l__spath_tmpa_tl {2} }
3560         < 0.01pt
3561       }
3562       &&
3563       \dim_compare_p:n
3564       {
3565         \dim_abs:n
3566         {\l__spath_move_y_dim - \tl_item:Nn \l__spath_tmpa_tl {3} }
3567         < 0.01pt
3568       }
3569     }
3570     {
3571       \prg_replicate:nn {3}
3572       {
3573         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
3574       }
3575       \int_decr:N \l__spath_tmpa_int
3576     }
3577     \tl_reverse:N \l__spath_tmpa_tl
3578     \dim_set:Nn \l__spath_move_x_dim {\tl_item:Nn \l__spath_tmpa_tl {2}}
3579     \dim_set:Nn \l__spath_move_y_dim {\tl_item:Nn \l__spath_tmpa_tl {1}}
3580     \tl_reverse:N \l__spath_tmpa_tl
3581     \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
3582   }

```

```

3583
3584 \tl_set:Nx \l__spath_tmpa_tl {\seq_use:Nn \l__spath_tmpb_seq {} }
3585 \spath_components_to_seq:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
3586
3587
3588 \spath_initialpoint:Nn \l__spath_tmpa_tl {#1}
3589 \spath_finalpoint:Nn \l__spath_tmpb_tl {#1}
3590
3591 \bool_if:nT
3592 {
3593   \dim_compare_p:n
3594   {
3595     \dim_abs:n
3596     {
3597       \tl_item:Nn \l__spath_tmpa_tl {1} - \tl_item:Nn \l__spath_tmpb_tl {1}
3598     }
3599     <
3600     0.01pt
3601   }
3602   &&
3603   \dim_compare_p:n
3604   {
3605     \dim_abs:n
3606     {
3607       \tl_item:Nn \l__spath_tmpa_tl {2} - \tl_item:Nn \l__spath_tmpb_tl {2}
3608     }
3609     <
3610     0.01pt
3611   }
3612 }
3613 {
3614   \int_compare:nTF
3615   {
3616     \seq_count:N \l__spath_tmpb_seq > 1
3617   }
3618   {
3619     \seq_pop_left:NN \l__spath_tmpb_seq \l__spath_tmpb_tl
3620
3621     \prg_replicate:nn {3}
3622     {
3623       \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
3624     }
3625     \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpb_tl
3626   }
3627   {
3628     \tl_set:NV \l__spath_tmpb_tl \c_spath_closepath_tl
3629     \tl_put_right:Nx \l__spath_tmpb_tl
3630     {
3631       { \tl_item:Nn \l__spath_tmpa_tl {1} }
3632       { \tl_item:Nn \l__spath_tmpa_tl {2} }
3633     }
3634     \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpb_tl
3635   }
3636 }

```

```

3637
3638 \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpb_seq {}}
3639 \group_end:
3640 }
3641 \cs_new_protected_nopar:Npn \spath_spot_weld_components:Nn #1#2
3642 {
3643   \__spath_spot_weld_components:n {#2}
3644   \tl_set_eq:NN #1 \g__spath_output_tl
3645   \tl_gclear:N \g__spath_output_tl
3646 }
3647 \cs_generate_variant:Nn \spath_spot_weld_components:Nn {NV, cV, cn}
3648 \cs_new_protected_nopar:Npn \spath_spot_weld_components:N #1
3649 {
3650   \spath_spot_weld_components:NV #1#1
3651 }
3652 \cs_generate_variant:Nn \spath_spot_weld_components:N {c}
3653 \cs_new_protected_nopar:Npn \spath_spot_gweld_components:Nn #1#2
3654 {
3655   \__spath_spot_weld_components:n {#2}
3656   \tl_gset_eq:NN #1 \g__spath_output_tl
3657   \tl_gclear:N \g__spath_output_tl
3658 }
3659 \cs_generate_variant:Nn \spath_spot_gweld_components:Nn {NV, cV, cn}
3660 \cs_new_protected_nopar:Npn \spath_spot_gweld_components:N #1
3661 {
3662   \spath_spot_gweld_components:NV #1#1
3663 }
3664 \cs_generate_variant:Nn \spath_spot_gweld_components:N {c}

```

(End definition for \spath_spot_weld_components:Nn and others.)

3.8 Exporting Commands

```

\spath_convert_to_svg:Nn Convert the soft path to an SVG document.
\spath_gconvert_to_svg:Nn
3665 \cs_new_protected_nopar:Npn \__spath_convert_to_svg:n #1
3666 {
3667   \group_begin:
3668   \tl_clear:N \l__spath_tmpa_tl
3669   \tl_put_right:Nn \l__spath_tmpa_tl
3670   {
3671     <?xml~ version="1.0"~ standalone="no"?>
3672     \iow_newline:
3673     <!DOCTYPE~ svg~ PUBLIC~ "-//W3C//DTD SVG 1.1//EN"~
3674     "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
3675     \iow_newline:
3676     <svg~ xmlns="http://www.w3.org/2000/svg"~ version="1.1"~viewBox="
3677   }
3678
3679   \spath_minbb:Nn \l__spath_tmpb_tl {#1}
3680   \spath_maxbb:Nn \l__spath_tmpc_tl {#1}
3681   \tl_put_right:Nx \l__spath_tmpa_tl
3682   {
3683     \dim_to_decimal:n
3684     {

```

```

3685     \tl_item:Nn \l__spath_tmpb_tl {1} - 10pt
3686   }
3687   \exp_not:n {~}
3688   \dim_to_decimal:n
3689   {
3690     \tl_item:Nn \l__spath_tmpb_tl {2} - 10pt
3691   }
3692   \exp_not:n {~}
3693   \dim_to_decimal:n
3694   {
3695     \tl_item:Nn \l__spath_tmpc_tl {1}
3696     -
3697     \tl_item:Nn \l__spath_tmpb_tl {1}
3698     + 20pt
3699   }
3700   \exp_not:n {~}
3701   \dim_to_decimal:n
3702   {
3703     \tl_item:Nn \l__spath_tmpc_tl {2}
3704     -
3705     \tl_item:Nn \l__spath_tmpb_tl {2}
3706     + 20pt
3707   }
3708 }
3709
3710 \tl_put_right:Nn \l__spath_tmpa_tl
3711 {
3712   ">
3713   \iow_newline:
3714   <path~ d="
3715 }
3716 \tl_set:Nn \l__spath_tmpc_tl {use:n}
3717 \tl_map_inline:nn {#1}
3718 {
3719   \tl_set:Nn \l__spath_tmpb_tl {##1}
3720   \tl_case:NnF \l__spath_tmpb_tl
3721   {
3722     \c_spath_moveto_tl
3723     {
3724       \tl_put_right:Nn \l__spath_tmpa_tl {M~}
3725       \tl_set:Nn \l__spath_tmpc_tl {use:n}
3726     }
3727     \c_spath_lineto_tl
3728     {
3729       \tl_put_right:Nn \l__spath_tmpa_tl {L~}
3730       \tl_set:Nn \l__spath_tmpc_tl {use:n}
3731     }
3732     \c_spath_closepath_tl
3733     {
3734       \tl_put_right:Nn \l__spath_tmpa_tl {Z~}
3735       \tl_set:Nn \l__spath_tmpc_tl {use_none:n}
3736     }
3737     \c_spath_curvetoa_tl
3738     {

```

```

3739         \tl_put_right:Nn \l__spath_tmpa_tl {C~}
3740         \tl_set:Nn \l__spath_tmpc_tl {use:n}
3741     }
3742     \c_spath_curvetob_tl {
3743         \tl_set:Nn \l__spath_tmpc_tl {use:n}
3744     }
3745     \c_spath_curveto_tl {
3746         \tl_set:Nn \l__spath_tmpc_tl {use:n}
3747     }
3748 }
3749 {
3750     \tl_put_right:Nx
3751     \l__spath_tmpa_tl
3752     {\use:c { \l__spath_tmpc_tl } {\dim_to_decimal:n {##1}} ~}
3753 }
3754 }
3755 \tl_put_right:Nn \l__spath_tmpa_tl
3756 {
3757     "~ fill="none"~ stroke="black"~ />
3758     \iow_newline:
3759     </svg>
3760     \iow_newline:
3761 }
3762 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
3763 \group_end:
3764 }
3765 \cs_new_protected_nopar:Npn \spath_convert_to_svg:Nn #1#2
3766 {
3767     \__spath_convert_to_svg:n {#2}
3768     \tl_set_eq:NN #1 \g__spath_output_tl
3769     \tl_gclear:N \g__spath_output_tl
3770 }
3771 \cs_new_protected_nopar:Npn \spath_gconvert_to_svg:Nn #1#2
3772 {
3773     \__spath_convert_to_svg:n {#2}
3774     \tl_gset_eq:NN #1 \g__spath_output_tl
3775     \tl_gclear:N \g__spath_output_tl
3776 }

```

(End definition for \spath_convert_to_svg:Nn and \spath_gconvert_to_svg:Nn.)

`\spath_export_to_svg:nn` Save a soft path to an SVG file.

```

3777 \iow_new:N \g__spath_stream
3778 \cs_new_protected_nopar:Npn \spath_export_to_svg:nn #1#2
3779 {
3780     \group_begin:
3781     \spath_convert_to_svg:Nn \l__spath_tmpa_tl {#2}
3782     \iow_open:Nn \g__spath_stream {#1 .svg}
3783     \iow_now:Nx \g__spath_stream
3784     {
3785         \tl_use:N \l__spath_tmpa_tl
3786     }
3787     \iow_close:N \g__spath_stream
3788     \group_end:

```

```

3789 }
3790 \cs_generate_variant:Nn \spath_export_to_svg:nn {nv, nV}

```

(End definition for \spath_export_to_svg:nn.)

`\spath_show:n` Displays the soft path on the terminal.

```

3791 \cs_new_protected_nopar:Npn \spath_show:n #1
3792 {
3793   \int_step_inline:nnnn {1} {3} {\tl_count:n {#1}}
3794   {
3795     \iow_term:x {
3796       \tl_item:nn {#1} {##1}
3797       {\tl_item:nn {#1} {##1+1}}
3798       {\tl_item:nn {#1} {##1+2}}
3799     }
3800   }
3801 }
3802 \cs_generate_variant:Nn \spath_show:n {V, v}

```

(End definition for \spath_show:n.)

3.8.1 PGF and TikZ Interface Functions

Spaths come from PGF so we need some functions that get and set spaths from the pgf system.

`\spath_get_current_path:N` Grab the current soft path from PGF.

```

\spath_gget_current_path:N
3803 \cs_new_protected_nopar:Npn \spath_get_current_path:N #1
3804 {
3805   \pgfsyssoftpath@getcurrentpath #1
3806 }
3807 \cs_new_protected_nopar:Npn \spath_gget_current_path:N #1
3808 {
3809   \pgfsyssoftpath@getcurrentpath #1
3810   \tl_gset_eq:NN #1 #1
3811 }

```

(End definition for \spath_get_current_path:N and \spath_gget_current_path:N.)

`\spath_protocol_path:n` This feeds the bounding box of the soft path to PGF to ensure that its current bounding box contains the soft path.

```

3812 \cs_new_protected_nopar:Npn \spath_protocol_path:n #1
3813 {
3814   \spath_minbb:Nn \l__spath_tmpa_tl {#1}
3815   \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl
3816
3817   \spath_maxbb:Nn \l__spath_tmpa_tl {#1}
3818   \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl
3819 }
3820 \cs_generate_variant:Nn \spath_protocol_path:n {V}

```

(End definition for \spath_protocol_path:n.)

`\spath_set_current_path:n` Sets the current path to the specified soft path.

`\spath_set_current_path:N`

```

3821 \cs_new_protected_nopar:Npn \spath_set_current_path:n #1
3822 {
3823   \spath_protocol_path:n {#1}
3824   \tl_set:Nn \l__spath_tmpa_tl {#1}
3825   \pgfsyssoftpath@setcurrentpath\l__spath_tmpa_tl
3826 }
3827 \cs_new_protected_nopar:Npn \spath_set_current_path:N #1
3828 {
3829   \spath_protocol_path:V #1
3830   \pgfsyssoftpath@setcurrentpath #1
3831 }
3832 \cs_generate_variant:Nn \spath_set_current_path:N {c}

```

(End definition for \spath_set_current_path:n and \spath_set_current_path:N.)

`\spath_use_path:nn` Uses the given soft path at the PGF level.

```

3833 \cs_new_protected_nopar:Npn \spath_use_path:nn #1#2
3834 {
3835   \spath_set_current_path:n {#1}
3836   \pgfusepath{#2}
3837 }

```

(End definition for \spath_use_path:nn.)

`\spath_tikz_path:nn` Uses the given soft path at the TikZ level.

```

3838 \cs_new_protected_nopar:Npn \spath_tikz_path:nn #1#2
3839 {
3840   \path[#1] \pgfextra{
3841     \spath_set_current_path:n {#2}
3842     \tl_put_right:Nn \tikz@preactions {\def\tikz@actions@path{#2}}
3843   };
3844 }
3845 \cs_generate_variant:Nn \spath_tikz_path:nn {Vn, VV, nv, Vv, nV}

```

(End definition for \spath_tikz_path:nn.)

`\spath_set_tikz_data:n` Sets the `\tikz@lastx` and other coordinates from the soft path.

```

3846 \cs_new_protected_nopar:Npn \spath_set_tikz_data:n #1
3847 {
3848   \spath_finalpoint:Nn \l__spath_tmpa_tl {#1}
3849   \tl_set:Nx \l__spath_tmpa_tl
3850   {
3851     \exp_not:c {pgf@x}=\tl_item:Nn \l__spath_tmpa_tl {1}
3852     \exp_not:c {pgf@y}=\tl_item:Nn \l__spath_tmpa_tl {2}
3853   }
3854   \use:c {pgf@process}{%
3855     \tl_use:N \l__spath_tmpa_tl
3856     \pgftransforminvert
3857     \use:c {pgf@pos@transform@glob}
3858   }
3859   \tl_set:Nx \l__spath_tmpa_tl
3860   {
3861     \exp_not:c {tikz@lastx}=\exp_not:c {pgf@x}

```

```

3862 \exp_not:c {tikz@lasty}=\exp_not:c {pgf@y}
3863 \exp_not:c {tikz@lastxsaved}=\exp_not:c {pgf@x}
3864 \exp_not:c {tikz@lastysaved}=\exp_not:c {pgf@y}
3865 }
3866 \tl_use:N \l__spath_tmpa_tl
3867 \spath_finalmovepoint:Nn \l__spath_tmpa_tl {#1}
3868 \ifpgfsyssoftpathmovetorelevant%
3869 \tl_gset_eq:cN {pgfsyssoftpath@lastmoveto} \l__spath_tmpa_tl
3870 \fi
3871 \tl_set:Nx \l__spath_tmpa_tl
3872 {
3873   \exp_not:c {pgf@x}=\tl_item:Nn \l__spath_tmpa_tl {1}
3874   \exp_not:c {pgf@y}=\tl_item:Nn \l__spath_tmpa_tl {2}
3875 }
3876 \use:c {pgf@process}{%
3877   \tl_use:N \l__spath_tmpa_tl
3878   \pgftransforminvert
3879   \use:c {pgf@pos@transform@glob}
3880 }
3881 \tl_set:Nx \l__spath_tmpa_tl
3882 {
3883   \exp_not:c {tikz@lastmovetox}=\exp_not:c {pgf@x}
3884   \exp_not:c {tikz@lastmovetoy}=\exp_not:c {pgf@y}
3885 }
3886 \tl_use:N \l__spath_tmpa_tl
3887 \tl_clear_new:c {tikz@timer}
3888 \tl_set:cn {tikz@timer}
3889 {
3890   \spath_reallength:Nn \l__spath_tmpa_int {#1}
3891   \tl_set_eq:Nc \l__spath_tmppb_tl {tikz@time}
3892   \tl_set:Nx \l__spath_tmppb_tl
3893   {\fp_to_decimal:n {(\l__spath_tmppb_tl) * (\l__spath_tmpa_int)}}
3894   \spath_transformation_at:NnV \l__spath_tmppc_tl {#1} \l__spath_tmppb_tl
3895
3896   \tl_set:Nx \l__spath_tmpa_tl
3897   {
3898     \exp_not:N \pgfpoint
3899     { \tl_item:Nn \l__spath_tmppc_tl {5} }
3900     { \tl_item:Nn \l__spath_tmppc_tl {6} }
3901   }
3902   \exp_args:NV \pgftransformshift \l__spath_tmpa_tl
3903
3904   \ifpgfresetnontranslationatime
3905   \pgftransformresetnontranslations
3906   \fi
3907
3908   \ifpgfslopedatime
3909
3910   \tl_set:Nx \l__spath_tmpa_tl
3911   {
3912     { \tl_item:Nn \l__spath_tmppc_tl {1} }
3913     { \tl_item:Nn \l__spath_tmppc_tl {2} }
3914     { \tl_item:Nn \l__spath_tmppc_tl {3} }
3915     { \tl_item:Nn \l__spath_tmppc_tl {4} }

```



```

3916 }
3917 \ifpgfallowupsidedownatime
3918 \else
3919 \fp_compare:nT { \tl_item:Nn \l__spath_tmpc_tl {4} < 0 }
3920 {
3921   \tl_set:Nx \l__spath_tmpa_tl
3922   {
3923     { \fp_eval:n { - (\tl_item:Nn \l__spath_tmpc_tl {1}) } }
3924     { \fp_eval:n { - (\tl_item:Nn \l__spath_tmpc_tl {2}) } }
3925     { \fp_eval:n { - (\tl_item:Nn \l__spath_tmpc_tl {3}) } }
3926     { \fp_eval:n { - (\tl_item:Nn \l__spath_tmpc_tl {4}) } }
3927   }
3928 }
3929 \fi
3930 \tl_put_right:Nn \l__spath_tmpa_tl {\pgfpointorigin}
3931 \exp_last_unbraced:NV \pgftransformcm \l__spath_tmpa_tl
3932 \fi
3933 }
3934 }
3935 \cs_generate_variant:Nn \spath_set_tikz_data:n {V, v}

```

(End definition for `\spath_set_tikz_data:n`.)

4 The TikZ interface

This provides an interface to the soft path manipulation routines via a series of TikZ keys. They all live in the `spath` family.

```

3936 \RequirePackage{spath3}
3937 \RequirePackage{expl3}
3938 \ExplSyntaxOn
3939
3940 \tl_new:N \l__spath_current_tl
3941 \tl_new:N \l__spath_reverse_tl
3942 \tl_new:N \l__spath_prefix_tl
3943 \tl_new:N \l__spath_suffix_tl
3944 \tl_new:N \g__spath_smuggle_tl
3945 \seq_new:N \g__spath_tmpa_seq
3946 \seq_new:N \g__spath_tmpb_seq
3947 \bool_new:N \l__spath_draft_bool

```

We surround all the keys with checks to ensure that the soft path under consideration does actually exist, but if it doesn't we should warn the user.

```

3948 \msg_new:nnn { spath3 } { missing soft path } { Soft~ path~ #1~ doesn't~ exist }
3949 \msg_new:nnn { spath3 } { empty soft path } { Soft~ path~ #1~ is~ empty }

```

When saving a soft path, by default we use a naming convention that is compatible with the intersections library so that paths saved here and paths saved by the `name path` facility of the intersections library are mutually exchangeable.

