

The `spath3` package: code

Andrew Stacey

loopspace@mathforge.org

v2.4 from 2021/02/21

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} {2021/02/21} {2.4} {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 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_tmpc_tl
37 \tl_new:N \l__spath_tmpd_tl
38 \tl_new:N \l__spath_tmpe_tl
39 \tl_new:N \l__spath_tmpf_tl
40 \tl_new:N \l__spath_tmpg_tl
41 \tl_new:N \l__spath_tmph_tl
42 \tl_new:N \l__spath_tmphi_tl
43
44 \seq_new:N \l__spath_tmpa_seq
45 \seq_new:N \l__spath_tmpb_seq
46 \seq_new:N \l__spath_tmpc_seq
47
48 \dim_new:N \l__spath_tmpa_dim
49 \dim_new:N \l__spath_tmpb_dim
50
51 \fp_new:N \l__spath_tmpa_fp
52 \fp_new:N \l__spath_tmpb_fp
53 \fp_new:N \l__spath_tmpc_fp
54
55 \int_new:N \l__spath_tmpa_int
56 \int_new:N \l__spath_tmpb_int
57
58 \bool_new:N \l__spath_tmpa_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_tmpe_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:mn {
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` `\spath_greallength: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).

```

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.

`\spath_gnumberofcomponents:Nn`

```

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` The starting point of the path.
`\spath_ginitialpoint:Nn`

```

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` The final point of the path.
`\spath_gfinalpoint:Nn`

```

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`
`\spath_gfinalmovepoint:Nn`

Get the last move on the path.

```

409 \cs_new_protected_nopar:Npn \__spath_finalmovepoint:n #1
410 {
411   \group_begin:
412   \tl_set:Nn \l__spath_tmpc_tl { {Opt} {Opt} }
413   \tl_set:Nn \l__spath_tmpa_tl {#1}
414   \bool_do_until:nm
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` This computes the reverse of the path.

```

\spath_greverse:Nn
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:nm {
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_tmpe_tl \c_spath_closepath_tl
529   {
530     \bool_set_true:N \l__spath_closed_bool
531   }
532   {
533     \msg_warning:nnx
534     { spath3 }
535     { unknown path construction }
536     {\l__spath_tmpe_tl }
537   }
538
539   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
540   \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl}
541   \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_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_tmpb_tl \l__spath_tmpd_tl
560
561 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_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_tmpa_tl
592   \int_compare:nT
593   {
594     \tl_count:n {#1} > 3
595   }
596   {
597     \tl_set:Nx \l__spath_tmpa_tl
598     {
599       \tl_item:Nn {#1} {4}
600     }
601   }
602   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_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.

<pre> \spath_translate:Nnnn \spath_translate:Nnn \spath_gtranslate:Nnnn \spath_gtranslate:Nnn </pre>	<p>Translates a path by an amount.</p> <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.)

```

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

```

```

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

```

\spath_transform:Nnnnnnnn
\spath_gtransform:Nnnnnnnn
\spath_gtransform:Nnnnnnnn
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, Nnxxxxxx, 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, Nnxxxxxx, 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 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 1026 \cs_new_protected_nopar:Npn \__spath_span:nnn #1#2#3
\spath_normalise:N 1027 {
\spath_gnormalise:Nn 1028   \group_begin:
\spath_gnormalise:N 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` 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.

`\spath_splice_between:Nnn`

`\spath_gsplice_between:Nnnn`

`\spath_gsplice_between:Nnn`

```

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_tmpb_tl {#2} \l__spath_tmpd_tl \l__spath_tmpe_tl
1200   \spath_append_no_move:NnV \l__spath_tmpe_tl {#1} \l__spath_tmpb_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, 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, Nvn, NVn}

```

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

`\spath_close_with:Nn` Closes the first path by splicing in the second.

`\spath_gclose_with:Nn`

```

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_tmpb_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_tmpb_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, cV}
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, cV}

```

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

```

\spath_append:Nnn Prepend the path from the second spath to the first.
\spath_append:Nn 1358 \cs_new_protected_nopar:Npn \spath_append:Nnn #1#2#3
\spath_gappend:Nnn 1359 {
\spath_gappend:Nn 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, 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, cV}

```

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

```

\spath_prepend_no_move:Nnn Prepend the path from the second spath to the first, removing the adjoining move.
\spath_prepend_no_move:Nn 1380 \cs_new_protected_nopar:Npn \spath_prepend_no_move:Nnn #1#2#3
\spath_gprepend_no_move:Nnn 1381 {
\spath_gprepend_no_move:Nn 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 {