```

3950 \tl_set:Nn \l__spath_prefix_tl {tikz@intersect@path@name@}
3951 \tl_set:Nn \l__spath_suffix_tl {}

```

When a soft path is grabbed from TikZ we're usually deep in a group so I've adapted the code from the intersections library to dig the definition out of the group without making everything global.

```

3952 \tl_new:N \g__spath_tikzfinish_tl
3953 \cs_new_protected_nopar:Npn \spath_at_end_of_path:
3954 {
3955   \tl_use:N \g__spath_tikzfinish_tl
3956   \tl_gclear:N \g__spath_tikzfinish_tl
3957 }
3958 \tl_put_right:Nn \tikz@finish {\spath_at_end_of_path:}
3959
3960 \cs_new_protected_nopar:Npn \spath_save_path:Nn #1#2
3961 {
3962   \tl_gput_right:Nn \g__spath_tikzfinish_tl
3963   {
3964     \tl_clear_new:N #1
3965     \tl_set:Nn #1 {#2}
3966   }
3967 }
3968 \cs_generate_variant:Nn \spath_save_path:Nn {cn, NV, cV}
3969
3970 \cs_new_protected_nopar:Npn \spath_gsave_path:Nn #1#2
3971 {
3972   \tl_gput_right:Nn \g__spath_tikzfinish_tl
3973   {
3974     \tl_gclear_new:N #1
3975     \tl_gset:Nn #1 {#2}
3976   }
3977 }
3978 \cs_generate_variant:Nn \spath_gsave_path:Nn {cn, NV, cV}

```

`_spath_process_tikz_point:Nn` Process a point via TikZ and store the resulting dimensions.

```

3979 \cs_new_protected_nopar:Npn \_spath_process_tikz_point:Nn #1#2
3980 {
3981   \group_begin:
3982   \use:c {tikz@scan@one@point} \use:n #2 \scan_stop:
3983   \tl_gset:Nx \g__spath_output_tl
3984   {
3985     { \dim_use:c {pgf@x} }
3986     { \dim_use:c {pgf@y} }
3987   }
3988   \group_end:
3989   \tl_set_eq:NN #1 \g__spath_output_tl
3990   \tl_gclear:N \g__spath_output_tl
3991 }

```

(End definition for `_spath_process_tikz_point:Nn`.)

When we split a soft path into components, we make it a comma separated list so that it can be fed into a `\foreach` loop. This can also make it possible to extract a single component, but to do this we need a wrapper around `\clist_item:Nn` (there doesn't appear to be a PGF way of getting an item of a CS list).

```

3992 \cs_set_eq:NN \GetComponentOf \clist_item:Nn

```

Now we define all of our keys.

```
3993 \tikzset{
```

We're in the `spath` key family.

```
3994   spath/.is~family,
3995   spath/.cd,
```

We provide for saving soft paths with a specific prefix and suffix in the name. The default is to make it compatible with the intersections library.

```
3996   set~ prefix/.store~ in=\l__spath_prefix_tl,
3997   prefix/.is~choice,
3998   prefix/default/.style={
3999     /tikz/spath/set~ prefix=tikz@intersect@path@name@
4000   },
4001   set~ suffix/.store~ in=\l__spath_suffix_tl,
4002   suffix/.is~choice,
4003   suffix/default/.style={
4004     /tikz/spath/set~ suffix={}
4005   },
4006   set~ name/.style={
4007     /tikz/spath/prefix=#1,
4008     /tikz/spath/suffix=#1
4009   },
```

Keys for saving and cloning a soft path.

```
4010   save/.code={
4011     \tikz@addmode{
4012       \spath_get_current_path:N \l__spath_tmpa_tl
4013       \spath_bake_round:N \l__spath_tmpa_tl \l__spath_tmpa_tl
4014       \spath_save_path:cV
4015       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4016       \l__spath_tmpa_tl
4017     }
4018   },
4019   clone/.code~ 2~ args={
4020     \tl_if_exist:cTF
4021     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4022     {
4023       \tl_clear_new:c
4024       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4025       \tl_set_eq:cc
4026       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4027       {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4028     }
4029     {
4030       \msg_warning:nnn { spath3 } { missing soft path } { #2 }
4031     }
4032   },
4033   clone~ global/.code~ 2~ args={
4034     \tl_if_exist:cTF
4035     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4036     {
4037       \tl_gclear_new:c
4038       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
```

```

4039     \tl_gset_eq:cc
4040     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4041     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4042   }
4043   {
4044     \msg_warning:nnn { spath3 } { missing soft path } { #2 }
4045   }
4046 },
4047 save~ global/.code={
4048   \tikz@addmode{
4049     \spath_get_current_path:N \l__spath_tmpa_tl
4050     \spath_bake_round:NV \l__spath_tmpa_tl \l__spath_tmpa_tl
4051     \spath_gsave_path:cV
4052     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4053     \l__spath_tmpa_tl
4054   }
4055 },

```

Saves a soft path to the aux file.

```

4056 save~ to~ aux/.code={
4057   \tl_if_exist:cTF
4058   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4059   {
4060     \spath_save_to_aux:c
4061     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4062   }
4063   {
4064     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4065   }
4066 },

```

Restores a soft path to the current path.

```

4067 restore/.code={
4068   \tl_if_exist:cTF
4069   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4070   {
4071     \tl_if_empty:cTF
4072     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4073     {
4074       \tl_clear:N \l__spath_tmpa_tl
4075       \tl_put_right:NV \l__spath_tmpa_tl \c_spath_moveto_tl
4076       \tl_put_right:Nn \l__spath_tmpa_tl {{0pt}}{0pt}}
4077       \spath_set_current_path:V \l__spath_tmpa_tl
4078       \spath_set_tikz_data:V \l__spath_tmpa_tl
4079       \msg_warning:nnn { spath3 } { empty soft path } { #1 }
4080     }
4081     {
4082       \spath_set_current_path:c
4083       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4084       \spath_set_tikz_data:v
4085       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4086     }
4087   }
4088   {

```

```

4089 \tl_clear:N \l__spath_tmpa_tl
4090 \tl_put_right:NV \l__spath_tmpa_tl \c_spath_moveto_tl
4091 \tl_put_right:Nn \l__spath_tmpa_tl {{Opt}{Opt}}
4092 \spath_set_current_path:V \l__spath_tmpa_tl
4093 \spath_set_tikz_data:V \l__spath_tmpa_tl
4094 \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4095 }
4096 },

```

Restores the reverse of a soft path to the current path.

```

4097 restore~ reverse/.code={
4098 \tl_if_exist:cTF
4099 {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4100 {
4101 \tl_if_empty:cTF
4102 {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4103 {
4104 \tl_clear:N \l__spath_tmpa_tl
4105 \tl_put_right:NV \l__spath_tmpa_tl \c_spath_moveto_tl
4106 \tl_put_right:Nn \l__spath_tmpa_tl {{Opt}{Opt}}
4107 \spath_set_current_path:V \l__spath_tmpa_tl
4108 \spath_set_tikz_data:V \l__spath_tmpa_tl
4109 \msg_warning:nnn { spath3 } { empty soft path } { #1 }
4110 }
4111 {
4112 \spath_reverse:Nv
4113 \l__spath_reverse_tl
4114 {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4115 \spath_set_current_path:N \l__spath_reverse_tl
4116 \spath_set_tikz_data:V \l__spath_reverse_tl
4117 }
4118 }
4119 {
4120 \tl_clear:N \l__spath_tmpa_tl
4121 \tl_put_right:NV \l__spath_tmpa_tl \c_spath_moveto_tl
4122 \tl_put_right:Nn \l__spath_tmpa_tl {{Opt}{Opt}}
4123 \spath_set_current_path:V \l__spath_tmpa_tl
4124 \spath_set_tikz_data:V \l__spath_tmpa_tl
4125 \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4126 }
4127 },

```

Diagnostic, show the current path in the terminal and log.

```

4128 show~current~path/.code={
4129 \tikz@addmode{
4130 \pgfsyssoftpath@getcurrentpath\l__spath_tmpa_tl
4131 \iow_term:n {---~ current~ soft~ path~ ---}
4132 \spath_show:V \l__spath_tmpa_tl
4133 }
4134 },

```

Diagnostic, show the named soft path in the terminal and log.

```

4135 show/.code={
4136 \tl_if_exist:cTF

```

```

4137 {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4138 {
4139   \tl_if_empty:cTF
4140   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4141   {
4142     \msg_warning:nnn { spath3 } { empty soft path } { #1 }
4143   }
4144   {
4145     \iow_term:n {---- soft~ path~ #1~ ---}
4146     \spath_show:v
4147     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4148   }
4149 }
4150 {
4151   \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4152 }
4153 },

```

Appends the named path to the current path with a weld.

```

4154 append/.code={
4155   \tl_if_exist:cTF
4156   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4157   {
4158     \spath_get_current_path:N \l__spath_current_tl
4159     \spath_finalpoint:NV \l__spath_tmpa_tl \l__spath_current_tl
4160     \tl_set_eq:Nc
4161     \l__spath_tmpb_tl
4162     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4163     \spath_translate_to:NV \l__spath_tmpb_tl \l__spath_tmpa_tl
4164     \spath_append_no_move:NV \l__spath_current_tl \l__spath_tmpb_tl
4165     \spath_set_current_path:N \l__spath_current_tl
4166     \spath_set_tikz_data:V \l__spath_tmpb_tl
4167   }
4168   {
4169     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4170   }
4171 },

```

Joins the second named path to the first.

```

4172 join~ with/.code~ 2~ args={
4173   \tl_if_exist:cTF
4174   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4175   {
4176     \tl_if_exist:cTF
4177     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4178     {
4179       \spath_append:cv
4180       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4181       {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4182     }
4183     {
4184       \msg_warning:nnn { spath3 } { missing soft path } { #2 }
4185     }
4186   }

```

```

4187     {
4188       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4189     }
4190   },

```

Does a “spot weld” on a soft path, which means that any components that start where the previous component ends are welded together.

```

4191   spot~ weld/.code={
4192     \tl_if_exist:cTF
4193     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4194     {
4195       \spath_spot_weld_components:c
4196       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4197     }
4198     {
4199       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4200     }
4201   },
4202   spot~ weld~ globally/.code={
4203     \tl_if_exist:cTF
4204     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4205     {
4206       \spath_spot_gweld_components:c
4207       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4208     }
4209     {
4210       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4211     }
4212   },

```

Reverses the named path.

```

4213   reverse/.code={
4214     \tl_if_exist:cTF
4215     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4216     {
4217       \spath_reverse:c
4218       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4219     }
4220     {
4221       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4222     }
4223   },
4224   reverse~ globally/.code={
4225     \tl_if_exist:cTF
4226     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4227     {
4228       \spath_reverse:c
4229       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4230     }
4231     {
4232       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4233     }
4234   },

```

Appends the reversal of the path to the current path.

```

4235 append~ reverse/.code={
4236   \tl_if_exist:cTF
4237   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4238   {
4239     \spath_reverse:Nv
4240     \l__spath_reverse_tl
4241     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4242     \spath_get_current_path:N \l__spath_current_tl
4243     \spath_finalpoint:Nv \l__spath_tmpa_tl \l__spath_current_tl
4244     \spath_translate_to:Nv \l__spath_reverse_tl \l__spath_tmpa_tl
4245     \spath_append_no_move:Nv \l__spath_current_tl \l__spath_reverse_tl
4246     \spath_set_current_path:N \l__spath_current_tl
4247     \spath_set_tikz_data:V \l__spath_reverse_tl
4248   }
4249   {
4250     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4251   }
4252 },

```

Inserts the named path into the current path as-is, meaning without transforming or welding it.

```

4253 insert/.code={
4254   \tl_if_exist:cTF
4255   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4256   {
4257     \spath_get_current_path:N \l__spath_current_tl
4258     \spath_append:Nv
4259     \l__spath_current_tl
4260     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4261     \spath_set_current_path:N \l__spath_current_tl
4262     \spath_set_tikz_data:v
4263     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4264   }
4265   {
4266     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4267   }
4268 },

```

Inserts the reverse of the named path.

```

4269 insert~ reverse/.code={
4270   \tl_if_exist:cTF
4271   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4272   {
4273     \spath_reverse:Nv
4274     \l__spath_reverse_tl
4275     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4276     \spath_get_current_path:N \l__spath_current_tl
4277     \spath_append:Nv \l__spath_current_tl \l__spath_reverse_tl
4278     \spath_set_current_path:N \l__spath_current_tl
4279     \spath_set_tikz_data:V \l__spath_reverse_tl
4280   }
4281   {
4282     \msg_warning:nnn { spath3 } { missing soft path } { #1 }

```



```

4283     }
4284 },

```

Appends the named path to the current path without translating it.

```

4285 append~ no~ move/.code={
4286   \tl_if_exist:cTF
4287   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4288   {
4289     \spath_get_current_path:N \l__spath_current_tl
4290     \spath_append_no_move:Nv
4291     \l__spath_current_tl
4292     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4293     \spath_set_current_path:N \l__spath_current_tl
4294     \spath_set_tikz_data:v
4295     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4296   }
4297   {
4298     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4299   }
4300 },

```

Appends the reverse of the named path without translating it.

```

4301 append~ reverse~ no~ move/.code={
4302   \tl_if_exist:cTF
4303   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4304   {
4305     \spath_reverse:Nv
4306     \l__spath_reverse_tl
4307     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4308     \spath_get_current_path:N \l__spath_current_tl
4309     \spath_append_no_move:Nv \l__spath_current_tl \l__spath_reverse_tl
4310     \spath_set_current_path:N \l__spath_current_tl
4311     \spath_set_tikz_data:V \l__spath_reverse_tl
4312   }
4313   {
4314     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4315   }
4316 },

```

Adjust a path to span between two points.

```

4317 span/.code ~n~ args={3}{
4318   \tl_if_exist:cTF
4319   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4320   {
4321     \__spath_process_tikz_point:Nn \l__spath_tmpa_tl {#2}
4322     \__spath_process_tikz_point:Nn \l__spath_tmpb_tl {#3}
4323     \spath_span:cVV
4324     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4325     \l__spath_tmpa_tl \l__spath_tmpb_tl
4326   }
4327   {
4328     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4329   }
4330 },

```

```

4331 span~ global/.code ~n~ args={3}{
4332   \tl_if_exist:cTF
4333   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4334   {
4335     \__spath_process_tikz_point:Nn \l__spath_tmpa_tl {#2}
4336     \__spath_process_tikz_point:Nn \l__spath_tmpb_tl {#3}
4337     \spath_gspan:cVV
4338     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4339     \l__spath_tmpa_tl \l__spath_tmpb_tl
4340   }
4341   {
4342     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4343   }
4344 },

```

Defines a to path

```

4345 to/.style={
4346   to~path={
4347     [
4348       spath/span={#1}{(\tikztostart)}{(\tikztotarget)},
4349       spath/append~no~move={#1},
4350     ]
4351     \tikztonodes
4352   }
4353 },

```

Splice three paths together, transforming the middle one so that it exactly fits between the first and third.

```

4354 splice/.code ~n~ args={3}{
4355   \tl_if_exist:cTF
4356   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4357   {
4358     \tl_if_exist:cTF
4359     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4360     {
4361       \tl_if_exist:cTF
4362       {\tl_use:N \l__spath_prefix_tl #3 \tl_use:N \l__spath_suffix_tl}
4363       {
4364         \spath_splice_between:cvv
4365         {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4366         {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4367         {\tl_use:N \l__spath_prefix_tl #3 \tl_use:N \l__spath_suffix_tl}
4368       }
4369       {
4370         \msg_warning:nnn { spath3 } { missing soft path } { #3 }
4371       }
4372     }
4373     {
4374       \msg_warning:nnn { spath3 } { missing soft path } { #2 }
4375     }
4376   }
4377   {
4378     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4379   }

```

```

4380 },
4381 splice~ global/.code ~n~ args={3}{
4382   \tl_if_exist:cTF
4383   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4384   {
4385     \tl_if_exist:cTF
4386     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4387     {
4388       \tl_if_exist:cTF
4389       {\tl_use:N \l__spath_prefix_tl #3 \tl_use:N \l__spath_suffix_tl}
4390       {
4391         \spath_gsplice_between:cvv
4392         {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4393         {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4394         {\tl_use:N \l__spath_prefix_tl #3 \tl_use:N \l__spath_suffix_tl}
4395       }
4396       {
4397         \msg_warning:nnn { spath3 } { missing soft path } { #3 }
4398       }
4399     }
4400     {
4401       \msg_warning:nnn { spath3 } { missing soft path } { #2 }
4402     }
4403   }
4404   {
4405     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4406   }
4407 },

```

Join the components of a path by splicing in the second path whenever the components are sufficiently far apart.

```

4408 join~ components~ with/.code~2~args={
4409   \tl_if_exist:cTF
4410   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4411   {
4412     \tl_if_exist:cTF
4413     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4414     {
4415       \spath_spot_weld_components:c
4416       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4417
4418       \spath_components_to_seq:Nv
4419       \l__spath_tmpa_seq
4420       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4421
4422       \seq_pop_left:NN \l__spath_tmpa_seq \l__spath_tmpa_tl
4423
4424       \seq_map_inline:Nn \l__spath_tmpa_seq
4425       {
4426         \spath_splice_between:Nvn \l__spath_tmpa_tl
4427         {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4428         {##1}
4429       }
4430       \tl_set_eq:cN

```

```

4431         {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4432         \l__spath_tmpa_tl
4433     }
4434     {
4435         \msg_warning:nnn { spath3 } { missing soft path } { #2 }
4436     }
4437 }
4438 {
4439     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4440 }
4441 },
4442 join~ components~ globally~ with/.code~2~args={
4443     \tl_if_exist:cTF
4444     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4445     {
4446         \tl_if_exist:cTF
4447         {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4448         {
4449             \spath_spot_gweld_components:c
4450             {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4451
4452             \spath_components_to_seq:Nv
4453             \l__spath_tmpa_seq
4454             {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4455
4456             \seq_pop_left:NN \l__spath_tmpa_seq \l__spath_tmpa_tl
4457
4458             \seq_map_inline:Nn \l__spath_tmpa_seq
4459             {
4460                 \spath_gsplice_between:Nvn \l__spath_tmpa_tl
4461                 {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4462                 {##1}
4463             }
4464             \tl_set_eq:cN
4465             {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4466             \l__spath_tmpa_tl
4467         }
4468         {
4469             \msg_warning:nnn { spath3 } { missing soft path } { #2 }
4470         }
4471     }
4472     {
4473         \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4474     }
4475 },

```

Close a path.

```

4476 close/.code={
4477     \tl_if_exist:cTF
4478     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4479     {
4480         \spath_close:c
4481         {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4482     }

```

```

4483     {
4484       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4485     }
4486   },
4487   close~ globally/.code={
4488     \tl_if_exist:cTF
4489     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4490     {
4491       \spath_gclose:c
4492       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4493     }
4494     {
4495       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4496     }
4497   },

```

Close a path with another path.