```

```

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_gcclose:Nn
\spath_gcclose: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_gcclose: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_gcclose: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_replace_lines:Nn Replace any line segments by Bézier curves.
\spath_replace_lines:Nn 1586 \cs_new_protected_nopar:Npn \__spath_replace_lines:n #1
\spath_replace_lines:Nn 1587 {
\spath_replace_lines:Nn 1588 \group_begin:
1589 \tl_set:Nn \l__spath_tmpa_tl {#1}
1590 \tl_clear:N \l__spath_tmpb_tl
1591 \dim_set:Nn \l__spath_tmpa_dim {0pt}
1592 \dim_set:Nn \l__spath_tmpb_dim {0pt}
1593
1594 \bool_do_until:n
1595 {
1596 \tl_if_empty_p:N \l__spath_tmpa_tl
1597 }
1598 {
1599 \tl_set:Nx \l__spath_tmpe_tl {\tl_item:Nn \l__spath_tmpa_tl {1}}
1600 \tl_set:Nx \l__spath_tmpe_tl {\tl_item:Nn \l__spath_tmpa_tl {2}}
1601 \tl_set:Nx \l__spath_tmpe_tl {\tl_item:Nn \l__spath_tmpa_tl {3}}
1602
1603 \tl_if_eq:NNTF \l__spath_tmpe_tl \c_spath_lineto_tl
1604 {
1605 \tl_put_right:NV \l__spath_tmpb_tl \c_spath_curvetoa_tl
1606 \tl_put_right:Nx \l__spath_tmpb_tl
1607 {
1608 {
1609 \fp_to_dim:n
1610 {
1611 2/3 * (\l__spath_tmpa_dim)
1612 +
1613 1/3 * (\l__spath_tmpe_tl)
1614 }
1615 }
1616 }
1617 \tl_put_right:Nx \l__spath_tmpb_tl
1618 {
1619 {
1620 \fp_to_dim:n
1621 {
1622 2/3 * (\l__spath_tmpe_dim)
1623 +
1624 1/3 * (\l__spath_tmpe_tl)
1625 }
1626 }
1627 }
1628 \tl_put_right:NV \l__spath_tmpb_tl \c_spath_curvetob_tl
1629 \tl_put_right:Nx \l__spath_tmpb_tl
1630 {
1631 {
1632 \fp_to_dim:n
1633 {
1634 1/3 * (\l__spath_tmpa_dim)
```

```

1635         +
1636         2/3 * (\l__spath_tmpd_tl)
1637     }
1638 }
1639 }
1640 \tl_put_right:Nx \l__spath_tmpb_tl
1641 {
1642     {
1643         \fp_to_dim:n
1644         {
1645             1/3 * (\l__spath_tmpb_dim)
1646             +
1647             2/3 * (\l__spath_tmpe_tl)
1648         }
1649     }
1650 }
1651 \tl_put_right:NV \l__spath_tmpb_tl \c_spath_curveto_tl
1652 \__spath_tl_put_right_braced:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
1653 \__spath_tl_put_right_braced:NV \l__spath_tmpb_tl \l__spath_tmpe_tl
1654 }
1655 {
1656     \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpe_tl
1657     \__spath_tl_put_right_braced:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
1658     \__spath_tl_put_right_braced:NV \l__spath_tmpb_tl \l__spath_tmpe_tl
1659 }
1660
1661 \dim_set:Nn \l__spath_tmpa_dim {\l__spath_tmpd_tl}
1662 \dim_set:Nn \l__spath_tmpb_dim {\l__spath_tmpe_tl}
1663
1664 \prg_replicate:nn {3}
1665 {
1666     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1667 }
1668 }
1669 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
1670 \group_end:
1671 }
1672 \cs_generate_variant:Nn \__spath_replace_lines:n {V}
1673 \cs_new_protected_nopar:Npn \spath_replace_lines:Nn #1#2
1674 {
1675     \__spath_replace_lines:n {#2}
1676     \tl_set_eq:NN #1 \g__spath_output_tl
1677     \tl_gclear:N \g__spath_output_tl
1678 }
1679 \cs_generate_variant:Nn \spath_replace_lines:Nn {NV, cV, cv, Nv}
1680 \cs_new_protected_nopar:Npn \spath_replace_lines:N #1
1681 {
1682     \spath_replace_lines:NV #1#1
1683 }
1684 \cs_generate_variant:Nn \spath_replace_lines:N {c}
1685 \cs_new_protected_nopar:Npn \spath_greplace_lines:Nn #1#2
1686 {
1687     \__spath_replace_lines:n {#2}
1688     \tl_gset_eq:NN #1 \g__spath_output_tl

```

```

1689 \tl_gclear:N \g__spath_output_tl
1690 }
1691 \cs_generate_variant:Nn \spath_greplace_lines:Nn {NV, cV, cv, Nv}
1692 \cs_new_protected_nopar:Npn \spath_greplace_lines:N #1
1693 {
1694   \spath_greplace_lines:NV #1#1
1695 }
1696 \cs_generate_variant:Nn \spath_greplace_lines:N {c}

```

(End definition for `\spath_replace_lines:Nn`.)

```

\spath_remove_empty_components:Nn Remove any component that is simply a moveto.
\spath_remove_empty_components:N
\spath_gremove_empty_components:Nn
\spath_gremove_empty_components:N
1697 \cs_new_protected_nopar:Npn \__spath_remove_empty_components:n #1
1698 {
1699   \group_begin:
1700   \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
1701   \tl_clear:N \l__spath_tmpa_tl
1702   \seq_map_inline:Nn \l__spath_tmpa_seq
1703   {
1704     \int_compare:nF
1705     {
1706       \tl_count:n {##1} == 3
1707     }
1708     {
1709       \tl_put_right:Nn \l__spath_tmpa_tl {##1}
1710     }
1711   }
1712   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
1713   \group_end:
1714 }
1715 \cs_new_protected_nopar:Npn \spath_remove_empty_components:Nn #1#2
1716 {
1717   \__spath_remove_empty_components:n {#2}
1718   \tl_set_eq:NN #1 \g__spath_output_tl
1719   \tl_gclear:N \g__spath_output_tl
1720 }
1721 \cs_generate_variant:Nn \spath_remove_empty_components:Nn {NV}
1722 \cs_new_protected_nopar:Npn \spath_remove_empty_components:N #1
1723 {
1724   \spath_remove_empty_components:NV #1#1
1725 }
1726 \cs_generate_variant:Nn \spath_remove_empty_components:N {c}
1727 \cs_new_protected_nopar:Npn \spath_gremove_empty_components:Nn #1#2
1728 {
1729   \__spath_remove_empty_components:n {#2}
1730   \tl_gset_eq:NN #1 \g__spath_output_tl
1731   \tl_gclear:N \g__spath_output_tl
1732 }
1733 \cs_generate_variant:Nn \spath_gremove_empty_components:Nn {NV}
1734 \cs_new_protected_nopar:Npn \spath_gremove_empty_components:N #1
1735 {
1736   \spath_gremove_empty_components:NV #1#1
1737 }
1738 \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.