```

4498   close~ with/.code~ 2~ args={
4499     \tl_if_exist:cTF
4500     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4501     {
4502       \tl_if_exist:cTF
4503       {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4504       {
4505         \spath_close_with:cv
4506         {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4507         {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4508       }
4509       {
4510         \msg_warning:nnn { spath3 } { missing soft path } { #2 }
4511       }
4512     }
4513     {
4514       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4515     }
4516   },
4517   close~ globally~ with/.code~ 2~ args={
4518     \tl_if_exist:cTF
4519     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4520     {
4521       \tl_if_exist:cTF
4522       {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4523       {
4524         \spath_gclose_with:cv
4525         {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4526         {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4527       }
4528       {
4529         \msg_warning:nnn { spath3 } { missing soft path } { #2 }
4530       }
4531     }
4532     {
4533       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4534     }

```

4535 },

These keys shorten the path.

```
4536 shorten~ at~ end/.code~ 2~ args={
4537   \tl_if_exist:cTF
4538   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4539   {
4540     \spath_shorten_at_end:cn
4541     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
4542   }
4543   {
4544     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4545   }
4546 },
4547 shorten~ at~ start/.code~ 2~ args ={
4548   \tl_if_exist:cTF
4549   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4550   {
4551     \spath_shorten_at_start:cn
4552     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
4553   }
4554   {
4555     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4556   }
4557 },
4558 shorten~ at~ both~ ends/.code~ 2~ args={
4559   \tl_if_exist:cTF
4560   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4561   {
4562     \spath_shorten_at_end:cn
4563     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
4564     \spath_shorten_at_start:cn
4565     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
4566   }
4567   {
4568     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4569   }
4570 },
4571 shorten~ globally~ at~ end/.code~ 2~ args={
4572   \tl_if_exist:cTF
4573   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4574   {
4575     \spath_gshorten_at_end:cn
4576     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
4577   }
4578   {
4579     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4580   }
4581 },
4582 shorten~ globally~ at~ start/.code~ 2~ args ={
4583   \tl_if_exist:cTF
4584   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4585   {
4586     \spath_gshorten_at_start:cn
```

```

4587     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
4588   }
4589   {
4590     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4591   }
4592 },
4593 shorten~ globally~ at~ both~ ends/.code~ 2~ args={
4594   \tl_if_exist:cTF
4595   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4596   {
4597     \spath_shorten_at_end:cn
4598     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
4599     \spath_shorten_at_start:cn
4600     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
4601   }
4602   {
4603     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4604   }
4605 },

```

This translates the named path.

```

4606   translate/.code~ n~ args={3}{
4607     \tl_if_exist:cTF
4608     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4609     {
4610       \spath_translate:cnn
4611       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}{#2}{#3}
4612     }
4613     {
4614       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4615     }
4616   },
4617   translate~ globally/.code~ n~ args={3}{
4618     \tl_if_exist:cTF
4619     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4620     {
4621       \spath_gtranslate:cnn
4622       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}{#2}{#3}
4623     }
4624     {
4625       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4626     }
4627   },

```

This normalises the named path.

```

4628   normalise/.code={
4629     \tl_if_exist:cTF
4630     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4631     {
4632       \spath_normalise:c
4633       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4634     }
4635     {
4636       \msg_warning:nnn { spath3 } { missing soft path } { #1 }

```

```

4637     }
4638   },
4639   normalise~ globally/.code={
4640     \tl_if_exist:cTF
4641     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4642     {
4643       \spath_gnormalise:c
4644       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4645     }
4646     {
4647       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4648     }
4649   },

```

Exports the path as an SVG file.

```

4650   export~ to~ svg/.code={
4651     \tl_if_exist:cTF
4652     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4653     {
4654       \spath_export_to_svg:nv {#1}
4655       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4656     }
4657     {
4658       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4659     }
4660   },

```

Transforms the named path using TikZ transformation specifications.

```

4661   transform/.code~ 2~ args={
4662     \tl_if_exist:cTF
4663     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4664     {
4665       \group_begin:
4666       \pgftransformreset
4667       \tikzset{#2}
4668       \pgfgettransform \l__spath_tmpa_tl
4669       \tl_gset:Nn \g__spath_smuggle_tl
4670       {
4671         \spath_transform:cnnnnnn
4672         {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4673       }
4674       \tl_gput_right:NV \g__spath_smuggle_tl \l__spath_tmpa_tl
4675       \group_end:
4676       \tl_use:N \g__spath_smuggle_tl
4677     }
4678     {
4679       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4680     }
4681   },
4682   transform~globally/.code~ 2~ args={
4683     \tl_if_exist:cTF
4684     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4685     {
4686       \group_begin:

```



```

4687 \pgftransformreset
4688 \tikzset{#2}
4689 \pgfgettransform \l__spath_tmpa_tl
4690 \tl_gset:Nn \g__spath_smuggle_tl
4691 {
4692   \spath_gtransform:cnnnnnn
4693   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4694 }
4695 \tl_gput_right:NV \g__spath_smuggle_tl \l__spath_tmpa_tl
4696 \group_end:
4697 \tl_use:N \g__spath_smuggle_tl
4698 }
4699 {
4700   \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4701 }
4702 },

```

Splits first path where it intersects with the second.

```

4703 split~ at~ intersections~ with/.code~ n~ args={2}{
4704   \tl_if_exist:cTF
4705   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4706   {
4707     \tl_if_exist:cTF
4708     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4709     {
4710       \spath_split_path_at_intersections:cv
4711       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4712       {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4713     }
4714     {
4715       \msg_warning:nnn { spath3 } { missing soft path } { #2 }
4716     }
4717   }
4718   {
4719     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4720   }
4721 },
4722 split~ globally~ at~ intersections~ with/.code~ n~ args={2}{
4723   \tl_if_exist:cTF
4724   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4725   {
4726     \tl_if_exist:cTF
4727     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4728     {
4729       \spath_gsplit_path_at_intersections:cv
4730       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4731       {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4732     }
4733     {
4734       \msg_warning:nnn { spath3 } { missing soft path } { #2 }
4735     }
4736   }
4737   {
4738     \msg_warning:nnn { spath3 } { missing soft path } { #1 }

```

```

4739     }
4740 },

```

Splits two paths at their mutual intersections.

```

4741 split~ at~ intersections/.code~ n~ args={2}{
4742   \tl_if_exist:cTF
4743   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4744   {
4745     \tl_if_exist:cTF
4746     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4747     {
4748       \spath_split_at_intersections:cc
4749       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4750       {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4751     }
4752     {
4753       \msg_warning:nnn { spath3 } { missing soft path } { #2 }
4754     }
4755   }
4756   {
4757     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4758   }
4759 },
4760 split~ globally~ at~ intersections/.code~ n~ args={2}{
4761   \tl_if_exist:cTF
4762   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4763   {
4764     \tl_if_exist:cTF
4765     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4766     {
4767       \spath_gsplit_at_intersections:cc
4768       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4769       {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4770     }
4771     {
4772       \msg_warning:nnn { spath3 } { missing soft path } { #2 }
4773     }
4774   }
4775   {
4776     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4777   }
4778 },

```

Splits a path at its self-intersections.

```

4779 split~ at~ self~ intersections/.code={
4780   \tl_if_exist:cTF
4781   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4782   {
4783     \spath_split_at_self_intersections:c
4784     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4785   }
4786   {
4787     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4788   }

```

```

4789 },
4790 split~ globally~ at~ self~ intersections/.code={
4791   \tl_if_exist:cTF
4792   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4793   {
4794     \spath_gsplit_at_self_intersections:c
4795     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4796   }
4797   {
4798     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4799   }
4800 },

```

Extract the components of a path into a comma separated list (suitable for using in a `\foreach` loop).

```

4801 get~ components~ of/.code~ 2~ args={
4802   \tl_if_exist:cTF
4803   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4804   {
4805     \clist_clear_new:N #2
4806     \spath_components_to_seq:Nv
4807     \l__spath_tmpa_seq
4808     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4809     \seq_map_inline:Nn \l__spath_tmpa_seq
4810     {
4811       \tl_new:c
4812       {
4813         \tl_use:N \l__spath_prefix_tl
4814         anonymous_\int_use:N \g__spath_anon_int
4815         \tl_use:N \l__spath_suffix_tl
4816       }
4817       \tl_set:cn
4818       {
4819         \tl_use:N \l__spath_prefix_tl
4820         anonymous_\int_use:N \g__spath_anon_int
4821         \tl_use:N \l__spath_suffix_tl
4822       } {##1}
4823       \clist_put_right:Nx #2 {anonymous_\int_use:N \g__spath_anon_int}
4824       \int_gincr:N \g__spath_anon_int
4825     }
4826   }
4827   {
4828     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4829   }
4830 },
4831 get~ components~ of~ globally/.code~ 2~ args={
4832   \tl_if_exist:cTF
4833   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4834   {
4835     \clist_gclear_new:N #2
4836     \spath_components_to_seq:Nv
4837     \l__spath_tmpa_seq
4838     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4839     \seq_map_inline:Nn \l__spath_tmpa_seq

```

```

4840 {
4841   \tl_new:c
4842   {
4843     \tl_use:N \l__spath_prefix_tl
4844     anonymous_\int_use:N \g__spath_anon_int
4845     \tl_use:N \l__spath_suffix_tl
4846   }
4847   \tl_gset:cn
4848   {
4849     \tl_use:N \l__spath_prefix_tl
4850     anonymous_\int_use:N \g__spath_anon_int
4851     \tl_use:N \l__spath_suffix_tl
4852   } {##1}
4853   \clist_gput_right:Nx #2 {anonymous_\int_use:N \g__spath_anon_int}
4854   \int_gincr:N \g__spath_anon_int
4855 }
4856 }
4857 {
4858   \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4859 }
4860 },

```

Loop through the components of a soft path and render each as a separate TikZ path so that they can be individually styled.

```

4861 render~ components/.code={
4862   \tl_if_exist:cTF
4863   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4864   {
4865     \group_begin:
4866     \spath_components_to_seq:Nv
4867     \l__spath_tmpa_seq
4868     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4869     \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
4870     {
4871       \spath_tikz_path:nn
4872       {
4873         every~ spath~ component/.try,
4874         spath ~component~ ##1/.try,
4875         spath ~component/.try={##1},
4876         every~ #1~ component/.try,
4877         #1 ~component~ ##1/.try,
4878         #1 ~component/.try={##1},
4879       }
4880       {
4881         ##2
4882       }
4883     }
4884     \group_end:
4885   }
4886   {
4887     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4888   }
4889 },

```

This puts gaps between components of a soft path. The list of components is passed through a `\foreach` loop so can use the shortcut syntax from those loops.

```

4890 insert~ gaps~ after~ components/.code~ n~ args={3}{
4891   \tl_if_exist:cTF
4892   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4893   {
4894     \group_begin:
4895     \seq_gclear:N \g__spath_tmpa_seq
4896     \seq_gclear:N \g__spath_tmpb_seq
4897     \spath_numberofcomponents:Nv \l__spath_tmpa_int
4898     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4899     \foreach \l__spath_tmpa_tl in {#3}
4900     {
4901       \seq_gput_right:Nv \g__spath_tmpa_seq \l__spath_tmpa_tl
4902       \seq_gput_right:Nx
4903       \g__spath_tmpb_seq
4904       {\int_eval:n
4905        {
4906          \int_mod:nn { \l__spath_tmpa_tl }{\l__spath_tmpa_int } + 1
4907        }
4908       }
4909     }
4910     \spath_components_to_seq:Nv
4911     \l__spath_tmpa_seq
4912     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4913     \seq_clear:N \l__spath_tmpb_seq
4914     \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
4915     {
4916       \tl_set:Nn \l__spath_tmpa_tl {##2}
4917       \seq_if_in:NnT \g__spath_tmpa_seq {##1}
4918       {
4919         \spath_shorten_at_end:Nn \l__spath_tmpa_tl {#2/2}
4920       }
4921       \seq_if_in:NnT \g__spath_tmpb_seq {##1}
4922       {
4923         \spath_shorten_at_start:Nn \l__spath_tmpa_tl {#2/2}
4924       }
4925       \seq_put_right:Nv \l__spath_tmpb_seq \l__spath_tmpa_tl
4926     }
4927     \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpb_seq {} }
4928     \group_end:
4929     \tl_set_eq:cN
4930     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4931     \g__spath_output_tl
4932     \tl_gclear:N \g__spath_output_tl
4933   }
4934   {
4935     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4936   }
4937 },
4938 insert~ gaps~ globally~ after~ components/.code~ n~ args={3}{
4939   \tl_if_exist:cTF
4940   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4941   {

```

```

4942 \group_begin:
4943 \seq_gclear:N \g__spath_tmpa_seq
4944 \seq_gclear:N \g__spath_tmpb_seq
4945 \spath_numberofcomponents:Nv \l__spath_tmpa_int
4946 {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4947 \foreach \l__spath_tmpa_tl in {#3}
4948 {
4949   \seq_gput_right:Nv \g__spath_tmpa_seq \l__spath_tmpa_tl
4950   \seq_gput_right:Nx
4951   \g__spath_tmpb_seq
4952   {
4953     \int_eval:n
4954     {
4955       \int_mod:nn { \l__spath_tmpa_tl }{ \l__spath_tmpa_int } + 1
4956     }
4957   }
4958 }
4959 \spath_components_to_seq:Nv
4960 \l__spath_tmpa_seq
4961 {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4962 \seq_clear:N \l__spath_tmpb_seq
4963 \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
4964 {
4965   \tl_set:Nn \l__spath_tmpa_tl {##2}
4966   \seq_if_in:NnT \g__spath_tmpa_seq {##1}
4967   {
4968     \spath_shorten_at_end:Nn \l__spath_tmpa_tl {#2/2}
4969   }
4970   \seq_if_in:NnT \g__spath_tmpb_seq {##1}
4971   {
4972     \spath_shorten_at_start:Nn \l__spath_tmpa_tl {#2/2}
4973   }
4974   \seq_put_right:Nv \l__spath_tmpb_seq \l__spath_tmpa_tl
4975 }
4976 \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpb_seq {} }
4977 \group_end:
4978 \tl_gset_eq:cN
4979 {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4980 \g__spath_output_tl
4981 \tl_gclear:N \g__spath_output_tl
4982 }
4983 {
4984   \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4985 }
4986 },

```

Join the specified components together, joining each to its previous one.

```

4987 join~ components/.code~ 2~ args={
4988   \tl_if_exist:cTF
4989   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4990   {
4991     \seq_gclear:N \g__spath_tmpa_seq
4992     \foreach \l__spath_tmpa_tl in {#2}
4993     {

```

```

4994     \seq_gput_right:NV \g__spath_tmpa_seq \l__spath_tmpa_tl
4995   }
4996   \seq_gsort:Nn \g__spath_tmpa_seq
4997   {
4998     \int_compare:nNnTF {##1} > {##2}
4999     { \sort_return_same: }
5000     { \sort_return_swapped: }
5001   }
5002   \seq_map_inline:Nn \g__spath_tmpa_seq
5003   {
5004     \spath_join_component:cn
5005     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}{##1}
5006   }
5007 }
5008 {
5009   \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5010 }
5011 },
5012 join~ components~ globally/.code~ 2~ args={
5013   \tl_if_exist:cTF
5014   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5015   {
5016     \seq_gclear:N \g__spath_tmpa_seq
5017     \foreach \l__spath_tmpa_tl in {#2}
5018     {
5019       \seq_gput_right:NV \g__spath_tmpa_seq \l__spath_tmpa_tl
5020     }
5021     \seq_gsort:Nn \g__spath_tmpa_seq
5022     {
5023       \int_compare:nNnTF {##1} > {##2}
5024       { \sort_return_same: }
5025       { \sort_return_swapped: }
5026     }
5027     \seq_map_inline:Nn \g__spath_tmpa_seq
5028     {
5029       \spath_gjoin_component:cn
5030       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}{##1}
5031     }
5032   }
5033   {
5034     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5035   }
5036 },

```

Remove all components of the path that don't actually draw anything.

```

5037   remove~ empty~ components/.code={
5038     \tl_if_exist:cTF
5039     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5040     {
5041       \spath_remove_empty_components:c
5042       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5043     }
5044     {
5045       \msg_warning:nnn { spath3 } { missing soft path } { #1 }

```

```

5046     }
5047 },
5048 remove~ empty~ components~ globally/.code={
5049   \tl_if_exist:cTF
5050   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5051   {
5052     \spath_gremove_empty_components:c
5053     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5054   }
5055   {
5056     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5057   }
5058 },

```

Join the specified components together, joining each to its previous one.

```

5059 remove~ components/.code~ 2~ args={
5060   \tl_if_exist:cTF
5061   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5062   {
5063     \seq_gclear:N \g__spath_tmpa_seq
5064     \foreach \l__spath_tmpa_tl in {#2}
5065     {
5066       \seq_gput_right:Nv \g__spath_tmpa_seq \l__spath_tmpa_tl
5067     }
5068     \seq_gsort:Nn \g__spath_tmpa_seq
5069     {
5070       \int_compare:nNnTF {##1} < {##2}
5071       { \sort_return_same: }
5072       { \sort_return_swapped: }
5073     }
5074     \spath_components_to_seq:Nv
5075     \l__spath_tmpa_seq
5076     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5077     \seq_gpop_left:NNF \g__spath_tmpa_seq \l__spath_tmpa_tl
5078     {
5079       \tl_clear:N \l__spath_tmpa_tl
5080     }
5081     \seq_clear:N \l__spath_tmpb_seq
5082     \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
5083     {
5084       \tl_set:Nn \l__spath_tmpb_tl {##1}
5085       \tl_if_eq:NNTF \l__spath_tmpb_tl \l__spath_tmpa_tl
5086       {
5087         \seq_gpop_left:NNF \g__spath_tmpa_seq \l__spath_tmpa_tl
5088         {
5089           \tl_clear:N \l__spath_tmpa_tl
5090         }
5091       }
5092       {
5093         \seq_put_right:Nn \l__spath_tmpb_seq {##2}
5094       }
5095     }
5096     \tl_set:cx {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5097     {\seq_use:Nn \l__spath_tmpb_seq {}} }

```



```

5098     }
5099     {
5100       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5101     }
5102   },
5103   remove~ components~ globally/.code~ 2~ args={
5104     \tl_if_exist:cTF
5105     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5106     {
5107       \seq_gclear:N \g__spath_tmpa_seq
5108       \foreach \l__spath_tmpa_tl in {#2}
5109       {
5110         \seq_gput_right:NV \g__spath_tmpa_seq \l__spath_tmpa_tl
5111       }
5112       \seq_gsort:Nn \g__spath_tmpa_seq
5113       {
5114         \int_compare:nNnTF {##1} < {##2}
5115         { \sort_return_same: }
5116         { \sort_return_swapped: }
5117       }
5118       \spath_components_to_seq:Nv
5119       \l__spath_tmpa_seq
5120       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5121       \seq_gpop_left:NNF \g__spath_tmpa_seq \l__spath_tmpa_tl
5122       {
5123         \tl_clear:N \l__spath_tmpa_tl
5124       }
5125       \seq_clear:N \l__spath_tmpb_seq
5126       \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
5127       {
5128         \tl_set:Nn \l__spath_tmpb_tl {##1}
5129         \tl_if_eq:NNTF \l__spath_tmpb_tl \l__spath_tmpa_tl
5130         {
5131           \seq_gpop_left:NNF \g__spath_tmpa_seq \l__spath_tmpa_tl
5132           {
5133             \tl_clear:N \l__spath_tmpa_tl
5134           }
5135         }
5136         {
5137           \seq_put_right:Nn \l__spath_tmpb_seq {##2}
5138         }
5139       }
5140       \tl_gset:cx {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5141       {\seq_use:Nn \l__spath_tmpb_seq {}}
5142     }
5143     {
5144       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5145     }
5146   },

```

This puts a conditional around the `spot weld` key because when figuring out a knot drawing then we will initially want to render it without the spot weld to keep the number of components constant.

```

5147   draft~ mode/.is~ choice,

```

```

5148 draft~ mode/true/.code={
5149   \bool_set_true:N \l__spath_draft_bool
5150 },
5151 draft~ mode/false/.code={
5152   \bool_set_false:N \l__spath_draft_bool
5153 },
5154 maybe~ spot~ weld/.code={
5155   \bool_if:NF \l__spath_draft_bool
5156   {
5157     \tl_if_exist:cTF
5158     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5159     {
5160       \spath_spot_weld_components:c
5161       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5162     }
5163     {
5164       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5165     }
5166   }
5167 },
5168 maybe~ spot~ weld~ globally/.code={
5169   \bool_if:NF \l__spath_draft_bool
5170   {
5171     \tl_if_exist:cTF
5172     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5173     {
5174       \spath_spot_gweld_components:c
5175       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5176     }
5177     {
5178       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5179     }
5180   }
5181 },

```

Set the transformation to lie along a path.

```

5182 transform~ to/.code~ 2~ args={
5183   \group_begin:
5184   \tl_if_exist:cTF
5185   {
5186     \tl_use:N \l__spath_prefix_tl
5187     #1
5188     \tl_use:N \l__spath_suffix_tl
5189   }
5190   {
5191     \spath_reallength:Nv
5192     \l__spath_tmpa_int
5193     {
5194       \tl_use:N \l__spath_prefix_tl
5195       #1
5196       \tl_use:N \l__spath_suffix_tl
5197     }
5198
5199     \tl_set:Nx \l__spath_tmpb_tl

```

```

5200     {\fp_to_decimal:n {(#2) * (\l__spath_tmpa_int)}}
5201     \spath_transformation_at:NvV \l__spath_tmpc_tl
5202     {
5203       \tl_use:N \l__spath_prefix_tl
5204       #1
5205       \tl_use:N \l__spath_suffix_tl
5206     }
5207     \l__spath_tmpb_tl
5208     \tl_gset_eq:NN \g__spath_smuggle_tl \l__spath_tmpc_tl
5209   }
5210   {
5211     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5212     \tl_gset_eq:NN \g__spath_smuggle_tl { {1}{0}{0}{1}{0pt}{0pt} }
5213   }
5214   \group_end:
5215   \exp_last_unbraced:NV \pgfsettransformentries \g__spath_smuggle_tl
5216   \tl_gclear:N \g__spath_smuggle_tl
5217 },

```

As above, but with a possible extra 180° rotation if needed to ensure that the new y -axis points vaguely upwards.

```

5218   upright~ transform~ to/.code~ 2~ args={
5219     \group_begin:
5220     \tl_if_exist:cTF
5221     {
5222       \tl_use:N \l__spath_prefix_tl
5223       #1
5224       \tl_use:N \l__spath_suffix_tl
5225     }
5226     {
5227       \spath_reallength:Nv
5228       \l__spath_tmpa_int
5229       {
5230         \tl_use:N \l__spath_prefix_tl
5231         #1
5232         \tl_use:N \l__spath_suffix_tl
5233       }
5234
5235       \tl_set:Nx \l__spath_tmpb_tl
5236       {\fp_to_decimal:n {(#2) * (\l__spath_tmpa_int)}}
5237       \spath_transformation_at:NvV \l__spath_tmpc_tl
5238       {
5239         \tl_use:N \l__spath_prefix_tl
5240         #1
5241         \tl_use:N \l__spath_suffix_tl
5242       }
5243       \l__spath_tmpb_tl
5244       \tl_gset_eq:NN \g__spath_smuggle_tl \l__spath_tmpc_tl
5245     }
5246     {
5247       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5248       \tl_gset_eq:NN \g__spath_smuggle_tl { {1}{0}{0}{1}{0pt}{0pt} }
5249     }
5250     \fp_compare:nT { \tl_item:Nn \g__spath_smuggle_tl {4} < 0 }

```

```

5251 {
5252   \tl_gset:Nx \g__spath_smuggle_tl
5253   {
5254     { \fp_eval:n { - (\tl_item:Nn \g__spath_smuggle_tl {1}) } }
5255     { \fp_eval:n { - (\tl_item:Nn \g__spath_smuggle_tl {2}) } }
5256     { \fp_eval:n { - (\tl_item:Nn \g__spath_smuggle_tl {3}) } }
5257     { \fp_eval:n { - (\tl_item:Nn \g__spath_smuggle_tl {4}) } }
5258     { \tl_item:Nn \g__spath_smuggle_tl {5} }
5259     { \tl_item:Nn \g__spath_smuggle_tl {6} }
5260   }
5261 }
5262 \group_end:
5263 \exp_last_unbraced:NV \pgfsettransformentries \g__spath_smuggle_tl
5264 \tl_gclear:N \g__spath_smuggle_tl
5265 },

```

This is a useful set of styles for drawing a knot diagram.

```

5266 knot/.style~ n~ args={3}{
5267   spath/split~ at~ self~ intersections=#1,
5268   spath/remove~ empty~ components=#1,
5269   spath/insert~ gaps~ after~ components={#1}{#2}{#3},
5270   spath/maybe~ spot~ weld=#1,
5271   spath/render~ components=#1
5272 },
5273 global~ knot/.style~ n~ args={3}{
5274   spath/split~ globally~ at~ self~ intersections=#1,
5275   spath/remove~ empty~ components~ globally=#1,
5276   spath/insert~ gaps~ globally ~after~ components={#1}{#2}{#3},
5277   spath/maybe~ spot~ weld~ globally=#1,
5278   spath/render~ components=#1
5279 },
5280 }

```

This defines a coordinate system that finds a position on a soft path.

```

5281 \tikzdeclarecoordinatesystem{spath}{%
5282   \group_begin:
5283   \tl_set:Nn \l__spath_tmpa_tl {#1}
5284   \tl_trim_spaces:N \l__spath_tmpa_tl
5285
5286   \seq_set_split:NnV \l__spath_tmpa_seq {~} \l__spath_tmpa_tl
5287   \seq_pop_right:NN \l__spath_tmpa_seq \l__spath_tmpb_tl
5288
5289   \tl_set:Nx \l__spath_tmpa_tl { \seq_use:Nn \l__spath_tmpa_seq {~} }
5290   \tl_if_exist:cTF
5291   {
5292     \tl_use:N \l__spath_prefix_tl
5293     \tl_use:N \l__spath_tmpa_tl
5294     \tl_use:N \l__spath_suffix_tl
5295   }
5296   {
5297
5298     \tl_set_eq:Nc
5299     \l__spath_tmpa_tl
5300     {

```

```

5301     \tl_use:N \l__spath_prefix_tl
5302     \tl_use:N \l__spath_tmpa_tl
5303     \tl_use:N \l__spath_suffix_tl
5304   }
5305
5306   \tl_if_empty:NTF \l__spath_tmpa_tl
5307   {
5308     \tl_gset_eq:NN \g__spath_smuggle_tl \pgfpointorigin
5309   }
5310   {
5311     \spath_reallength:NV \l__spath_tmpa_int \l__spath_tmpa_tl
5312     \tl_set:Nx \l__spath_tmpb_tl
5313     {\fp_to_decimal:n {(\l__spath_tmpb_tl) * (\l__spath_tmpa_int)}}
5314     \spath_point_at:NVV \l__spath_tmpc_tl \l__spath_tmpa_tl \l__spath_tmpb_tl
5315
5316     \tl_clear:N \l__spath_tmpd_tl
5317     \tl_put_right:Nn \l__spath_tmpd_tl {\pgf@x=}
5318     \tl_put_right:Nx \l__spath_tmpd_tl {\tl_item:Nn \l__spath_tmpc_tl {1}}
5319     \tl_put_right:Nn \l__spath_tmpd_tl {\relax}
5320     \tl_put_right:Nn \l__spath_tmpd_tl {\pgf@y=}
5321     \tl_put_right:Nx \l__spath_tmpd_tl {\tl_item:Nn \l__spath_tmpc_tl {2}}
5322     \tl_put_right:Nn \l__spath_tmpd_tl {\relax}
5323     \tl_gset_eq:NN \g__spath_smuggle_tl \l__spath_tmpd_tl
5324   }
5325   }
5326   {
5327     \msg_warning:nx { spath3 } { missing soft path } { \tl_use:N \l__spath_tmpa_tl }
5328     \tl_gset_eq:NN \g__spath_smuggle_tl \pgfpointorigin
5329   }
5330   \group_end:
5331   \use:c {pgf@process}{%
5332     \tl_use:N \g__spath_smuggle_tl
5333     \pgftransforminvert
5334     \use:c {pgf@pos@transform@glob}
5335   }
5336 }
5337
5338 \ExplSyntaxOff

```

5 The Calligraphy Package

```

5339 <@@=cal>

```

5.1 Initialisation

```

5340 \RequirePackage{spath3}
5341 \ExplSyntaxOn
5342
5343 \tl_new:N \l__cal_tmpa_tl
5344 \tl_new:N \l__cal_tmpb_tl
5345 \tl_new:N \l__cal_tmp_path_tl
5346 \tl_new:N \l__cal_tmp_rpath_tl
5347 \tl_new:N \l__cal_tmp_rpathb_tl
5348 \tl_new:N \l__cal_tmp_patha_tl

```

```

5349
5350 \seq_new:N \l__cal_tmpa_seq
5351
5352 \int_new:N \l__cal_tmpa_int
5353 \int_new:N \l__cal_tmpb_int
5354 \int_new:N \g__cal_path_component_int
5355 \int_new:N \g__cal_label_int
5356
5357 \fp_new:N \l__cal_tmpa_fp
5358 \fp_new:N \l__cal_tmpb_fp
5359 \fp_new:N \l__cal_tmpe_fp
5360 \fp_new:N \l__cal_tmpe_fp
5361 \fp_new:N \l__cal_tmpe_fp
5362
5363 \dim_new:N \l__cal_tmpa_dim
5364 \dim_new:N \l__cal_tmpb_dim
5365 \dim_new:N \l__cal_tmpe_dim
5366 \dim_new:N \l__cal_tmpe_dim
5367 \dim_new:N \l__cal_tmpe_dim
5368 \dim_new:N \l__cal_tmpe_dim
5369 \dim_new:N \l__cal_tmpe_dim
5370 \dim_new:N \l__cal_tmpe_dim
5371
5372 \bool_new:N \l__cal_annotate_bool
5373 \bool_new:N \l__cal_taper_start_bool
5374 \bool_new:N \l__cal_taper_end_bool
5375 \bool_new:N \l__cal_taperable_bool
5376
5377 \dim_new:N \l__cal_taper_width_dim
5378 \dim_new:N \l__cal_line_width_dim
5379
5380 \bool_set_true:N \l__cal_taper_start_bool
5381 \bool_set_true:N \l__cal_taper_end_bool
5382
5383 \cs_generate_variant:Nn \tl_put_right:Nn {Nv}
5384
5385 \msg_new:nnn { calligraphy } { undefined pen } { The~ pen~ "#1"~ is~ not~ defined. }

```

5.2 TikZ Keys

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

```

5386 \tikzset{
5387   define~pen/.code={
5388     \tikzset{pen~name=#1}
5389     \pgf@relevantforpicturesizefalse
5390     \tikz@addmode{
5391       \pgfsyssoftpath@getcurrentpath\l__cal_tmpa_tl
5392       \spath_components_to_seq:N \l__cal_tmpa_seq \l__cal_tmpa_tl
5393       \seq_gclear_new:c {g__cal_pen_\pgfkeysvalueof{/tikz/pen~name}_seq}
5394       \seq_gset_eq:cN
5395       {g__cal_pen_\pgfkeysvalueof{/tikz/pen~name}_seq} \l__cal_tmpa_seq
5396       \pgfusepath{discard}%
5397     }
5398   },

```

```

5399 define~pen/.default={default},
5400 use~pen/.code={
5401   \tikzset{pen~name=#1}
5402   \int_gzero:N \g__cal_path_component_int
5403   \cs_set_eq:NN \pgfpathmoveto \cal_moveto:n
5404   \tikz@addmode{
5405     \pgfsyssoftpath@getcurrentpath\l__cal_tmpa_tl
5406     \spath_components_to_seq:NV \l__cal_tmpa_seq \l__cal_tmpa_tl
5407     \tl_if_exist:cTF {g__cal_pen_\pgfkeysvalueof{/tikz/pen~name}_seq}
5408     {
5409       \cal_path_create:Nc \l__cal_tmpa_seq
5410       {g__cal_pen_\pgfkeysvalueof{/tikz/pen~name}_seq}
5411     }
5412     {
5413       \msg_warning:nmx { calligraphy } { undefined pen }
5414       { \pgfkeysvalueof{/tikz/pen~name} }
5415     }
5416   }
5417 },
5418 use~pen/.default={default},
5419 pen~name/.initial={default},
5420 copperplate/.style={pen~name=copperplate},
5421 pen~colour/.initial={black},
5422 weight/.is~choice,
5423 weight/heavy/.style={
5424   line~width=\pgfkeysvalueof{/tikz/heavy~line~width},
5425   taper~width=\pgfkeysvalueof{/tikz/light~line~width},
5426 },
5427 weight/light/.style={
5428   line~width=\pgfkeysvalueof{/tikz/light~line~width},
5429   taper~width=0pt,
5430 },
5431 heavy/.style={
5432   weight=heavy
5433 },
5434 light/.style={
5435   weight=light
5436 },
5437 heavy~line~width/.initial=2pt,
5438 light~line~width/.initial=1pt,
5439 taper/.is~choice,
5440 taper/.default=both,
5441 taper/none/.style={
5442   taper~start=false,
5443   taper~end=false,
5444 },
5445 taper/both/.style={
5446   taper~start=true,
5447   taper~end=true,
5448 },
5449 taper/start/.style={
5450   taper~start=true,
5451   taper~end=false,
5452 },

```

```

5453 taper/end/.style={
5454     taper~start=false,
5455     taper~end=true,
5456 },
5457 taper~start/.code={
5458     \tl_if_eq:nnTF {#1} {true}
5459     {
5460         \bool_set_true:N \l__cal_taper_start_bool
5461     }
5462     {
5463         \bool_set_false:N \l__cal_taper_start_bool
5464     }
5465 },
5466 taper~start/.default={true},
5467 taper~end/.code={
5468     \tl_if_eq:nnTF {#1} {true}
5469     {
5470         \bool_set_true:N \l__cal_taper_end_bool
5471     }
5472     {
5473         \bool_set_false:N \l__cal_taper_end_bool
5474     }
5475 },
5476 taper~end/.default={true},
5477 taper~width/.code={\dim_set:Nn \l__cal_taper_width_dim {#1}},
5478 nib~style/.code~2~args={
5479     \tl_clear_new:c {l__cal_nib_style_#1}
5480     \tl_set:cn {l__cal_nib_style_#1} {#2}
5481 },
5482 stroke~style/.code~2~args={
5483     \tl_clear_new:c {l__cal_stroke_style_#1}
5484     \tl_set:cn {l__cal_stroke_style_#1} {#2}
5485 },
5486 this~stroke~style/.code={
5487     \tl_clear_new:c
5488     {l__cal_stroke_inline_style_ \int_use:N \g__cal_path_component_int}
5489     \tl_set:cn
5490     {l__cal_stroke_inline_style_ \int_use:N \g__cal_path_component_int} {#1}
5491 },
5492 annotate/.style={
5493     annotate~if,
5494     annotate~reset,
5495     annotation~style/.update~value={#1},
5496 },
5497 annotate~if/.default={true},
5498 annotate~if/.code={
5499     \tl_if_eq:nnTF {#1} {true}
5500     {
5501         \bool_set_true:N \l__cal_annotate_bool
5502     }
5503     {
5504         \bool_set_false:N \l__cal_annotate_bool
5505     }
5506 },

```



```

5507 annotate~reset/.code={
5508     \int_gzero:N \g__cal_label_int
5509 },
5510 annotation~style/.initial={draw,->},
5511 annotation~shift/.initial={{0,lex}},
5512 every~annotation~node/.initial={anchor=south-west},
5513 annotation~node~style/.code~2~args={
5514     \tl_clear_new:c {l__cal_annotation_style_ #1 _tl}
5515     \tl_set:cn {l__cal_annotation_style_ #1 _tl}{#2}
5516 },
5517 tl~use:N/.code={
5518     \exp_args:Nv \pgfkeysalso #1
5519 },
5520 tl~use:c/.code={
5521     \tl_if_exist:cT {#1}
5522     {
5523         \exp_args:Nv \pgfkeysalso {#1}
5524     }
5525 },
5526 /handlers/.update~style/.code={
5527     \tl_if_eq:nnF {#1} {\pgfkeysnovalue}
5528     {
5529         \pgfkeys{\pgfkeyscurrentpath/.code=\pgfkeysalso{#1}}
5530     }
5531 },
5532 /handlers/.update~value/.code={
5533     \tl_if_eq:nnF {#1} {\pgfkeysnovalue}
5534     {
5535         \pgfkeyssetvalue{\pgfkeyscurrentpath}{#1}
5536     }
5537 },
5538 }

```

Some wrappers around the TikZ keys.