```

1739 \prg_new_protected_conditional:Npnn \spath_if_eq:nn #1#2 { T, F, TF }
1740 {
1741   \group_begin:
1742   \tl_set:Nn \l__spath_tmpa_tl {#1}
1743   \tl_set:Nn \l__spath_tmpb_tl {#2}
1744   \bool_gset_true:N \g__spath_tmpa_bool
1745   \int_compare:nNnTF
1746   {\tl_count:N \l__spath_tmpa_tl}
1747   =
1748   {\tl_count:N \l__spath_tmpb_tl}
1749   {
1750     \int_step_inline:nnnn {1} {3} {\tl_count:N \l__spath_tmpa_tl}
1751     {
1752       \tl_set:Nx \l__spath_tmpc_tl {\tl_item:Nn \l__spath_tmpa_tl {##1}}
1753       \tl_set:Nx \l__spath_tmpd_tl {\tl_item:Nn \l__spath_tmpb_tl {##1}}
1754       \tl_if_eq:NNF \l__spath_tmpc_tl \l__spath_tmpd_tl
1755       {
1756         \bool_gset_false:N \g__spath_tmpa_bool
1757       }
1758       \dim_set:Nn \l__spath_tmpa_dim {\tl_item:Nn \l__spath_tmpa_tl {##1+1}}
1759       \dim_set:Nn \l__spath_tmpb_dim {\tl_item:Nn \l__spath_tmpb_tl {##1+1}}
1760       \dim_compare:nF
1761       {
1762         \dim_abs:n
1763         {
1764           \l__spath_tmpa_dim - \l__spath_tmpb_dim
1765         }
1766         < 0.001pt
1767       }
1768       {
1769         \bool_gset_false:N \g__spath_tmpa_bool
1770       }
1771       \dim_set:Nn \l__spath_tmpa_dim {\tl_item:Nn \l__spath_tmpa_tl {##1+2}}
1772       \dim_set:Nn \l__spath_tmpb_dim {\tl_item:Nn \l__spath_tmpb_tl {##1+2}}
1773       \dim_compare:nF
1774       {
1775         \dim_abs:n
1776         {
1777           \l__spath_tmpa_dim - \l__spath_tmpb_dim
1778         }
1779         < 0.001pt
1780       }
1781       {
1782         \bool_gset_false:N \g__spath_tmpa_bool
1783       }
1784     }
1785   }
1786   {
1787     \bool_gset_false:N \g__spath_tmpa_bool
1788   }
1789   \group_end:

```

```

1790 \bool_if:NTF \g__spath_tpa_bool
1791 {
1792   \prg_return_true:
1793 }
1794 {
1795   \prg_return_false:
1796 }
1797 }
1798 \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 1799 \cs_new_protected_nopar:Npn \__spath_split_curve:nn #1#2
1800 {
1801   \group_begin:
1802   \tl_set_eq:NN \l__spath_tpa_tl \c_spath_moveto_tl
1803   \tl_put_right:Nx \l__spath_tpa_tl {
1804     {\tl_item:nn {#1} {2}}
1805     {\tl_item:nn {#1} {3}}
1806   }
1807   \tl_put_right:NV \l__spath_tpa_tl \c_spath_curvetoa_tl
1808   \tl_put_right:Nx \l__spath_tpa_tl
1809   {
1810     {\fp_to_dim:n
1811       {
1812         (1 - #2) * \tl_item:nn {#1} {2} + (#2) * \tl_item:nn {#1} {5}
1813       }}
1814     {\fp_to_dim:n
1815       {
1816         (1 - #2) * \tl_item:nn {#1} {3} + (#2) * \tl_item:nn {#1} {6}
1817       }}
1818   }
1819
1820   \tl_put_right:NV \l__spath_tpa_tl \c_spath_curvetob_tl
1821   \tl_put_right:Nx \l__spath_tpa_tl
1822   {
1823     {\fp_to_dim:n
1824       {
1825         (1 - #2)^2 * \tl_item:nn {#1} {2}
1826         + 2 * (1 - #2) * (#2) * \tl_item:nn {#1} {5}
1827         + (#2)^2 * \tl_item:nn {#1} {8}
1828       }}
1829     {\fp_to_dim:n
1830       {
1831         (1 - #2)^2 * \tl_item:nn {#1} {3}
1832         + 2 * (1 - #2) * (#2) * \tl_item:nn {#1} {6}
1833         + (#2)^2 * \tl_item:nn {#1} {9}
1834       }}
1835   }
1836
1837   \tl_put_right:NV \l__spath_tpa_tl \c_spath_curveto_tl

```

```

1838 \tl_put_right:Nx \l__spath_tmpa_tl
1839 {
1840   {\fp_to_dim:n
1841     {
1842       (1 - #2)^3 * \tl_item:nn {#1} {2}
1843       + 3 * (1 - #2)^2 * (#2) * \tl_item:nn {#1} {5}
1844       + 3 * (1 - #2) * (#2)^2 * \tl_item:nn {#1} {8}
1845       + (#2)^3 * \tl_item:nn {#1} {11}
1846     }}
1847   {\fp_to_dim:n
1848     {
1849       (1 - #2)^3 * \tl_item:nn {#1} {3}
1850       + 3 * (1 - #2)^2 * (#2) * \tl_item:nn {#1} {6}
1851       + 3 * (1 - #2) * (#2)^2 * \tl_item:nn {#1} {9}
1852       + (#2)^3 * \tl_item:nn {#1} {12}
1853     }}
1854 }
1855
1856 \tl_gclear:N \g__spath_output_tl
1857 \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpa_tl
1858
1859 \tl_clear:N \l__spath_tmpa_tl
1860 \tl_set_eq:NN \l__spath_tmpa_tl \c_spath_moveto_tl
1861 \tl_put_right:Nx \l__spath_tmpa_tl
1862 {
1863   {\fp_to_dim:n
1864     {
1865       (1 - #2)^3 * \tl_item:nn {#1} {2}
1866       + 3 * (1 - #2)^2 * (#2) * \tl_item:nn {#1} {5}
1867       + 3 * (1 - #2) * (#2)^2 * \tl_item:nn {#1} {8}
1868       + (#2)^3 * \tl_item:nn {#1} {11}
1869     }}
1870   {\fp_to_dim:n
1871     {
1872       (1 - #2)^3 * \tl_item:nn {#1} {3}
1873       + 3 * (1 - #2)^2 * (#2) * \tl_item:nn {#1} {6}
1874       + 3 * (1 - #2) * (#2)^2 * \tl_item:nn {#1} {9}
1875       + (#2)^3 * \tl_item:nn {#1} {12}
1876     }}
1877 }
1878
1879 \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curvetoa_tl
1880 \tl_put_right:Nx \l__spath_tmpa_tl
1881 {
1882   {\fp_to_dim:n
1883     {
1884       (1 - #2)^2 * \tl_item:nn {#1} {5}
1885       + 2 * (1 - #2) * (#2) * \tl_item:nn {#1} {8}
1886       + (#2)^2 * \tl_item:nn {#1} {11}
1887     }}
1888   {\fp_to_dim:n
1889     {
1890       (1 - #2)^2 * \tl_item:nn {#1} {6}
1891       + 2 * (1 - #2) * (#2) * \tl_item:nn {#1} {9}

```



```

1892     + (#2)^2 * \tl_item:nn {#1} {12}
1893   }}
1894 }
1895 \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curvetob_tl
1896 \tl_put_right:Nx \l__spath_tmpa_tl
1897 {
1898   {\fp_to_dim:n
1899   {
1900     (1 - #2) * \tl_item:nn {#1} {8} + (#2) * \tl_item:nn {#1} {11}
1901   }}
1902   {\fp_to_dim:n
1903   {
1904     (1 - #2) * \tl_item:nn {#1} {9} + (#2) * \tl_item:nn {#1} {12}
1905   }}
1906 }
1907 \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curveto_tl
1908 \tl_put_right:Nx \l__spath_tmpa_tl {
1909   {\tl_item:nn {#1} {11}}
1910   {\tl_item:nn {#1} {12}}
1911 }
1912
1913 \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpa_tl
1914 \group_end:
1915 }
1916 \cs_generate_variant:Nn \__spath_split_curve:nn {nv, nV}
1917 \cs_new_protected_nopar:Npn \spath_split_curve:NNnn #1#2#3#4
1918 {
1919   \__spath_split_curve:nn {#3}{#4}
1920   \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
1921   \tl_set:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
1922   \tl_gclear:N \g__spath_output_tl
1923 }
1924 \cs_generate_variant:Nn \spath_split_curve:NNnn {NNnV, NNVn, NNVV}
1925 \cs_new_protected_nopar:Npn \spath_gsplit_curve:NNnn #1#2#3#4
1926 {
1927   \__spath_split_curve:nn {#3}{#4}
1928   \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
1929   \tl_gset:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
1930   \tl_gclear:N \g__spath_output_tl
1931 }
1932 \cs_generate_variant:Nn \spath_gsplit_curve:NNnn {NNnV, NNVn, NNVV}

```

(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. \spath_maybe_gsplit_curve:Nn 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!

```

1933 \cs_new_protected_nopar:Npn \__spath_maybe_split_curve:n #1
1934 {
1935   \group_begin:
1936   \fp_set:Nn \l__spath_tmpa_fp
1937   {
1938     (

```

```

1939 \tl_item:nn {#1} {3}
1940 - 3 * \tl_item:nn {#1} {6}
1941 + 3 * \tl_item:nn {#1} {9}
1942 - \tl_item:nn {#1} {12}
1943 )
1944 *
1945 (3 * \tl_item:nn {#1} {8} - 3 * \tl_item:nn {#1} {11})
1946 -
1947 (
1948 \tl_item:nn {#1} {2}
1949 - 3 * \tl_item:nn {#1} {5}
1950 + 3 * \tl_item:nn {#1} {8}
1951 - \tl_item:nn {#1} {11}
1952 )
1953 *
1954 (3 * \tl_item:nn {#1} {9} - 3 * \tl_item:nn {#1} {12})
1955 }
1956 \fp_set:Nn \l__spath_tmpb_fp
1957 {
1958 (
1959 \tl_item:nn {#1} {2}
1960 - 3 * \tl_item:nn {#1} {5}
1961 + 3 * \tl_item:nn {#1} {8}
1962 - \tl_item:nn {#1} {11}
1963 )
1964 *
1965 (
1966 3 * \tl_item:nn {#1} {6}
1967 - 6 * \tl_item:nn {#1} {9}
1968 + 3 * \tl_item:nn {#1} {12}
1969 )
1970 -
1971 (
1972 \tl_item:nn {#1} {3}
1973 - 3 * \tl_item:nn {#1} {6}
1974 + 3 * \tl_item:nn {#1} {9}
1975 - \tl_item:nn {#1} {12}
1976 )
1977 *
1978 (
1979 3 * \tl_item:nn {#1} {5}
1980 - 6 * \tl_item:nn {#1} {8}
1981 + 3 * \tl_item:nn {#1} {11}
1982 )
1983 }
1984 \fp_compare:nTF
1985 {
1986 \l__spath_tmpb_fp != 0
1987 }
1988 {
1989 \fp_set:Nn \l__spath_tmpa_fp {.5 * \l__spath_tmpa_fp / \l__spath_tmpb_fp}
1990 \fp_compare:nTF
1991 {
1992 0 < \l__spath_tmpa_fp && \l__spath_tmpa_fp < 1

```