```

5539 \NewDocumentCommand \pen { 0{} }
5540 {
5541     \path[define~ pen,every~ calligraphy~ pen/.try,#1]
5542 }
5543
5544 \NewDocumentCommand \definepen { 0{} }
5545 {
5546     \tikz \path[define~ pen,every~ calligraphy~ pen/.try,#1]
5547 }
5548
5549 \NewDocumentCommand \calligraphy { 0{} }
5550 {
5551     \path[use~ pen,every~ calligraphy/.try,#1]
5552 }

```

5.3 The Path Creation

`\cal_path_create:NN` This is the main command for creating the calligraphic paths. First argument is the given path Second argument is the pen path

```

5553 \cs_new_protected_nopar:Npn \cal_path_create:NN #1#2

```

```

5554 {
5555   \int_zero:N \l__cal_tmpa_int
5556   \seq_map_inline:Nn #1
5557   {
5558     \int_compare:nT {\tl_count:n {##1} > 3}
5559     {
5560
5561       \int_incr:N \l__cal_tmpa_int
5562       \int_zero:N \l__cal_tmpb_int
5563
5564       \tl_set:Nn \l__cal_tmp_path_tl {##1}
5565       \spath_open:N \l__cal_tmp_path_tl
5566       \spath_reverse:NV \l__cal_tmp_rpath_tl \l__cal_tmp_path_tl
5567
5568       \seq_map_inline:Nn #2
5569       {
5570         \int_incr:N \l__cal_tmpb_int
5571         \group_begin:
5572         \pgfsys@beginscope
5573         \cal_apply_style:c {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_int}
5574         \cal_apply_style:c {l__cal_stroke_inline_style_ \int_use:N \l__cal_tmpa_int}
5575         \cal_apply_style:c {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}
5576
5577         \spath_initialpoint:Nn \l__cal_tmpa_tl {####1}
5578         \tl_set_eq:NN \l__cal_tmp_patha_tl \l__cal_tmp_path_tl
5579         \spath_translate:NV \l__cal_tmp_patha_tl \l__cal_tmpa_tl
5580
5581         \int_compare:nTF {\tl_count:n {####1} == 3}
5582         {
5583           \cal_at_least_three:N \l__cal_tmp_patha_tl
5584           \spath_protocol_path:V \l__cal_tmp_patha_tl
5585
5586           \tikz@options
5587           \dim_set:Nn \l__cal_line_width_dim {\pgflinewidth}
5588           \cal_maybe_taper:N \l__cal_tmp_patha_tl
5589         }
5590         {
5591           \spath_weld:Nn \l__cal_tmp_patha_tl {####1}
5592           \spath_weld:NV \l__cal_tmp_patha_tl \l__cal_tmp_rpath_tl
5593           \spath_reverse:Nn \l__cal_tmp_rpathb_tl {####1}
5594           \spath_weld:NV \l__cal_tmp_patha_tl \l__cal_tmp_rpathb_tl
5595
5596           \tl_clear:N \l__cal_tmpa_tl
5597           \tl_set:Nn \l__cal_tmpa_tl
5598           {
5599             fill=\pgfkeysvalueof{/tikz/pen-colour},
5600             draw=none
5601           }
5602           \tl_if_exist:cT {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_int}
5603           {
5604             \tl_put_right:Nv \l__cal_tmpa_tl
5605             {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_int}
5606           }
5607           \tl_if_exist:cT {l__cal_stroke_inline_style_ \int_use:N \l__cal_tmpa_int}

```

```

5608     {
5609         \tl_put_right:Nn \l__cal_tmpa_tl {,}
5610         \tl_put_right:Nv \l__cal_tmpa_tl
5611         {l__cal_stroke_inline_style_ \int_use:N \l__cal_tmpa_int}
5612     }
5613     \tl_if_exist:cT {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}
5614     {
5615         \tl_put_right:Nn \l__cal_tmpa_tl {,}
5616         \tl_put_right:Nv \l__cal_tmpa_tl
5617         {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}
5618     }
5619     \spath_tikz_path:VV \l__cal_tmpa_tl \l__cal_tmp_patha_tl
5620 }
5621 \pgfsys@endscope
5622 \group_end:
5623 }
5624
5625 \bool_if:NT \l__cal_annotate_bool
5626 {
5627     \seq_get_right:NN #2 \l__cal_tmpa_tl
5628     \spath_finalpoint:NV \l__cal_tmpa_tl \l__cal_tmpa_tl
5629     \spath_translate:NV \l__cal_tmp_path_tl \l__cal_tmpa_tl
5630     \tikz@scan@one@point
5631     \pgfutil@firstofone
5632     \pgfkeysvalueof{/tikz/annotation~shift}
5633
5634     \spath_translate:Nnn \l__cal_tmp_path_tl {\pgf@x} {\pgf@y}
5635
5636     \pgfkeysgetvalue{/tikz/annotation~style}{\l__cal_tmpa_tl}
5637     \spath_tikz_path:VV \l__cal_tmpa_tl \l__cal_tmp_path_tl
5638
5639     \spath_finalpoint:NV \l__cal_tmpa_tl \l__cal_tmp_path_tl
5640
5641     \exp_last_unbraced:NV \pgfqpoint \l__cal_tmpa_tl
5642     \begin{scope}[reset~ cm]
5643     \node[
5644         every~annotation~node/.try,
5645         tl~use:c = {l__cal_annotation_style_ \int_use:N \l__cal_tmpa_int _tl}
5646     ] at (\pgf@x,\pgf@y) {\int_use:N \l__cal_tmpa_int};
5647     \end{scope}
5648 }
5649 }
5650 }
5651 }
5652 \cs_generate_variant:Nn \cal_path_create:NN {Nc}

```

(End definition for \cal_path_create:NN.)

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

```

5653 \cs_new_eq:NN \cal_orig_moveto:n \pgfpathmoveto
5654 \cs_new_nopar:Npn \cal_moveto:n #1
5655 {
5656     \int_gincr:N \g__cal_path_component_int

```

```

5657 \cal_orig_moveto:n {#1}
5658 }

```

(End definition for \cal_moveto:n.)

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

```

5659 \cs_new_nopar:Npn \cal_apply_style:N #1
5660 {
5661   \tl_if_exist:NT #1 {
5662     \exp_args:NV \tikzset #1
5663   }
5664 }
5665 \cs_generate_variant:Nn \cal_apply_style:N {c}

```

(End definition for \cal_apply_style:N.)

`\cal_at_least_three:Nn` A tapered path has to have at least three components. This figures out if it is necessary and sets up the splitting.

```

5666 \cs_new_protected_nopar:Npn \cal_at_least_three:Nn #1#2
5667 {
5668   \spath_reallength:Nn \l__cal_tmpa_int {#2}
5669   \tl_clear:N \l__cal_tmpb_tl
5670   \tl_set:Nn \l__cal_tmpb_tl {#2}
5671   \int_compare:nTF {\l__cal_tmpa_int = 1}
5672   {
5673     \spath_split_at:Nn \l__cal_tmpb_tl {2/3}
5674     \spath_split_at:Nn \l__cal_tmpb_tl {1/2}
5675   }
5676   {
5677     \int_compare:nT {\l__cal_tmpa_int = 2}
5678     {
5679       \spath_split_at:Nn \l__cal_tmpb_tl {1.5}
5680       \spath_split_at:Nn \l__cal_tmpb_tl {.5}
5681     }
5682   }
5683   \tl_set_eq:NN #1 \l__cal_tmpb_tl
5684 }
5685 \cs_generate_variant:Nn \cal_at_least_three:Nn {NV}
5686 \cs_new_protected_nopar:Npn \cal_at_least_three:N #1
5687 {
5688   \cal_at_least_three:NV #1#1
5689 }
5690 \cs_generate_variant:Nn \cal_at_least_three:N {c}

```

(End definition for \cal_at_least_three:Nn.)

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

```

5691 \cs_new_protected_nopar:Npn \cal_maybe_taper:N #1
5692 {
5693   \tl_set_eq:NN \l__cal_tmpa_tl #1
5694
5695   \bool_if:NT \l__cal_taper_start_bool
5696   {
5697
5698     \dim_set:Nn \l__cal_tmpa_dim {\tl_item:Nn \l__cal_tmpa_tl {2}}

```

```

5699 \dim_set:Nn \l__cal_tmpb_dim {\tl_item:Nn \l__cal_tmpa_tl {3}}
5700 \tl_set:Nx \l__cal_tmpb_tl {\tl_item:Nn \l__cal_tmpa_tl {4}}
5701
5702 \tl_case:NnF \l__cal_tmpb_tl
5703 {
5704   \c_spath_lineto_tl
5705   {
5706
5707     \bool_set_true:N \l__cal_taperable_bool
5708     \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {5}}
5709     \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {6}}
5710     \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpa_dim + \l__cal_tmph_dim)/3}
5711     \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpe_dim + \l__cal_tmph_dim)/3}
5712     \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpe_dim + \l__cal_tmph_dim)/3}
5713     \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpe_dim + \l__cal_tmph_dim)/3}
5714     \prg_replicate:nn {4}
5715     {
5716       \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
5717     }
5718     \tl_put_left:NV \l__cal_tmpa_tl \c_spath_moveto_tl
5719   }
5720   \c_spath_curvetoa_tl
5721   {
5722     \bool_set_true:N \l__cal_taperable_bool
5723     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {5}}
5724     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {6}}
5725     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {8}}
5726     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {9}}
5727     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {11}}
5728     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {12}}
5729     \prg_replicate:nn {10}
5730     {
5731       \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
5732     }
5733     \tl_put_left:NV \l__cal_tmpa_tl \c_spath_moveto_tl
5734   }
5735 }
5736 {
5737   \bool_set_false:N \l__cal_taperable_bool
5738 }
5739
5740 \bool_if:NT \l__cal_taperable_bool
5741 {
5742   \__cal_taper_aux:
5743 }
5744
5745 }
5746
5747 \bool_if:NT \l__cal_taper_end_bool
5748 {
5749
5750   \dim_set:Nn \l__cal_tmpa_dim {\tl_item:Nn \l__cal_tmpa_tl {-2}}
5751   \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-1}}
5752   \tl_set:Nx \l__cal_tmpe_tl {\tl_item:Nn \l__cal_tmpa_tl {-3}}

```

```

5753
5754 \tl_case:NnF \l__cal_tmpb_tl
5755 {
5756   \c_spath_lineto_tl
5757   {
5758
5759     \bool_set_true:N \l__cal_taperable_bool
5760     \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {-5}}
5761     \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {-4}}
5762     \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpe_dim + \l__cal_tmph_dim)/3}
5763     \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpe_dim + \l__cal_tmph_dim)/3}
5764     \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpe_dim + \l__cal_tmph_dim)/3}
5765     \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpe_dim + \l__cal_tmph_dim)/3}
5766     \tl_reverse:N \l__cal_tmpa_tl
5767     \prg_replicate:nn {3}
5768     {
5769       \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
5770     }
5771     \tl_reverse:N \l__cal_tmpa_tl
5772   }
5773   \c_spath_curveto_tl
5774   {
5775     \bool_set_true:N \l__cal_taperable_bool
5776     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-5}}
5777     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-4}}
5778     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-8}}
5779     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-7}}
5780     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-11}}
5781     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-10}}
5782     \tl_reverse:N \l__cal_tmpa_tl
5783     \prg_replicate:nn {9}
5784     {
5785       \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
5786     }
5787     \tl_reverse:N \l__cal_tmpa_tl
5788   }
5789 }
5790 {
5791   \bool_set_false:N \l__cal_taperable_bool
5792 }
5793
5794 \bool_if:NT \l__cal_taperable_bool
5795 {
5796   \__cal_taper_aux:
5797 }
5798
5799 }
5800
5801 \pgfsyssoftpath@setcurrentpath\l__cal_tmpa_tl
5802 \pgfsetstrokecolor{\pgfkeysvalueof{/tikz/pen~colour}}
5803 \pgfusepath{stroke}
5804
5805 }

```

(End definition for \cal_maybe_taper:N.)

`_cal_taper_aux:` Auxiliary macro to avoid unnecessary code duplication.

```

5806 \cs_new_protected_nopar:Npn \_cal_taper_aux:
5807 {
5808   \tl_clear:N \l__cal_tmpe_tl
5809   \tl_put_right:NV \l__cal_tmpe_tl \c_spath_moveto_tl
5810
5811   \fp_set:Nn \l__cal_tmpe_fp
5812   {
5813     \l__cal_tmpe_dim - \l__cal_tmpe_dim
5814   }
5815   \fp_set:Nn \l__cal_tmpe_fp
5816   {
5817     \l__cal_tmpe_dim - \l__cal_tmpe_dim
5818   }
5819   \fp_set:Nn \l__cal_tmpe_fp
5820   {
5821     ( $\l__cal_tmpe\_fp^2 + \l__cal_tmpe\_fp^2$ ).5
5822   }
5823
5824   \fp_set:Nn \l__cal_tmpe_fp
5825   {
5826     .5*\l__cal_taper_width_dim
5827     *
5828     \l__cal_tmpe_fp / \l__cal_tmpe_fp
5829   }
5830   \fp_set:Nn \l__cal_tmpe_fp
5831   {
5832     .5*\l__cal_taper_width_dim
5833     *
5834     \l__cal_tmpe_fp / \l__cal_tmpe_fp
5835   }
5836
5837   \fp_set:Nn \l__cal_tmpe_fp
5838   {
5839     \l__cal_tmpe_dim - \l__cal_tmpe_dim
5840   }
5841   \fp_set:Nn \l__cal_tmpe_fp
5842   {
5843     \l__cal_tmpe_dim - \l__cal_tmpe_dim
5844   }
5845   \fp_set:Nn \l__cal_tmpe_fp
5846   {
5847     ( $\l__cal_tmpe\_fp^2 + \l__cal_tmpe\_fp^2$ ).5
5848   }
5849
5850   \fp_set:Nn \l__cal_tmpe_fp
5851   {
5852     .5*\l__cal_line_width_dim
5853     *
5854     \l__cal_tmpe_fp / \l__cal_tmpe_fp
5855   }
5856   \fp_set:Nn \l__cal_tmpe_fp
5857   {
5858     .5*\l__cal_line_width_dim

```

```

5859      *
5860      \l__cal_tmpd_fp / \l__cal_tmpe_fp
5861    }
5862
5863    \tl_put_right:Nx \l__cal_tmpe_tl
5864    {
5865      {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim}}
5866      {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim}}
5867    }
5868
5869    \tl_put_right:NV \l__cal_tmpe_tl \c_spath_curvetoa_tl
5870
5871    \tl_put_right:Nx \l__cal_tmpe_tl
5872    {
5873      {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim}}
5874      {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim}}
5875    }
5876
5877    \tl_put_right:NV \l__cal_tmpe_tl \c_spath_curvetob_tl
5878
5879    \tl_put_right:Nx \l__cal_tmpe_tl
5880    {
5881      {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim}}
5882      {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim}}
5883    }
5884
5885    \tl_put_right:NV \l__cal_tmpe_tl \c_spath_curveto_tl
5886
5887    \tl_put_right:Nx \l__cal_tmpe_tl
5888    {
5889      {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim}}
5890      {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim}}
5891    }
5892
5893    \tl_put_right:NV \l__cal_tmpe_tl \c_spath_curvetoa_tl
5894
5895    \tl_put_right:Nx \l__cal_tmpe_tl
5896    {
5897      {
5898        \dim_eval:n
5899        {
5900          \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim
5901          - \fp_to_dim:n{ 1.32 * \l__cal_tmpe_fp
5902          }
5903        }
5904      }
5905      {
5906        \dim_eval:n
5907        {
5908          \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim
5909          + \fp_to_dim:n {1.32* \l__cal_tmpe_fp
5910          }
5911        }
5912      }
5913    }

```



```

5913 }
5914
5915 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetob_tl
5916
5917 \tl_put_right:Nx \l__cal_tmpb_tl
5918 {
5919   {
5920     \dim_eval:n
5921     {
5922       -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim
5923       - \fp_to_dim:n {1.32 * \l__cal_tmpe_dim}
5924     }
5925   }
5926 }
5927 {
5928   \dim_eval:n
5929   {
5930     -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim
5931     + \fp_to_dim:n {1.32 * \l__cal_tmpe_dim}
5932   }
5933 }
5934 }
5935 }
5936
5937 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curveto_tl
5938
5939 \tl_put_right:Nx \l__cal_tmpb_tl
5940 {
5941   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
5942   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
5943 }
5944
5945 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetoa_tl
5946
5947 \tl_put_right:Nx \l__cal_tmpb_tl
5948 {
5949   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
5950   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
5951 }
5952
5953 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetob_tl
5954
5955 \tl_put_right:Nx \l__cal_tmpb_tl
5956 {
5957   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
5958   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
5959 }
5960
5961 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curveto_tl
5962
5963 \tl_put_right:Nx \l__cal_tmpb_tl
5964 {
5965   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
5966   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}

```

```

5967 }
5968
5969 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetoa_tl
5970
5971 \tl_put_right:Nx \l__cal_tmpb_tl
5972 {
5973   {
5974     \dim_eval:n
5975     {
5976       -\fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim
5977       + \fp_to_dim:n{ 1.32 * \l__cal_tmpb_fp}
5978     }
5979   }
5980   {
5981     \dim_eval:n
5982     {
5983       -\fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim
5984       - \fp_to_dim:n {1.32* \l__cal_tmpa_fp}
5985     }
5986   }
5987 }
5988
5989 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetob_tl
5990
5991 \tl_put_right:Nx \l__cal_tmpb_tl
5992 {
5993   {
5994     \dim_eval:n
5995     {
5996       \fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim
5997       + \fp_to_dim:n {1.32 * \l__cal_tmpb_fp}
5998     }
5999   }
6000   {
6001     \dim_eval:n
6002     {
6003       \fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim
6004       - \fp_to_dim:n {1.32 * \l__cal_tmpa_fp}
6005     }
6006   }
6007 }
6008
6009 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curveto_tl
6010
6011 \tl_put_right:Nx \l__cal_tmpb_tl
6012 {
6013   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim}}
6014   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim}}
6015 }
6016
6017 \pgfsyssoftpath@setcurrentpath\l__cal_tmpb_tl
6018 \pgfsetfillcolor{\pgfkeysvalueof{/tikz/pen~colour}}
6019 \pgfusepath{fill}
6020 }

```

(End definition for _cal_taper_aux:.)

Defines a copperplate pen.

```
6021 \tl_set:Nn \l__cal_tmpa_tl {\pgfsyssoftpath@movetotoken{Opt}{Opt}}
6022 \spath_components_to_seq:N \l__cal_tmpa_seq \l__cal_tmpa_tl
6023 \seq_gclear_new:N \g__cal_pen_copperplate_seq
6024 \seq_gset_eq:NN \g__cal_pen_copperplate_seq \l__cal_tmpa_seq
```

\CopperplatePath This is used in the decorations section to convert a path to a copperplate path.

```
6025 \DeclareDocumentCommand \CopperplatePath { m }
6026 {
6027   \spath_components_to_seq:N \l__cal_tmpa_seq #1
6028   \cal_path_create:NN \l__cal_tmpa_seq \g__cal_pen_copperplate_seq
6029 }
```

(End definition for \CopperplatePath.)

```
6030 \ExplSyntaxOff
```

5.4 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.

```
6031 \expandafter\ifx\curname pgfdeclarededecoration\endcsname\relax
6032 \else
6033 \pgfdeclarededecoration{calligraphic brace}{brace}%
6034 {%
6035   \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]%
6036   {%
6037     \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}%
6038     \pgfpathmoveto{\pgfpointorigin}%
6039     \pgfpathcurveto%
6040     {%
6041       \pgfpoint%
6042       {.15\pgfdecorationsegmentamplitude}%
6043       {.3\pgfdecorationsegmentamplitude}%
6044     }%
6045     {%
6046       \pgfpoint%
6047       {.5\pgfdecorationsegmentamplitude}%
6048       {.5\pgfdecorationsegmentamplitude}%
6049     }%
6050     {%
6051       \pgfpoint%
6052       {\pgfdecorationsegmentamplitude}%
6053       {.5\pgfdecorationsegmentamplitude}%
6054     }%
6055     {%
6056       \pgftransformxshift%
6057       {+\pgfdecorationsegmentaspect\pgfdecoratedremainingdistance}%
6058       \pgfpathlineto%
6059       {%
6060         \pgfpoint%
6061         {-\pgfdecorationsegmentamplitude}%

```

```

6062         {.5\pgfdecorationsegmentamplitude}%
6063     }%
6064     \pgfpathcurveto
6065     {%
6066         \pgfqpoint%
6067         {- .5\pgfdecorationsegmentamplitude}%
6068         {.5\pgfdecorationsegmentamplitude}%
6069     }%
6070     {%
6071         \pgfqpoint%
6072         {- .15\pgfdecorationsegmentamplitude}%
6073         {.7\pgfdecorationsegmentamplitude}%
6074     }%
6075     {%
6076         \pgfqpoint%
6077         {0\pgfdecorationsegmentamplitude}%
6078         {1\pgfdecorationsegmentamplitude}%
6079     }%
6080     \pgfpathmoveto%
6081     {%
6082         \pgfqpoint%
6083         {0\pgfdecorationsegmentamplitude}%
6084         {1\pgfdecorationsegmentamplitude}%
6085     }%
6086     \pgfpathcurveto%
6087     {%
6088         \pgfqpoint%
6089         {.15\pgfdecorationsegmentamplitude}%
6090         {.7\pgfdecorationsegmentamplitude}%
6091     }%
6092     {%
6093         \pgfqpoint%
6094         {.5\pgfdecorationsegmentamplitude}%
6095         {.5\pgfdecorationsegmentamplitude}%
6096     }%
6097     {%
6098         \pgfqpoint%
6099         {\pgfdecorationsegmentamplitude}%
6100         {.5\pgfdecorationsegmentamplitude}%
6101     }%
6102 }%
6103 {%
6104     \pgftransformxshift{+\pgfdecoratedremainingdistance}%
6105     \pgfpathlineto%
6106     {%
6107         \pgfqpoint%
6108         {-\pgfdecorationsegmentamplitude}%
6109         {.5\pgfdecorationsegmentamplitude}%
6110     }%
6111     \pgfpathcurveto%
6112     {%
6113         \pgfqpoint%
6114         {- .5\pgfdecorationsegmentamplitude}%
6115         {.5\pgfdecorationsegmentamplitude}%

```

```

6116     }%
6117     {%
6118         \pgfqpoint%
6119         {- .15\pgfdecorationsegmentamplitude}%
6120         {.3\pgfdecorationsegmentamplitude}%
6121     }%
6122     {\pgfqpoint{0pt}{0pt}}%
6123 }%
6124 \tikzset{%
6125     taper width=.5\pgflinewidth,%
6126     taper%
6127 }%
6128 \pgfsyssoftpath@getcurrentpath\cal@tmp@path%
6129 \CopperplatePath{\cal@tmp@path}%
6130 }%
6131 \state{final}{}%
6132 }%

```

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

```

6133 \pgfdeclaredecoration{calligraphic straight parenthesis}{brace}
6134 {
6135     \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]%
6136     {%
6137         \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}%
6138         \pgfpathmoveto{\pgfpointorigin}%
6139         \pgfpathcurveto%
6140         {%
6141             \pgfqpoint%
6142             {.76604\pgfdecorationsegmentamplitude}%
6143             {.64279\pgfdecorationsegmentamplitude}%
6144         }%
6145         {%
6146             \pgfqpoint%
6147             {2.3333\pgfdecorationsegmentamplitude}%
6148             {\pgfdecorationsegmentamplitude}%
6149         }%
6150         {%
6151             \pgfqpoint%
6152             {3.3333\pgfdecorationsegmentamplitude}%
6153             {\pgfdecorationsegmentamplitude}%
6154         }%
6155         {%
6156             \pgftransformxshift{+\pgfdecoratedremainingdistance}%
6157             \pgfpathlineto%
6158             {%
6159                 \pgfqpoint%
6160                 {-3.3333\pgfdecorationsegmentamplitude}%
6161                 {\pgfdecorationsegmentamplitude}%
6162             }%
6163             \pgfpathcurveto%
6164             {%
6165                 \pgfqpoint%
6166                 {-2.3333\pgfdecorationsegmentamplitude}%

```

```

6167         {\pgfdecorationsegmentamplitude}%
6168     }%
6169     {%
6170         \pgfqpoint%
6171         {- .76604\pgfdecorationsegmentamplitude}%
6172         {.64279\pgfdecorationsegmentamplitude}%
6173     }%
6174     {\pgfqpoint{0pt}{0pt}}%
6175 }%
6176 \tikzset{%
6177     taper width=.5\pgflinewidth,%
6178     taper%
6179 }%
6180 \pgfsyssoftpath@getcurrentpath\cal@tmp@path%
6181 \CopperplatePath{\cal@tmp@path}%
6182 }%
6183 \state{final}{}%
6184 }

The third is a curved parenthesis.
6185 \pgfdeclaredcoration{calligraphic curved parenthesis}{brace}
6186 {
6187     \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]%
6188     {%
6189         \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}%
6190         \pgfpathmoveto{\pgfpointorigin}%
6191         \pgf@xa=\pgfdecoratedremainingdistance\relax%
6192         \advance\pgf@xa by -1.5890\pgfdecorationsegmentamplitude\relax%
6193         \edef\cgrphy@xa{\the\pgf@xa}%
6194         \pgfpathcurveto%
6195         {%
6196             \pgfqpoint%
6197             {1.5890\pgfdecorationsegmentamplitude}%
6198             {1.3333\pgfdecorationsegmentamplitude}%
6199         }%
6200         {\pgfqpoint{\cgrphy@xa}{1.3333\pgfdecorationsegmentamplitude}}%
6201         {\pgfqpoint{\pgfdecoratedremainingdistance}{0pt}}%
6202         \tikzset{%
6203             taper width=.5\pgflinewidth,%
6204             taper%
6205         }%
6206         \pgfsyssoftpath@getcurrentpath\cal@tmp@path%
6207         \CopperplatePath{\cal@tmp@path}%
6208     }%
6209     \state{final}{}%
6210 }

End the conditional for if pgfdecoration module is loaded
6211 \fi

```

6 Drawing Knots

```

6212 <@@=knot>

```

6.1 Initialisation

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

```
6213 \RequirePackage{spath3}
6214 \usetikzlibrary{intersections,spath3}
6215
6216 \ExplSyntaxOn
6217
6218 \tl_new:N \l__knot_tmpa_tl
6219 \tl_new:N \l__knot_tmpb_tl
6220 \tl_new:N \l__knot_tmpc_tl
6221 \tl_new:N \l__knot_tmpd_tl
6222 \tl_new:N \l__knot_tmpg_tl
6223 \tl_new:N \l__knot_redraws_tl
6224 \tl_new:N \l__knot_clip_width_tl
6225 \tl_new:N \l__knot_name_tl
6226 \tl_new:N \l__knot_node_tl
6227 \tl_new:N \l__knot_aux_tl
6228 \tl_new:N \l__knot_auxa_tl
6229 \tl_new:N \l__knot_prefix_tl
6230
6231 \seq_new:N \l__knot_segments_seq
6232
6233 \int_new:N \l__knot_tmpa_int
6234 \int_new:N \l__knot_strands_int
6235 \int_new:N \g__knot_intersections_int
6236 \int_new:N \g__knot_filaments_int
6237 \int_new:N \l__knot_component_start_int
6238
6239 \fp_new:N \l__knot_tmpa_fp
6240 \fp_new:N \l__knot_tmpb_fp
6241
6242 \dim_new:N \l__knot_tmpa_dim
6243 \dim_new:N \l__knot_tmpb_dim
6244 \dim_new:N \l__knot_tolerance_dim
6245 \dim_new:N \l__knot_redraw_tolerance_dim
6246 \dim_new:N \l__knot_clip_bg_radius_dim
6247 \dim_new:N \l__knot_clip_draw_radius_dim
6248
6249 \bool_new:N \l__knot_draft_bool
6250 \bool_new:N \l__knot_ignore_ends_bool
6251 \bool_new:N \l__knot_self_intersections_bool
6252 \bool_new:N \l__knot_splits_bool
6253 \bool_new:N \l__knot_super_draft_bool
6254
6255 \bool_new:N \l__knot_prepend_prev_bool
6256 \bool_new:N \l__knot_append_next_bool
6257 \bool_new:N \l__knot_skip_bool
6258 \bool_new:N \l__knot_save_bool
6259
6260 \seq_new:N \g__knot_nodes_seq
6261
6262 \bool_set_true:N \l__knot_ignore_ends_bool
```

Configuration is via TikZ keys and styles.

```

6263 \tikzset{
6264   spath/prefix/knot/.style={
6265     spath/set~ prefix=knot strand,
6266   },
6267   spath/suffix/knot/.style={
6268     spath/set~ suffix={},
6269   },
6270   knot/.code={
6271     \tl_if_eq:nnTF {#1} {none}
6272     {
6273       \tikz@addmode{\tikz@mode@doublefalse}
6274     }
6275     {
6276       \tikz@addmode{\tikz@mode@doubletrue}
6277       \tl_if_eq:nnTF {\pgfkeysnovalue} {#1}
6278       {
6279         \tikz@addoption{\pgfsetinnerstrokecolor{.}}
6280       }
6281       {
6282         \pgfsetinnerstrokecolor{#1}
6283       }
6284       \tikz@addoption{
6285         \pgfsetstrokecolor{knotbg}
6286       }
6287       \tl_set:Nn \tikz@double@setup{
6288         \pgfsetinnerlinewidth{\pgflinewidth}
6289         \pgfsetlinewidth{\dim_eval:n {\tl_use:N \l__knot_gap_tl \pgflinewidth}}
6290       }
6291     }
6292   },
6293   knot~ gap/.store~ in=\l__knot_gap_tl,
6294   knot~ gap=3,
6295   knot~ diagram/.is~family,
6296   knot~ diagram/.unknown/.code={
6297     \tl_set_eq:NN \l__knot_tmpa_tl \pgfkeyscurrentname
6298     \pgfkeysalso{
6299       /tikz/\l__knot_tmpa_tl=#1
6300     }
6301   },
6302   background~ colour/.code={%
6303     \colorlet{knotbg}{#1}%
6304   },
6305   background~ color/.code={%
6306     \colorlet{knotbg}{#1}%
6307   },
6308   background~ colour=white,
6309   knot~ diagram,
6310   name/.store~ in=\l__knot_name_tl,
6311   name={knot},
6312   save~ intersections/.is~ choice,
6313   save~ intersections/.default=true,
6314   save~ intersections/true/.code={
6315     \bool_set_true:N \l__knot_save_bool

```



```

6316 },
6317 save~ intersections/false/.code={
6318   \bool_set_false:N \l__knot_save_bool
6319 },
6320 every~ strand/.style={draw},
6321 ignore~ endpoint~ intersections/.code={
6322   \tl_if_eq:nnTF {#1} {true}
6323   {
6324     \bool_set_true:N \l__knot_ignore_ends_bool
6325   }
6326   {
6327     \bool_set_false:N \l__knot_ignore_ends_bool
6328   }
6329 },
6330 ignore~ endpoint~ intersections/.default=true,
6331 consider~ self~ intersections/.is~choice,
6332 consider~ self~ intersections/true/.code={
6333   \bool_set_true:N \l__knot_self_intersections_bool
6334   \bool_set_true:N \l__knot_splits_bool
6335 },
6336 consider~ self~ intersections/false/.code={
6337   \bool_set_false:N \l__knot_self_intersections_bool
6338   \bool_set_false:N \l__knot_splits_bool
6339 },
6340 consider~ self~ intersections/no~ splits/.code={
6341   \bool_set_true:N \l__knot_self_intersections_bool
6342   \bool_set_false:N \l__knot_splits_bool
6343 },
6344 consider~ self~ intersections/.default={true},
6345 clip~ radius/.code={
6346   \dim_set:Nn \l__knot_clip_bg_radius_dim {#1}
6347   \dim_set:Nn \l__knot_clip_draw_radius_dim {#1+2pt}
6348 },
6349 clip~ draw~ radius/.code={
6350   \dim_set:Nn \l__knot_clip_draw_radius_dim {#1}
6351 },
6352 clip~ background~ radius/.code={
6353   \dim_set:Nn \l__knot_clip_bg_radius_dim {#1}
6354 },
6355 clip~ radius=10pt,
6356 end~ tolerance/.code={
6357   \dim_set:Nn \l__knot_tolerance_dim {#1}
6358 },
6359 end~ tolerance=14pt,
6360 clip/.style={
6361   clip
6362 },
6363 background~ clip/.style={
6364   clip
6365 },
6366 clip~ width/.code={
6367   \tl_set:Nn \l__knot_clip_width_tl {#1}
6368 },
6369 clip~ width=3,

```

```

6370 flip~ crossing/.code={%
6371   \tl_clear_new:c {l__knot_crossing_#1}
6372   \tl_set:cn {l__knot_crossing_#1} {x}
6373 },
6374 ignore~ crossing/.code={%
6375   \tl_clear_new:c {l__knot_ignore_crossing_#1}
6376   \tl_set:cn {l__knot_ignore_crossing_#1} {x}
6377 },
6378 draft~ mode/.is~ choice,
6379 draft~ mode/off/.code={%
6380   \bool_set_false:N \l__knot_draft_bool
6381   \bool_set_false:N \l__knot_super_draft_bool
6382 },
6383 draft~ mode/crossings/.code={%
6384   \bool_set_true:N \l__knot_draft_bool
6385   \bool_set_false:N \l__knot_super_draft_bool
6386 },
6387 draft~ mode/strands/.code={%
6388   \bool_set_true:N \l__knot_draft_bool
6389   \bool_set_true:N \l__knot_super_draft_bool
6390 },
6391 draft/.is~ family,
6392 draft,
6393 crossing~ label/.style={
6394   overlay,
6395   fill=white,
6396   fill~ opacity=.5,
6397   text~ opacity=1,
6398   text=blue,
6399   pin~ edge={blue,<-}
6400 },
6401 strand~ label/.style={
6402   overlay,
6403   circle,
6404   draw=purple,
6405   fill=white,
6406   fill~ opacity=.5,
6407   text~ opacity=1,
6408   text=purple,
6409   inner~ sep=0pt
6410 },
6411 }

```

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

```

6412 \cs_new_nopar:Npn \knot_apply_style:N #1
6413 {
6414   \tl_if_exist:NT #1 {
6415     \exp_args:NV \tikzset #1
6416   }
6417 }
6418 \cs_generate_variant:Nn \knot_apply_style:N {c}

```

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

```

6419 \NewDocumentCommand \flipcrossings {m}
6420 {
6421   \tikzset{knot~ diagram/flip~ crossing/.list={#1}}%
6422 }

```

(End definition for \flipcrossings.)

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

```

6423 \NewDocumentCommand \strand { 0{} }
6424 {
6425   \int_incr:N \l__knot_strands_int
6426   \tl_clear_new:c {l__knot_options_strand \int_use:N \l__knot_strands_int}
6427   \tl_set:cn {l__knot_options_strand \int_use:N \l__knot_strands_int} {#1}
6428   \path[#1,spath/set~ name=knot,spath/save=\int_use:N \l__knot_strands_int]
6429 }

```

(End definition for \strand.)

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

```

6430 \NewDocumentEnvironment{knot} { 0{} }
6431 {
6432   \knot_initialise:n {#1}
6433 }
6434 {
6435   \knot_render:
6436 }

```

(End definition for knot.)

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

```

6437 \cs_new_protected_nopar:Npn \knot_initialise:n #1
6438 {
6439   \tikzset{knot~ diagram/.cd,every~ knot~ diagram/.try,#1}
6440   \int_zero:N \l__knot_strands_int
6441   \tl_clear:N \l__knot_redraws_tl
6442   \seq_gclear:N \g__knot_nodes_seq
6443 }

```

(End definition for \knot_initialise:n.)

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

```

6444 \cs_new_protected_nopar:Npn \knot_render:
6445 {

```

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

```

6446   \pgfscope
6447   \pgftransformreset

```

Set the dimension for deciding when to include neighbouring strands

```

6448   \dim_set:Nn \l__knot_redraw_tolerance_dim {\fp_to_dim:n
6449     {
6450       sqrt(2) * max(\l__knot_clip_bg_radius_dim, \l__knot_clip_draw_radius_dim)
6451     }
6452 }

```

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

```
6453 \int_step_function:nnnN {1} {1} {\l__knot_strands_int} \knot_draw_strand:n
```

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

```
6454 \bool_if:NF \l__knot_super_draft_bool
6455 {
```

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.

```
6456 \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.

```
6457 \bool_if:NTF \l__knot_self_intersections_bool
6458 {
6459   \knot_split_strands:
6460   \int_set_eq:NN \l__knot_tmpa_int \g__knot_filaments_int
6461   \tl_set:Nn \l__knot_prefix_tl {filament}
6462 }
6463 {
6464   \int_set_eq:NN \l__knot_tmpa_int \l__knot_strands_int
6465   \tl_set:Nn \l__knot_prefix_tl {strand}
6466 }
```

Initialise the intersection count.

```
6467 \int_gzero:N \g__knot_intersections_int
```

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

```
6468 \tl_clear:N \l__knot_node_tl
6469 \bool_if:NT \l__knot_draft_bool
6470 {
6471   \tl_set:Nn \l__knot_node_tl {
6472     \exp_not:N \node[coordinate,
6473       pin={
6474         node~ contents={\int_use:N \g__knot_intersections_int},
6475         knot~ diagram/draft/crossing~ label,
6476         knot~ diagram/draft/crossing~
6477         \int_use:N \g__knot_intersections_int \c_space_tl label/.try
6478       ]
6479     }
6480   }
6481 }
```

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

```
6482 \int_step_variable:nnnNn {1} {1} {\l__knot_tmpa_int - 1} \l__knot_tmpa_tl
6483 {
6484   \int_step_variable:nnnNn
6485   {\tl_use:N \l__knot_tmpa_tl + 1}
6486   {1}
6487   {\l__knot_tmpa_int} \l__knot_tmpb_tl
6488   {
6489     \knot_intersections:VV \l__knot_tmpa_tl \l__knot_tmpb_tl
6490   }
6491 }
```

If any redraws were requested, do them here.

```
6492 \tl_use:N \l__knot_redraws_tl
```

Draw the crossing nodes

```
6493 \seq_use:Nn \g__knot_nodes_seq {}
6494 }
```

Close the scope

```
6495 \endpgfscope
6496 }
```

(End definition for \knot_render:.)

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

```
6497 \cs_new_protected_nopar:Npn \knot_draw_strand:n #1
6498 {
6499 \pgfscope
6500 \group_begin:
6501 \spath_bake_round:c {knot strand #1}
6502 \tl_set:Nn \l__knot_tmpa_tl {knot~ diagram/every~ strand/.try,}
6503 \tl_put_right:Nv \l__knot_tmpa_tl {l__knot_options_strand #1}
6504 \tl_put_right:Nn \l__knot_tmpa_tl
6505 {
6506 ,
6507 knot~ diagram/only~ when~ rendering/.try,
6508 only~ when~ rendering/.try
6509 }
6510 \spath_tikz_path:Vv \l__knot_tmpa_tl {knot strand #1}
6511 \group_end:
6512 \endpgfscope
6513 }
6514 \cs_generate_variant:Nn \tl_put_right:Nn {Nv}
```

(End definition for \knot_draw_strand:n.)