```

1993     }
1994     {
1995     \__spath_split_curve:nV {#1} \l__spath_tmpa_fp
1996     }
1997     {
1998     \tl_gset:Nn \g__spath_output_tl { {#1} {} }
1999     }
2000   }
2001   {
2002     \tl_gset:Nn \g__spath_output_tl { {#1} {} }
2003   }
2004   \group_end:
2005 }
2006 \cs_new_protected_nopar:Npn \spath_maybe_split_curve:NNn #1#2#3
2007 {
2008   \__spath_maybe_split_curve:n {#3}
2009   \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
2010   \tl_set:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
2011   \tl_gclear:N \g__spath_output_tl
2012 }
2013 \cs_generate_variant:Nn \spath_maybe_split_curve:NNn {NNn, NNV }
2014 \cs_new_protected_nopar:Npn \spath_maybe_gsplit_curve:NNn #1#2#3
2015 {
2016   \__spath_maybe_split_curve:n {#3}
2017   \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
2018   \tl_gset:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
2019   \tl_gclear:N \g__spath_output_tl
2020 }
2021 \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
2022 \cs_new_protected_nopar:Npn \__spath_split_curves:n #1
2023 {
2024   \group_begin:
2025   \tl_set:Nn \l__spath_tmpa_tl {#1}
2026   \tl_clear:N \l__spath_tmpb_tl
2027   \tl_clear:N \l__spath_tmpc_tl
2028   \bool_do_until:nm
2029   {
2030     \tl_if_empty_p:N \l__spath_tmpa_tl
2031   }
2032   {
2033     \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
2034     \tl_case:NnF \l__spath_tmpc_tl
2035     {
2036       \c_spath_curvetoa_tl
2037       {
2038         \tl_clear:N \l__spath_tmpd_tl
2039         \tl_set_eq:NN \l__spath_tmpd_tl \c_spath_moveto_tl
2040         \tl_put_right:Nx \l__spath_tmpd_tl
2041         {
2042           { \dim_use:N \l__spath_tmpa_dim }

```

```

2043     { \dim_use:N \l__spath_tmpb_dim }
2044   }
2045   \dim_set:Nn \l__spath_tmpa_dim
2046   {
2047     \tl_item:Nn \l__spath_tmpa_tl {8}
2048   }
2049   \dim_set:Nn \l__spath_tmpb_dim
2050   {
2051     \tl_item:Nn \l__spath_tmpa_tl {9}
2052   }
2053   \prg_replicate:nn {3}
2054   {
2055     \tl_put_right:Nx \l__spath_tmpd_tl
2056     {
2057       \tl_item:Nn \l__spath_tmpa_tl {1}
2058       {\tl_item:Nn \l__spath_tmpa_tl {2}}
2059       {\tl_item:Nn \l__spath_tmpa_tl {3}}
2060     }
2061     \prg_replicate:nn {3}
2062     {
2063       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
2064     }
2065   }
2066
2067   \spath_maybe_split_curve:NNV
2068   \l__spath_tmpd_tl
2069   \l__spath_tmpe_tl
2070   \l__spath_tmpd_tl
2071   \prg_replicate:nn {3}
2072   {
2073     \tl_set:Nx \l__spath_tmpd_tl {\tl_tail:N \l__spath_tmpd_tl}
2074     \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl}
2075   }
2076   \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
2077   \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpe_tl
2078 }
2079 }
2080 {
2081   \dim_set:Nn \l__spath_tmpa_dim
2082   {
2083     \tl_item:Nn \l__spath_tmpa_tl {2}
2084   }
2085   \dim_set:Nn \l__spath_tmpb_dim
2086   {
2087     \tl_item:Nn \l__spath_tmpa_tl {3}
2088   }
2089   \tl_put_right:Nx \l__spath_tmpb_tl
2090   {
2091     \tl_item:Nn \l__spath_tmpa_tl {1}
2092     {\tl_item:Nn \l__spath_tmpa_tl {2}}
2093     {\tl_item:Nn \l__spath_tmpa_tl {3}}
2094   }
2095   \prg_replicate:nn {3}
2096   {

```

```

2097         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
2098     }
2099 }
2100 }
2101 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
2102 \group_end:
2103 }
2104 \cs_new_protected_nopar:Npn \spath_split_curves:Nn #1#2
2105 {
2106     \__spath_split_curves:n {#2}
2107     \tl_set_eq:NN #1 \g__spath_output_tl
2108     \tl_gclear:N \g__spath_output_tl
2109 }
2110 \cs_generate_variant:Nn \spath_split_curves:Nn {NV, cV, cn, cv }
2111 \cs_new_protected_nopar:Npn \spath_split_curves:N #1
2112 {
2113     \spath_split_curves:NV #1#1
2114 }
2115 \cs_generate_variant:Nn \spath_split_curves:N {c}
2116 \cs_new_protected_nopar:Npn \spath_gsplit_curves:Nn #1#2
2117 {
2118     \__spath_split_curves:n {#2}
2119     \tl_gset_eq:NN #1 \g__spath_output_tl
2120     \tl_gclear:N \g__spath_output_tl
2121 }
2122 \cs_generate_variant:Nn \spath_gsplit_curves:Nn {NV, cV, cn, cv }
2123 \cs_new_protected_nopar:Npn \spath_gsplit_curves:N #1
2124 {
2125     \spath_gsplit_curves:NV #1#1
2126 }
2127 \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.

```

2128 \cs_new_protected_nopar:Npn \__spath_split_line:nn #1#2
2129 {
2130     \group_begin:
2131     \tl_set_eq:NN \l__spath_tmpa_tl \c_spath_moveto_tl
2132     \tl_put_right:Nx \l__spath_tmpa_tl {
2133         {\tl_item:nn {#1} {2}}
2134         {\tl_item:nn {#1} {3}}
2135     }
2136     \tl_put_right:NV \l__spath_tmpa_tl \c_spath_lineto_tl
2137     \tl_put_right:Nx \l__spath_tmpa_tl
2138     {
2139         {\fp_to_dim:n
2140         {
2141             (1 - #2) * \tl_item:nn {#1} {2} + (#2) * \tl_item:nn {#1} {5}
2142         }}
2143         {\fp_to_dim:n
2144         {
2145             (1 - #2) * \tl_item:nn {#1} {3} + (#2) * \tl_item:nn {#1} {6}
2146         }}

```

```

2147 }
2148 \tl_gclear:N \g__spath_output_tl
2149 \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpa_tl
2150
2151 \tl_clear:N \l__spath_tmpa_tl
2152 \tl_set_eq:NN \l__spath_tmpa_tl \c_spath_moveto_tl
2153 \tl_put_right:Nx \l__spath_tmpa_tl
2154 {
2155   {\fp_to_dim:n
2156   {
2157     (1 - #2) * \tl_item:nn {#1} {2} + (#2) * \tl_item:nn {#1} {5}
2158   }}
2159   {\fp_to_dim:n
2160   {
2161     (1 - #2) * \tl_item:nn {#1} {3} + (#2) * \tl_item:nn {#1} {6}
2162   }}
2163 }
2164 \tl_put_right:NV \l__spath_tmpa_tl \c_spath_lineto_tl
2165 \tl_put_right:Nx \l__spath_tmpa_tl {
2166   {\tl_item:nn {#1} {5}}
2167   {\tl_item:nn {#1} {6}}
2168 }
2169
2170 \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpa_tl
2171 \group_end:
2172 }
2173 \cs_new_protected_nopar:Npn \spath_split_line:NNnn #1#2#3#4
2174 {
2175   \__spath_split_line:nn {#3}{#4}
2176   \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
2177   \tl_set:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
2178   \tl_gclear:N \g__spath_output_tl
2179 }
2180 \cs_generate_variant:Nn \spath_split_line:NNnn {NNnV, NNvn, NNVV}
2181 \cs_new_protected_nopar:Npn \spath_gsplit_line:NNnn #1#2#3#4
2182 {
2183   \__spath_split_line:nn {#3}{#4}
2184   \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
2185   \tl_gset:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
2186   \tl_gclear:N \g__spath_output_tl
2187 }
2188 \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_gsplit_at:NNnn\spath_gsplit_at:Nnn
\spath_gsplit_at:Nn
2189 \cs_new_protected_nopar:Npn \__spath_split_at:nn #1#2
2190 {
2191   \group_begin:
2192   \int_set:Nn \l__spath_tmpa_int {\fp_to_int:n {floor(#2) + 1}}
2193   \fp_set:Nn \l__spath_tmpa_fp {#2 - floor(#2)}
2194

```

```

2195 % Is split point near one end or other of a component?
2196 \fp_compare:nT
2197 {
2198   \l__spath_tmpa_fp < 0.01
2199 }
2200 {
2201   % Near the start, so we'll place it at the start
2202   \fp_set:Nn \l__spath_tmpa_fp {0}
2203 }
2204 \fp_compare:nT
2205 {
2206   \l__spath_tmpa_fp > 0.99
2207 }
2208 {
2209   % Near the end, so we'll place it at the end
2210   \fp_set:Nn \l__spath_tmpa_fp {0}
2211   \int_incr:N \l__spath_tmpa_int
2212 }
2213
2214 \int_zero:N \l__spath_tmpb_int
2215 \bool_set_true:N \l__spath_tmpa_bool
2216
2217 \tl_set:Nn \l__spath_tmpe_tl {#1}
2218 \tl_clear:N \l__spath_tmpe_tl
2219
2220 \dim_zero:N \l__spath_tmpa_dim
2221 \dim_zero:N \l__spath_tmpb_dim
2222
2223 \bool_until_do:nn {
2224   \tl_if_empty_p:N \l__spath_tmpe_tl
2225   ||
2226   \int_compare_p:n { \l__spath_tmpa_int == \l__spath_tmpb_int }
2227 }
2228 {
2229   \tl_set:Nx \l__spath_tmpf_tl {\tl_head:N \l__spath_tmpe_tl}
2230   \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2231   \tl_case:Nn \l__spath_tmpf_tl
2232   {
2233     \c_spath_lineto_tl
2234     {
2235       \int_incr:N \l__spath_tmpb_int
2236     }
2237     \c_spath_curvetoa_tl
2238     {
2239       \int_incr:N \l__spath_tmpb_int
2240     }
2241   }
2242   \int_compare:nT { \l__spath_tmpb_int < \l__spath_tmpa_int }
2243   {
2244     \tl_put_right:NV \l__spath_tmpe_tl \l__spath_tmpf_tl
2245
2246     \tl_put_right:Nx \l__spath_tmpe_tl
2247     {{ \tl_head:N \l__spath_tmpe_tl }}
2248     \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpe_tl}