`\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.

```
6515 \cs_new_protected_nopar:Npn \knot_draw_labels:n #1
6516 {
6517 \bool_if:NT \l__knot_draft_bool
6518 {
6519 \spath_finalpoint:Nv \l__knot_tmpb_tl {knot strand #1}
6520 \dim_set:Nn \l__knot_tmpa_dim {\tl_item:Nn \l__knot_tmpb_tl {1}}
6521 \dim_set:Nn \l__knot_tmpb_dim {\tl_item:Nn \l__knot_tmpb_tl {2}}
6522 \node[
6523 knot~ diagram/draft/strand~label
6524 ] at (\l__knot_tmpa_dim,\l__knot_tmpb_dim) {#1};
6525 \spath_initialpoint:Nv \l__knot_tmpb_tl {knot strand #1}
6526 \dim_set:Nn \l__knot_tmpa_dim {\tl_item:Nn \l__knot_tmpb_tl {1}}
6527 \dim_set:Nn \l__knot_tmpb_dim {\tl_item:Nn \l__knot_tmpb_tl {2}}
6528 \node[
6529 knot~ diagram/draft/strand~label
6530 ] at (\l__knot_tmpa_dim,\l__knot_tmpb_dim) {#1};
6531 }
6532 \bool_if:nT {
```

```

6533 \l__knot_self_intersections_bool
6534 &&
6535 \l__knot_splits_bool
6536 }
6537 {
6538 \tl_clear:N \l__knot_tmpa_tl
6539 \spath_initialpoint:Nv \l__knot_tmpa_tl {knot strand #1}
6540 \tl_put_left:Nv \l__knot_tmpa_tl \c_spath_moveto_tl
6541 \spath_segments_to_seq:Nv \l__knot_segments_seq {knot strand #1}
6542 \seq_map_function:NN \l__knot_segments_seq \knot_split_self_intersects:N
6543 \tl_set_eq:cN {knot strand #1} \l__knot_tmpa_tl
6544 }
6545 }

```

(End definition for \knot_draw_labels:n.)

\knot_split_self_intersects:N

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!

```

6546 \cs_new_protected_nopar:Npn \knot_split_self_intersects:N #1
6547 {
6548 \tl_set:Nx \l__knot_tmpc_tl {\tl_item:nn {#1} {4}}
6549 \tl_case:NnF \l__knot_tmpc_tl
6550 {
6551 \c_spath_curvetoa_tl
6552 {
6553 \fp_set:Nn \l__knot_tmpa_fp
6554 {
6555 (\tl_item:nn {#1} {3} - 3 * \tl_item:nn {#1} {6}
6556 + 3 * \tl_item:nn {#1} {9} - \tl_item:nn {#1} {12})
6557 *
6558 (3 * \tl_item:nn {#1} {8} - 3 * \tl_item:nn {#1} {11})
6559 -
6560 (\tl_item:nn {#1} {2} - 3 * \tl_item:nn {#1} {5}
6561 + 3 * \tl_item:nn {#1} {8} - \tl_item:nn {#1} {11})
6562 *
6563 (3 * \tl_item:nn {#1} {9} - 3 * \tl_item:nn {#1} {12})
6564 }
6565 \fp_set:Nn \l__knot_tmpb_fp
6566 {
6567 (\tl_item:nn {#1} {2} - 3 * \tl_item:nn {#1} {5}
6568 + 3 * \tl_item:nn {#1} {8} - \tl_item:nn {#1} {11})
6569 *
6570 (3 * \tl_item:nn {#1} {6} - 6 * \tl_item:nn {#1} {9}
6571 + 3 * \tl_item:nn {#1} {12})
6572 -
6573 (\tl_item:nn {#1} {3} - 3 * \tl_item:nn {#1} {6}
6574 + 3 * \tl_item:nn {#1} {9} - \tl_item:nn {#1} {12})
6575 *
6576 (3 * \tl_item:nn {#1} {5} - 6 * \tl_item:nn {#1} {8}
6577 + 3 * \tl_item:nn {#1} {11})
6578 }
6579 \fp_compare:nTF

```

```

6580 {
6581   \l__knot_tmpb_fp != 0
6582 }
6583 {
6584   \fp_set:Nn \l__knot_tmpa_fp {.5 * \l__knot_tmpa_fp / \l__knot_tmpb_fp}
6585   \fp_compare:nTF
6586   {
6587     0 < \l__knot_tmpa_fp && \l__knot_tmpa_fp < 1
6588   }
6589   {
6590     \spath_split_curve:NNnV
6591     \l__knot_tmpc_tl
6592     \l__knot_tmpd_tl
6593     {#1}
6594     \l__knot_tmpa_fp
6595     \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
6596     \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
6597     \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
6598     \tl_set:Nx \l__knot_tmpd_tl {\tl_tail:N \l__knot_tmpd_tl}
6599     \tl_set:Nx \l__knot_tmpd_tl {\tl_tail:N \l__knot_tmpd_tl}
6600     \tl_set:Nx \l__knot_tmpd_tl {\tl_tail:N \l__knot_tmpd_tl}
6601     \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpc_tl
6602     \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpd_tl
6603   }
6604   {
6605     \tl_set:Nn \l__knot_tmpc_tl {#1}
6606     \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
6607     \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
6608     \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
6609     \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpc_tl
6610   }
6611 }
6612 {
6613   \tl_set:Nn \l__knot_tmpc_tl {#1}
6614   \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
6615   \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
6616   \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
6617   \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpc_tl
6618 }
6619 }
6620 \c_spath_lineto_tl
6621 {
6622   \tl_set:Nn \l__knot_tmpc_tl {#1}
6623   \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
6624   \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
6625   \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
6626   \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpc_tl
6627 }
6628 }
6629 {
6630   \tl_put_right:Nn \l__knot_tmpa_tl {#1}
6631 }
6632 }

```

(End definition for \knot_split_self_intersects:N.)

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

```

6633 \cs_new_protected_nopar:Npn \knot_intersections:nn #1#2
6634 {
6635   \group_begin:
6636   \tl_set_eq:NN \l__knot_tmpa_tl \l__knot_prefix_tl
6637   \tl_put_right:Nn \l__knot_tmpa_tl {#1}
6638   \tl_set_eq:NN \l__knot_tmpb_tl \l__knot_prefix_tl
6639   \tl_put_right:Nn \l__knot_tmpb_tl {#2}
6640   \tl_set_eq:Nc \l__knot_tmpc_tl {knot \tl_use:N \l__knot_tmpa_tl}
6641   \tl_set_eq:Nc \l__knot_tmpd_tl {knot \tl_use:N \l__knot_tmpb_tl}
6642
6643   \bool_if:nTF {
6644     \l__knot_save_bool
6645     &&
6646     \tl_if_exist_p:c {
6647       knot~ intersections~
6648       \tl_use:N \l__knot_name_tl -
6649       \tl_use:N \l__knot_tmpa_tl -
6650       \tl_use:N \l__knot_tmpb_tl
6651     }
6652   }
6653   {
6654     \tl_use:c
6655     {
6656       knot~ intersections~ \tl_use:N \l__knot_name_tl -
6657       \tl_use:N \l__knot_tmpa_tl -
6658       \tl_use:N \l__knot_tmpb_tl
6659     }
6660   }
6661   {
6662     \pgfintersectionofpaths{\pgfsetpath\l__knot_tmpc_tl}{\pgfsetpath\l__knot_tmpd_tl}
6663   }
6664
6665   \int_compare:nT {\pgfintersectionsolutions > 0}
6666   {
6667     \int_step_function:nnnN
6668     {1}
6669     {1}
6670     {\pgfintersectionsolutions}
6671     \knot_do_intersection:n
6672   }
6673
6674   \knot_save_intersections:VV \l__knot_tmpa_tl \l__knot_tmpb_tl
6675   \group_end:
6676 }
6677

```

(End definition for `\knot_intersections:nn`.)

`\knot_save_intersections:nn`

```

6678 \cs_new_protected_nopar:Npn \knot_save_intersections:nn #1#2
6679 {
6680   \bool_if:NT \l__knot_save_bool
6681   {

```



```

6682 \tl_clear:N \l__knot_aux_tl
6683 \tl_put_right:Nn \l__knot_aux_tl
6684 {
6685   \def\pgfintersectionsolutions
6686 }
6687 \tl_put_right:Nx \l__knot_aux_tl
6688 {
6689   {\int_eval:n {\pgfintersectionsolutions}}
6690 }
6691 \int_compare:nT {\pgfintersectionsolutions > 0}
6692 {
6693   \int_step_inline:nnnn {1} {1} {\pgfintersectionsolutions}
6694   {
6695     \pgfpointintersectionsolution{##1}
6696     \dim_set:Nn \l__knot_tmpa_dim {\pgf@x}
6697     \dim_set:Nn \l__knot_tmpb_dim {\pgf@y}
6698     \tl_put_right:Nn \l__knot_aux_tl
6699     {
6700       \expandafter\def\csname pgfpoint@intersect@solution@##1\endcsname
6701     }
6702     \tl_put_right:Nx \l__knot_aux_tl
6703     {
6704       {
6705         \exp_not:N \pgf@x
6706         =
6707         \dim_use:N \l__knot_tmpa_dim
6708         \exp_not:N \relax
6709         \exp_not:N \pgf@y
6710         =
6711         \dim_use:N \l__knot_tmpb_dim
6712         \exp_not:N \relax
6713       }
6714     }
6715   }
6716   \tl_set:Nn \l__knot_auxa_tl {\expandafter \gdef \csname knot~ intersections~}
6717   \tl_put_right:Nx \l__knot_auxa_tl {\tl_use:N \l__knot_name_tl - #1 - #2}
6718   \tl_put_right:Nn \l__knot_auxa_tl {\endcsname}
6719   \tl_put_right:Nx \l__knot_auxa_tl {\{\tl_to_str:N \l__knot_aux_tl}\}
6720   \protected@write\@auxout{}\{\tl_to_str:N \l__knot_auxa_tl}
6721 }
6722 }
6723 }
6724 \cs_generate_variant:Nn \knot_save_intersections:nn {VV}

```

(End definition for \knot_save_intersections:nn.)

\knot_do_intersection:n This handles a specific intersection.

```

6725 \cs_new_protected_nopar:Npn \knot_do_intersection:n #1
6726 {

```

Get the intersection coordinates.

```

6727 \pgfpointintersectionsolution{#1}
6728 \dim_set:Nn \l__knot_tmpa_dim {\pgf@x}
6729 \dim_set:Nn \l__knot_tmpb_dim {\pgf@y}

```

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

```
6730 \bool_set_false:N \l__knot_skip_bool
6731 \bool_if:NT \l__knot_self_intersections_bool
6732 {
```

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

```
6733 \tl_set:Nn \l__knot_tmpc_tl {knot previous}
6734 \tl_put_right:NV \l__knot_tmpc_tl \l__knot_tmpa_tl
6735 \tl_set:Nv \l__knot_tmpc_tl \l__knot_tmpc_tl
6736 \tl_if_eq:NNT \l__knot_tmpc_tl \l__knot_tmpb_tl
6737 {
6738 \knot_test_endpoint:NVT \l__knot_tolerance_dim \l__knot_tmpb_tl {final point}
6739 {
6740 \bool_set_true:N \l__knot_skip_bool
6741 }
6742 }
6743
6744 \tl_set:Nn \l__knot_tmpc_tl {knot previous}
6745 \tl_put_right:NV \l__knot_tmpc_tl \l__knot_tmpb_tl
6746 \tl_set:Nv \l__knot_tmpc_tl \l__knot_tmpc_tl
6747 \tl_if_eq:NNT \l__knot_tmpc_tl \l__knot_tmpa_tl
6748 {
6749 \knot_test_endpoint:NVT \l__knot_tolerance_dim \l__knot_tmpa_tl {final point}
6750 {
6751 \bool_set_true:N \l__knot_skip_bool
6752 }
6753 }
6754 }
```

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

```
6755 \bool_if:NT \l__knot_ignore_ends_bool
6756 {
6757 \knot_test_endpoint:NVT \l__knot_tolerance_dim \l__knot_tmpa_tl {initial point}
6758 {
6759 \bool_set_true:N \l__knot_skip_bool
6760 }
6761 \knot_test_endpoint:NVT \l__knot_tolerance_dim \l__knot_tmpa_tl {final point}
6762 {
6763 \bool_set_true:N \l__knot_skip_bool
6764 }
6765 \knot_test_endpoint:NVT \l__knot_tolerance_dim \l__knot_tmpb_tl {initial point}
6766 {
6767 \bool_set_true:N \l__knot_skip_bool
6768 }
6769 \knot_test_endpoint:NVT \l__knot_tolerance_dim \l__knot_tmpb_tl {final point}
6770 {
6771 \bool_set_true:N \l__knot_skip_bool
6772 }
6773 }
```

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

```
6774 \bool_if:NF \l__knot_skip_bool
6775 {
```

```

6776
6777     \int_gincr:N \g__knot_intersections_int

```

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

```

6778     \bool_if:nF
6779     {
6780         \tl_if_exist_p:c {l__knot_ignore_crossing_ \int_use:N
6781             \g__knot_intersections_int}
6782         &&
6783         ! \tl_if_empty_p:c {l__knot_ignore_crossing_ \int_use:N
6784             \g__knot_intersections_int}
6785     }
6786     {

```

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

```

6787     \bool_if:nTF
6788     {
6789         \tl_if_exist_p:c {l__knot_crossing_ \int_use:N
6790             \g__knot_intersections_int}
6791         &&
6792         ! \tl_if_empty_p:c {l__knot_crossing_ \int_use:N
6793             \g__knot_intersections_int}
6794     }
6795     {
6796         \tl_set_eq:NN \l__knot_tmpg_tl \l__knot_tmpp_tl
6797     }
6798     {
6799         \tl_set_eq:NN \l__knot_tmpp_tl \l__knot_tmppa_tl
6800     }

```

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.

```

6801     \bool_if:NT \l__knot_self_intersections_bool
6802     {
6803         \knot_test_endpoint:NvNT
6804         \l__knot_redraw_tolerance_dim \l__knot_tmppg_tl {initial point}
6805         {
6806             \bool_set_true:N \l__knot_prepend_prev_bool
6807         }
6808         {
6809             \bool_set_false:N \l__knot_prepend_prev_bool
6810         }
6811         \knot_test_endpoint:NvNT
6812         \l__knot_redraw_tolerance_dim \l__knot_tmppg_tl {final point}
6813         {
6814             \bool_set_true:N \l__knot_append_next_bool
6815         }
6816         {
6817             \bool_set_false:N \l__knot_append_next_bool
6818         }

```

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

```

6819 \bool_if:nT
6820 {
6821   \l__knot_prepend_prev_bool || \l__knot_append_next_bool
6822 }
6823 {
6824   \tl_clear_new:c {knot \tl_use:N \l__knot_prefix_tl -1}
6825   \tl_set_eq:cc
6826   {knot \tl_use:N \l__knot_prefix_tl -1}
6827   {knot \tl_use:N \l__knot_tmpg_tl}
6828
6829   \tl_clear_new:c {l__knot_options_ \tl_use:N \l__knot_prefix_tl -1}
6830   \tl_set_eq:cc
6831   {l__knot_options_ \tl_use:N \l__knot_prefix_tl -1}
6832   {l__knot_options_ \tl_use:N \l__knot_tmpg_tl}
6833
6834   \bool_if:nT
6835   {
6836     \l__knot_prepend_prev_bool
6837     &&
6838     \tl_if_exist_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
6839     &&
6840     !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
6841   }
6842   {
6843     \spath_prepend_no_move:cv
6844     {knot \tl_use:N \l__knot_prefix_tl -1}
6845     {knot \tl_use:c {knot previous \tl_use:N \l__knot_tmpg_tl}}

```

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

```

6846 \bool_if:nT
6847 {
6848   \l__knot_splits_bool
6849   &&
6850   \tl_if_exist_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
6851   &&
6852   !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
6853 }
6854 {
6855   \knot_test_endpoint:NvnT
6856   \l__knot_redraw_tolerance_dim
6857   {knot previous \tl_use:N \l__knot_tmpg_tl}
6858   {initial point}
6859   {
6860     \spath_prepend_no_move:cv
6861     {knot \tl_use:N \l__knot_prefix_tl -1}
6862     {knot \tl_use:c
6863       {knot previous \tl_use:c
6864         {knot previous \tl_use:N \l__knot_tmpg_tl}
6865       }
6866     }
6867     \tl_set_eq:Nc \l__knot_tmpa_tl {knot \tl_use:N \l__knot_prefix_tl -
1}
6868   }

```

```

6869     }
6870 }

```

Now the same for appending.

```

6871     \bool_if:nT
6872     {
6873         \l__knot_append_next_bool
6874         &&
6875         \tl_if_exist_p:c {knot next \tl_use:N \l__knot_tmpg_tl}
6876         &&
6877         !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
6878     }
6879     {
6880         \spath_append_no_move:cv
6881         {knot \tl_use:N \l__knot_prefix_tl -1}
6882         {knot \tl_use:c {knot next \tl_use:N \l__knot_tmpg_tl}}
6883         \bool_if:nT
6884         {
6885             \l__knot_splits_bool
6886             &&
6887             \tl_if_exist_p:c {knot previous \tl_use:N
6888                 \l__knot_tmpg_tl}
6889             &&
6890             !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
6891         }
6892         {
6893             \knot_test_endpoint:NvnT
6894             \l__knot_redraw_tolerance_dim
6895             {knot previous \tl_use:N \l__knot_tmpg_tl}
6896             {final point}
6897             {
6898                 \spath_append_no_move:cv
6899                 {knot \tl_use:N \l__knot_prefix_tl -1}
6900                 {knot \tl_use:c
6901                     {knot next \tl_use:c
6902                         {knot next \tl_use:N \l__knot_tmpg_tl}
6903                     }
6904                 }
6905             }
6906         }
6907     }
6908     \tl_set:Nn \l__knot_tmpg_tl {\tl_use:N \l__knot_prefix_tl -1}
6909 }
6910 }

```

Now we render the crossing.

```

6911     \pgfscope
6912     \group_begin:
6913     \tikzset{
6914         knot~ diagram/every~ intersection/.try,
6915         every~ intersection/.try,
6916         knot~ diagram/intersection~ \int_use:N \g__knot_intersections_int/.try
6917     }
6918     \knot_draw_crossing:VVV \l__knot_tmpg_tl \l__knot_tmpa_dim \l__knot_tmpb_dim
6919     \coordinate

```

```

6920      (\l__knot_name_tl \c_space_tl \int_use:N \g__knot_intersections_int)
6921      at (\dim_use:N \l__knot_tmpa_dim, \dim_use:N \l__knot_tmpb_dim);
6922      \group_end:
6923      \endpgfscope

```

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

```

6924      }

```

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

```

6925      \tl_if_empty:NF \l__knot_node_tl
6926      {
6927        \seq_gpush:Nx
6928        \g__knot_nodes_seq
6929        {
6930          \l__knot_node_tl
6931          at
6932          (\dim_use:N \l__knot_tmpa_dim, \dim_use:N \l__knot_tmpb_dim) {};
6933        }
6934      }
6935    }
6936  }
6937
6938  \cs_generate_variant:Nn \knot_intersections:nn {VV}

```

(End definition for \knot_do_intersection:n.)