```

```

2249     \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2250
2251     \tl_put_right:Nx \l__spath_tmpe_tl
2252     {{ \tl_head:N \l__spath_tmpe_tl }}
2253     \dim_set:Nn \l__spath_tmpe_dim {\tl_head:N \l__spath_tmpe_tl}
2254     \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2255
2256   }
2257 }
2258
2259 \tl_clear:N \l__spath_tmpd_tl
2260 \tl_put_right:NV \l__spath_tmpd_tl \c_spath_moveto_tl
2261 \tl_put_right:Nx \l__spath_tmpd_tl
2262 {
2263   {\dim_use:N \l__spath_tmpe_dim}
2264   {\dim_use:N \l__spath_tmpe_dim}
2265 }
2266
2267 \fp_compare:nTF
2268 {
2269   \l__spath_tmpe_fp == 0
2270 }
2271 {
2272   \tl_set_eq:NN \l__spath_tmpe_tl \l__spath_tmpd_tl
2273   \tl_if_empty:NF \l__spath_tmpe_tl
2274   {
2275     \tl_put_right:NV \l__spath_tmpe_tl \l__spath_tmpf_tl
2276     \tl_put_right:NV \l__spath_tmpe_tl \l__spath_tmpe_tl
2277   }
2278 }
2279 {
2280
2281   \tl_case:Nn \l__spath_tmpe_tl
2282   {
2283     \c_spath_lineto_tl
2284     {
2285       \tl_put_right:NV \l__spath_tmpe_tl \l__spath_tmpf_tl
2286       \tl_put_right:Nx \l__spath_tmpe_tl
2287       {{ \tl_head:N \l__spath_tmpe_tl }}
2288       \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2289
2290       \tl_put_right:Nx \l__spath_tmpe_tl
2291       {{ \tl_head:N \l__spath_tmpe_tl }}
2292       \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2293
2294       \spath_split_line:NNVV
2295       \l__spath_tmpe_tl
2296       \l__spath_tmpe_tl
2297       \l__spath_tmpe_tl
2298       \l__spath_tmpe_tl
2299
2300       \prg_replicate:nn {3} {
2301         \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl}
2302       }

```



```

2303
2304     \tl_put_right:NV \l__spath_tmpc_tl \l__spath_tmpa_tl
2305     \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpe_tl
2306   }
2307   \c_spath_curvetoa_tl
2308   {
2309     \tl_put_right:NV \l__spath_tmpd_tl \l__spath_tmpf_tl
2310     \tl_put_right:Nx \l__spath_tmpd_tl
2311     {{ \tl_head:N \l__spath_tmpe_tl }}
2312     \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2313
2314     \tl_put_right:Nx \l__spath_tmpd_tl
2315     {{ \tl_head:N \l__spath_tmpe_tl }}
2316     \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2317
2318     \prg_replicate:nn {2} {
2319
2320       \tl_put_right:Nx \l__spath_tmpd_tl
2321       { \tl_head:N \l__spath_tmpe_tl }
2322       \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2323
2324       \tl_put_right:Nx \l__spath_tmpd_tl
2325       {{ \tl_head:N \l__spath_tmpe_tl }}
2326       \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2327
2328       \tl_put_right:Nx \l__spath_tmpd_tl
2329       {{ \tl_head:N \l__spath_tmpe_tl }}
2330       \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2331     }
2332
2333     \spath_split_curve:NNVV
2334     \l__spath_tmpa_tl
2335     \l__spath_tmpb_tl
2336     \l__spath_tmpd_tl \l__spath_tmpa_fp
2337
2338     \prg_replicate:nn {3} {
2339       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
2340     }
2341
2342     \tl_put_right:NV \l__spath_tmpc_tl \l__spath_tmpa_tl
2343     \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpe_tl
2344   }
2345 }
2346 }
2347
2348 \tl_gclear:N \g__spath_output_tl
2349 \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpc_tl
2350 \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpb_tl
2351 \group_end:
2352 }
2353 \cs_new_protected_nopar:Npn \spath_split_at:NNnn #1#2#3#4
2354 {
2355   \__spath_split_at:nn {#3}{#4}
2356   \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}

```

```

2357 \tl_set:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
2358 \tl_gclear:N \g__spath_output_tl
2359 }
2360 \cs_generate_variant:Nn \spath_split_at:NNnn {NNVn, NNVV, NNnV}
2361 \cs_new_protected_nopar:Npn \spath_gsplit_at:NNnn #1#2#3#4
2362 {
2363   \__spath_split_at:nn {#3}{#4}
2364   \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
2365   \tl_gset:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
2366   \tl_gclear:N \g__spath_output_tl
2367 }
2368 \cs_generate_variant:Nn \spath_gsplit_at:NNnn {NNVn, NNVV, NNnV}
2369 \cs_new_protected_nopar:Npn \spath_split_at:Nnn #1#2#3
2370 {
2371   \__spath_split_at:nn {#2}{#3}
2372   \tl_set:Nx #1
2373   {
2374     \tl_item:Nn \g__spath_output_tl {1}
2375     \tl_item:Nn \g__spath_output_tl {2}
2376   }
2377   \tl_gclear:N \g__spath_output_tl
2378 }
2379 \cs_generate_variant:Nn \spath_split_at:Nnn {NVn, NVV}
2380 \cs_new_protected_nopar:Npn \spath_split_at:Nn #1#2
2381 {
2382   \spath_split_at:NVn #1#1{#2}
2383 }
2384 \cs_new_protected_nopar:Npn \spath_gsplit_at:Nnn #1#2#3
2385 {
2386   \__spath_split_at:nn {#2}{#3}
2387   \tl_gset:Nx #1
2388   {
2389     \tl_item:Nn \g__spath_output_tl {1}
2390     \tl_item:Nn \g__spath_output_tl {2}
2391   }
2392   \tl_gclear:N \g__spath_output_tl
2393 }
2394 \cs_generate_variant:Nn \spath_gsplit_at:Nnn {NVn, NVV}
2395 \cs_new_protected_nopar:Npn \spath_gsplit_at:Nn #1#2
2396 {
2397   \spath_gsplit_at:NVn #1#1{#2}
2398 }