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

```

6939  \prg_new_conditional:Npnn \knot_test_endpoint:NN #1#2 {p,T,F,TF}
6940  {
6941    \dim_compare:nTF
6942    {
6943      \dim_abs:n { \l__knot_tmpa_dim - \tl_item:Nn #2 {1}}
6944      +
6945      \dim_abs:n { \l__knot_tmpb_dim - \tl_item:Nn #2 {2}}
6946      <
6947      #1
6948    }
6949    {
6950      \prg_return_true:
6951    }
6952    {
6953      \prg_return_false:
6954    }
6955  }

```

(End definition for \knot_test_endpoint:N.)

`\knot_test_endpoint:nn` Wrapper around the above.

```

6956  \prg_new_protected_conditional:Npnn \knot_test_endpoint:Nnn #1#2#3 {T,F,TF}
6957  {
6958    \use:c {spath_#3:Nv} \l__knot_tmpd_tl {knot #2}
6959    \knot_test_endpoint:NNTF #1 \l__knot_tmpd_tl
6960    {
6961      \prg_return_true:
6962    }

```

```

6963 {
6964   \prg_return_false:
6965 }
6966 }
6967
6968 \cs_generate_variant:Nn \knot_test_endpoint:NnnT {NVnT,NvnT}
6969 \cs_generate_variant:Nn \knot_test_endpoint:NnnF {NVnF,NvnF}
6970 \cs_generate_variant:Nn \knot_test_endpoint:NnnTF {NVnTF,NvnTF}

```

(End definition for \knot_test_endpoint:nn.)

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

```

6971 \cs_new_protected_nopar:Npn \knot_draw_crossing:nnn #1#2#3
6972 {
6973   \group_begin:
6974   \pgfscope
6975   \path[knot~ diagram/background~ clip] (#2, #3)
6976   circle[radius=\l__knot_clip_bg_radius_dim];
6977
6978   \tl_set:Nn \l__knot_tmpa_tl {knot~ diagram/every~ strand/.try,}
6979   \tl_if_exist:cT {l__knot_options_ #1}
6980   {
6981     \tl_put_right:Nv \l__knot_tmpa_tl {l__knot_options_ #1}
6982   }
6983   \tl_put_right:Nn \l__knot_tmpa_tl
6984   {
6985     ,knotbg
6986     ,line~ width= \tl_use:N \l__knot_clip_width_tl * \pgflinewidth
6987   }
6988   \spath_tikz_path:Vv \l__knot_tmpa_tl {knot #1}
6989
6990   \endpgfscope
6991
6992   \pgfscope
6993   \path[knot~ diagram/clip] (#2, #3)
6994   circle[radius=\l__knot_clip_draw_radius_dim];
6995
6996   \tl_set:Nn \l__knot_tmpa_tl {knot~ diagram/every~ strand/.try,}
6997   \tl_if_exist:cT {l__knot_options_ #1}
6998   {
6999     \tl_put_right:Nv \l__knot_tmpa_tl {l__knot_options_ #1}
7000   }
7001   \tl_put_right:Nn \l__knot_tmpa_tl
7002   {
7003     ,knot~ diagram/only~ when~ rendering/.try
7004     ,only~ when~ rendering/.try
7005   }
7006   \spath_tikz_path:Vv \l__knot_tmpa_tl {knot #1}
7007
7008   \endpgfscope
7009   \group_end:
7010 }
7011
7012 \cs_generate_variant:Nn \knot_draw_crossing:nnn {nVV, VVV}

```

```

7013
7014 \cs_new_protected_nopar:Npn \knot_draw_crossing:nn #1#2
7015 {
7016   \tikz@scan@one@point\pgfutil@firstofone #2 \relax
7017   \knot_draw_crossing:nVV {#1} \pgf@x \pgf@y
7018 }

```

(End definition for \knot_draw_crossing:nnn.)

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

```

7019 \cs_new_protected_nopar:Npn \knot_split_strands:
7020 {
7021   \int_gzero:N \g__knot_filaments_int
7022   \int_step_function:nnnN {1} {1} {\l__knot_strands_int} \knot_split_strand:n
7023   \int_step_function:nnnN {1} {1} {\g__knot_filaments_int} \knot_compute_nexts:n
7024 }

```

(End definition for \knot_split_strands:.)

`\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.

```

7025 \cs_new_protected_nopar:Npn \knot_compute_nexts:n #1
7026 {
7027   \tl_clear_new:c {knot next \tl_use:c {knot previous filament #1}}
7028   \tl_set:cn {knot next \tl_use:c {knot previous filament #1}} {filament #1}
7029 }

```

(End definition for \knot_compute_nexts:n.)

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

```

7030 \cs_new_protected_nopar:Npn \knot_split_strand:n #1
7031 {
7032   \int_set_eq:NN \l__knot_component_start_int \g__knot_filaments_int
7033   \int_incr:N \l__knot_component_start_int
7034   \tl_set_eq:Nc \l__knot_tmpa_tl {l__knot_options_strand #1}
7035   \spath_segments_to_seq:Nv \l__knot_segments_seq {knot strand #1}
7036   \seq_map_function:NN \l__knot_segments_seq \knot_save_filament:N
7037 }

```

(End definition for \knot_split_strand:n.)

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

```

7038 \cs_new_protected_nopar:Npn \knot_save_filament:N #1
7039 {
7040   \tl_set:Nx \l__knot_tmpb_tl {\tl_item:nn {#1} {4}}
7041   \tl_case:NnF \l__knot_tmpb_tl
7042   {
7043     \c_spath_moveto_tl
7044     {
7045       \int_compare:nT {\l__knot_component_start_int < \g__knot_filaments_int}
7046       {
7047         \int_set_eq:NN \l__knot_component_start_int \g__knot_filaments_int
7048       }
7049     }
7050     \c_spath_lineto_tl

```



```

7051 {
7052   \int_gincr:N \g__knot_filaments_int
7053   \tl_clear_new:c {knot filament \int_use:N \g__knot_filaments_int}
7054   \tl_set:cn {knot filament \int_use:N \g__knot_filaments_int} {#1}
7055
7056   \tl_clear_new:c {l__knot_options_filament \int_use:N \g__knot_filaments_int}
7057   \tl_set_eq:cN {l__knot_options_filament \int_use:N \g__knot_filaments_int}
7058   \l__knot_tmpa_tl
7059
7060   \tl_clear_new:c {knot previous filament \int_use:N \g__knot_filaments_int}
7061   \int_compare:nF {\l__knot_component_start_int == \g__knot_filaments_int}
7062   {
7063     \tl_set:cx {knot previous filament \int_use:N \g__knot_filaments_int}
7064     {filament \int_eval:n {\g__knot_filaments_int - 1}}
7065   }
7066 }
7067 \c_spath_curvetoa_tl
7068 {
7069   \int_gincr:N \g__knot_filaments_int
7070   \tl_clear_new:c {knot filament \int_use:N \g__knot_filaments_int}
7071   \tl_set:cn {knot filament \int_use:N \g__knot_filaments_int} {#1}
7072   \tl_clear_new:c {l__knot_options_filament \int_use:N \g__knot_filaments_int}
7073   \tl_set_eq:cN {l__knot_options_filament \int_use:N \g__knot_filaments_int}
7074   \l__knot_tmpa_tl
7075
7076   \tl_clear_new:c {knot previous filament \int_use:N \g__knot_filaments_int}
7077   \int_compare:nF {\l__knot_component_start_int == \g__knot_filaments_int}
7078   {
7079     \tl_set:cx
7080     {knot previous filament \int_use:N \g__knot_filaments_int}
7081     {filament \int_eval:n {\g__knot_filaments_int - 1}}
7082   }
7083 }
7084 \c_spath_closepath_tl
7085 {
7086   \int_gincr:N \g__knot_filaments_int
7087   \tl_clear_new:c {knot filament \int_use:N \g__knot_filaments_int}
7088   \tl_clear:N \l__knot_tmpa_tl
7089   \tl_put_right:Nx
7090   {
7091     \tl_item:nn {#1} {1}\tl_item:nn {#1} {2}\tl_item:nn {#1} {3}
7092   }
7093   \tl_put_right:NV \l__knot_tmpa_tl \c_spath_lineto_tl
7094   \tl_put_right:Nx {\tl_item:nn {#1} {5}\tl_item:nn {#1} {6}}
7095
7096   \tl_set:cV {knot filament \int_use:N \g__knot_filaments_int} \l__knot_tmpa_tl
7097   \tl_set_eq:cN {l__knot_options_filament \int_use:N \g__knot_filaments_int}
7098   \l__knot_tmpa_tl
7099   \tl_clear_new:c {knot previous filament \int_use:N \g__knot_filaments_int}
7100   \int_compare:nF {\l__knot_component_start_int == \g__knot_filaments_int}
7101   {
7102     \tl_set:cx
7103     {knot previous filament \int_use:N \g__knot_filaments_int}
7104     {filament \int_eval:n {\g__knot_filaments_int - 1}}

```

```

7105     }
7106     \tl_set:cx
7107     {knot previous filament \int_use:N \l__knot_component_start_int}
7108     {filament \int_use:N \g__knot_filaments_int}
7109   }
7110 }
7111 {
7112 }
7113 }

```

(End definition for \knot_save_filament:N.)

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

```

7114 \NewDocumentCommand \redraw { m m }
7115 {
7116   % \tikz@scan@one@point\pgfutil@firstofone #2 \relax
7117   \tl_put_right:Nn \l__knot_redraws_tl {\knot_draw_crossing:nn}
7118   \tl_put_right:Nx \l__knot_redraws_tl {
7119     {strand #1} {#2}% {\dim_use:N \pgf@x} {\dim_use:N \pgf@y}
7120   }
7121 }

```

(End definition for \redraw.)

```

7122 \ExplSyntaxOff

```

<@@=>

\pgf@sh__knotknotanchor Add the extra anchors for the knot crossing nodes.

```

7123 \def\pgf@sh__knotknotanchor#1#2{%
7124   \anchor{#2 north west}{%
7125     \csname pgf@anchor@knot #1@north west\endcsname%
7126     \pgf@x=#2\pgf@x%
7127     \pgf@y=#2\pgf@y%
7128   }%
7129   \anchor{#2 north east}{%
7130     \csname pgf@anchor@knot #1@north east\endcsname%
7131     \pgf@x=#2\pgf@x%
7132     \pgf@y=#2\pgf@y%
7133   }%
7134   \anchor{#2 south west}{%
7135     \csname pgf@anchor@knot #1@south west\endcsname%
7136     \pgf@x=#2\pgf@x%
7137     \pgf@y=#2\pgf@y%
7138   }%
7139   \anchor{#2 south east}{%
7140     \csname pgf@anchor@knot #1@south east\endcsname%
7141     \pgf@x=#2\pgf@x%
7142     \pgf@y=#2\pgf@y%
7143   }%
7144   \anchor{#2 north}{%
7145     \csname pgf@anchor@knot #1@north\endcsname%
7146     \pgf@x=#2\pgf@x%
7147     \pgf@y=#2\pgf@y%
7148   }%
7149   \anchor{#2 east}{%

```

```

7150     \csname pgf@anchor@knot #1@east\endcsname%
7151     \pgf@x=#2\pgf@x%
7152     \pgf@y=#2\pgf@y%
7153 }%
7154 \anchor{#2 west}{%
7155     \csname pgf@anchor@knot #1@west\endcsname%
7156     \pgf@x=#2\pgf@x%
7157     \pgf@y=#2\pgf@y%
7158 }%
7159 \anchor{#2 south}{%
7160     \csname pgf@anchor@knot #1@south\endcsname%
7161     \pgf@x=#2\pgf@x%
7162     \pgf@y=#2\pgf@y%
7163 }%
7164 }

```

(End definition for \pgf@sh__knotknotanchor.)

knot_crossing

```

7165 \pgfdeclareshape{knot crossing}
7166 {
7167     \inheritshapedanchors[from=circle] % this is nearly a circle
7168     \inheritanchorborder[from=circle]
7169     \inheritanchor[from=circle]{north}
7170     \inheritanchor[from=circle]{north west}
7171     \inheritanchor[from=circle]{north east}
7172     \inheritanchor[from=circle]{center}
7173     \inheritanchor[from=circle]{west}
7174     \inheritanchor[from=circle]{east}
7175     \inheritanchor[from=circle]{mid}
7176     \inheritanchor[from=circle]{mid west}
7177     \inheritanchor[from=circle]{mid east}
7178     \inheritanchor[from=circle]{base}
7179     \inheritanchor[from=circle]{base west}
7180     \inheritanchor[from=circle]{base east}
7181     \inheritanchor[from=circle]{south}
7182     \inheritanchor[from=circle]{south west}
7183     \inheritanchor[from=circle]{south east}
7184     \inheritanchorborder[from=circle]
7185     \pgf@sh__knotknotanchor{crossing}{2}
7186     \pgf@sh__knotknotanchor{crossing}{3}
7187     \pgf@sh__knotknotanchor{crossing}{4}
7188     \pgf@sh__knotknotanchor{crossing}{8}
7189     \pgf@sh__knotknotanchor{crossing}{16}
7190     \pgf@sh__knotknotanchor{crossing}{32}
7191     \backgroundpath{
7192         \pgfutil@tempdima=\radius%
7193         \pgfmathsetlength{\pgf@xb}{\pgfkeysvalueof{/pgf/outer xsep}}%
7194         \pgfmathsetlength{\pgf@yb}{\pgfkeysvalueof{/pgf/outer ysep}}%
7195         \ifdim\pgf@xb<\pgf@yb%
7196             \advance\pgfutil@tempdima by-\pgf@yb%
7197         \else%
7198             \advance\pgfutil@tempdima by-\pgf@xb%
7199         \fi%

```

```

7200 }
7201 }

```

(End definition for knot crossing.)

knot_over_cross

```

7202 \pgfdeclareshape{knot over cross}
7203 {
7204   \inheritsavedanchors[from=rectangle] % this is nearly a circle
7205   \inheritanchorborder[from=rectangle]
7206   \inheritanchor[from=rectangle]{north}
7207   \inheritanchor[from=rectangle]{north west}
7208   \inheritanchor[from=rectangle]{north east}
7209   \inheritanchor[from=rectangle]{center}
7210   \inheritanchor[from=rectangle]{west}
7211   \inheritanchor[from=rectangle]{east}
7212   \inheritanchor[from=rectangle]{mid}
7213   \inheritanchor[from=rectangle]{mid west}
7214   \inheritanchor[from=rectangle]{mid east}
7215   \inheritanchor[from=rectangle]{base}
7216   \inheritanchor[from=rectangle]{base west}
7217   \inheritanchor[from=rectangle]{base east}
7218   \inheritanchor[from=rectangle]{south}
7219   \inheritanchor[from=rectangle]{south west}
7220   \inheritanchor[from=rectangle]{south east}
7221   \inheritanchorborder[from=rectangle]
7222   \backgroundpath{
7223     \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
7224     \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
7225     \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
7226     \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
7227   }
7228   \foregroundpath{
7229     % store lower right in xa/ya and upper right in xb/yb
7230     \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
7231     \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
7232     \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
7233     \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
7234   }
7235 }

```

(End definition for knot over cross.)

knot_under_cross

```

7236 \pgfdeclareshape{knot under cross}
7237 {
7238   \inheritsavedanchors[from=rectangle] % this is nearly a circle
7239   \inheritanchorborder[from=rectangle]
7240   \inheritanchor[from=rectangle]{north}
7241   \inheritanchor[from=rectangle]{north west}
7242   \inheritanchor[from=rectangle]{north east}
7243   \inheritanchor[from=rectangle]{center}
7244   \inheritanchor[from=rectangle]{west}
7245   \inheritanchor[from=rectangle]{east}
7246   \inheritanchor[from=rectangle]{mid}

```

```

7247 \inheritanchor[from=rectangle]{mid west}
7248 \inheritanchor[from=rectangle]{mid east}
7249 \inheritanchor[from=rectangle]{base}
7250 \inheritanchor[from=rectangle]{base west}
7251 \inheritanchor[from=rectangle]{base east}
7252 \inheritanchor[from=rectangle]{south}
7253 \inheritanchor[from=rectangle]{south west}
7254 \inheritanchor[from=rectangle]{south east}
7255 \inheritanchorborder[from=rectangle]
7256 \backgroundpath{
7257   \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
7258   \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
7259   \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
7260   \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
7261 }
7262 \foregroundpath{
7263 % store lower right in xa/ya and upper right in xb/yb
7264   \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
7265   \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
7266   \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
7267   \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
7268 }
7269 }

```

(End definition for knot under cross.)

knot_vert

```

7270 \pgfdeclareshape{knot vert}
7271 {
7272   \inheritssavedanchors[from=rectangle] % this is nearly a circle
7273   \inheritanchorborder[from=rectangle]
7274   \inheritanchor[from=rectangle]{north}
7275   \inheritanchor[from=rectangle]{north west}
7276   \inheritanchor[from=rectangle]{north east}
7277   \inheritanchor[from=rectangle]{center}
7278   \inheritanchor[from=rectangle]{west}
7279   \inheritanchor[from=rectangle]{east}
7280   \inheritanchor[from=rectangle]{mid}
7281   \inheritanchor[from=rectangle]{mid west}
7282   \inheritanchor[from=rectangle]{mid east}
7283   \inheritanchor[from=rectangle]{base}
7284   \inheritanchor[from=rectangle]{base west}
7285   \inheritanchor[from=rectangle]{base east}
7286   \inheritanchor[from=rectangle]{south}
7287   \inheritanchor[from=rectangle]{south west}
7288   \inheritanchor[from=rectangle]{south east}
7289   \inheritanchorborder[from=rectangle]
7290   \backgroundpath{
7291 % store lower right in xa/ya and upper right in xb/yb
7292   \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
7293   \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
7294   \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
7295   \pgfpathlineto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
7296   \pgfpathmoveto{\pgfqpoint{\pgf@xb}{\pgf@yb}}

```

```

7297 \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
7298 }
7299 }

```

(End definition for knot vert.)

knot_horiz

```

7300 \pgfdeclareshape{knot horiz}
7301 {
7302 \inheritsavedanchors[from=rectangle] % this is nearly a circle
7303 \inheritanchorborder[from=rectangle]
7304 \inheritanchor[from=rectangle]{north}
7305 \inheritanchor[from=rectangle]{north west}
7306 \inheritanchor[from=rectangle]{north east}
7307 \inheritanchor[from=rectangle]{center}
7308 \inheritanchor[from=rectangle]{west}
7309 \inheritanchor[from=rectangle]{east}
7310 \inheritanchor[from=rectangle]{mid}
7311 \inheritanchor[from=rectangle]{mid west}
7312 \inheritanchor[from=rectangle]{mid east}
7313 \inheritanchor[from=rectangle]{base}
7314 \inheritanchor[from=rectangle]{base west}
7315 \inheritanchor[from=rectangle]{base east}
7316 \inheritanchor[from=rectangle]{south}
7317 \inheritanchor[from=rectangle]{south west}
7318 \inheritanchor[from=rectangle]{south east}
7319 \inheritanchorborder[from=rectangle]
7320 \foregroundpath{
7321 % store lower right in xa/ya and upper right in xb/yb
7322 \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
7323 \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
7324 \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
7325 \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
7326 \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
7327 \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
7328 }
7329 }

```

(End definition for knot horiz.)