```

(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.

```
2399 \cs_new_protected_nopar:Npn \__spath_shorten_at_end:nn #1#2
2400 {
2401   \int_compare:nTF
2402   {
2403     \tl_count:n {#1} > 3
2404   }
2405   {
2406     \group_begin:
2407     \tl_set:Nn \l__spath_tmpa_tl {#1}
2408     \tl_reverse:N \l__spath_tmpa_tl
2409
2410     \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {3}}
2411
2412     \tl_clear:N \l__spath_tmpe_tl
2413     \tl_if_eq:NNTF \l__spath_tmpb_tl \c_spath_curveto_tl
2414     {
2415       \int_set:Nn \l__spath_tmpa_int {3}
2416     }
2417     {
2418       \int_set:Nn \l__spath_tmpa_int {1}
2419     }
2420
2421     \prg_replicate:nn { \l__spath_tmpa_int }
2422     {
2423       \tl_put_right:Nx \l__spath_tmpe_tl
2424       {
2425         {\tl_head:N \l__spath_tmpa_tl}
2426       }
2427       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
2428       \tl_put_right:Nx \l__spath_tmpe_tl
2429       {
2430         {\tl_head:N \l__spath_tmpa_tl}
2431       }
2432       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
2433       \tl_put_right:Nx \l__spath_tmpe_tl
2434       {
2435         \tl_head:N \l__spath_tmpa_tl
2436       }
2437       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
2438     }
2439
2440     \tl_put_right:Nx \l__spath_tmpe_tl
2441     {
2442       {\tl_item:Nn \l__spath_tmpa_tl {1}}
2443       {\tl_item:Nn \l__spath_tmpa_tl {2}}
2444     }
2445     \tl_put_right:NV \l__spath_tmpe_tl \c_spath_moveto_tl
2446
2447     \tl_reverse:N \l__spath_tmpa_tl
2448
2449     \fp_set:Nn \l__spath_tmpa_fp
2450     {
2451       \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {4}}
```

```

2452     -
2453     \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {1}}
2454 }
2455
2456 \fp_set:Nn \l__spath_tmpb_fp
2457 {
2458     \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {5}}
2459     -
2460     \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {2}}
2461 }
2462
2463 \fp_set:Nn \l__spath_tmpe_fp
2464 {
2465     sqrt(
2466     \l__spath_tmpe_fp * \l__spath_tmpe_fp
2467     +
2468     \l__spath_tmpb_fp * \l__spath_tmpb_fp
2469     ) * \l__spath_tmpe_int
2470 }
2471
2472 \fp_compare:nTF
2473 {
2474     \l__spath_tmpe_fp > #2
2475 }
2476 {
2477
2478     \fp_set:Nn \l__spath_tmpe_fp
2479     {
2480         (\l__spath_tmpe_fp - #2) / \l__spath_tmpe_fp
2481     }
2482
2483     \tl_reverse:N \l__spath_tmpe_tl
2484
2485     \tl_if_eq:NNTF \l__spath_tmpe_tl \c_spath_curveto_tl
2486     {
2487         \spath_split_curve:NNVV
2488         \l__spath_tmpe_tl
2489         \l__spath_tmpe_tl
2490         \l__spath_tmpe_tl
2491         \l__spath_tmpe_fp
2492     }
2493     {
2494         \spath_split_line:NNVV
2495         \l__spath_tmpe_tl
2496         \l__spath_tmpe_tl
2497         \l__spath_tmpe_tl
2498         \l__spath_tmpe_fp
2499     }
2500
2501     \prg_replicate:nn {3}
2502     {
2503         \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl}
2504     }
2505

```

```

2506     \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpe_tl
2507   }
2508   {
2509
2510
2511     \int_compare:nT
2512     {
2513       \tl_count:N \l__spath_tmpa_tl > 3
2514     }
2515     {
2516       \dim_set:Nn \l__spath_tmpa_dim {\fp_to_dim:n {#2 - \l__spath_tmpe_fp } }
2517       \spath_shorten_at_end:NV \l__spath_tmpa_tl \l__spath_tmpa_dim
2518     }
2519   }
2520
2521   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
2522   \group_end:
2523 }
2524 {
2525   \tl_gset:Nn \g__spath_output_tl {#1}
2526 }
2527 }
2528 \cs_new_protected_nopar:Npn \spath_shorten_at_end:Nnn #1#2#3
2529 {
2530   \__spath_shorten_at_end:nn {#2}{#3}
2531   \tl_set_eq:NN #1 \g__spath_output_tl
2532   \tl_gclear:N \g__spath_output_tl
2533 }
2534 \cs_generate_variant:Nn \spath_shorten_at_end:Nnn {NVV, cnn, cVV, NVn}
2535 \cs_new_protected_nopar:Npn \spath_shorten_at_end:Nn #1#2
2536 {
2537   \spath_shorten_at_end:NVn #1#1{#2}
2538 }
2539 \cs_generate_variant:Nn \spath_shorten_at_end:Nn {cn, cV, NV}
2540 \cs_new_protected_nopar:Npn \spath_gshorten_at_end:Nnn #1#2#3
2541 {
2542   \__spath_shorten_at_end:nn {#2}{#3}
2543   \tl_gset_eq:NN #1 \g__spath_output_tl
2544   \tl_gclear:N \g__spath_output_tl
2545 }
2546 \cs_generate_variant:Nn \spath_gshorten_at_end:Nnn {NVV, cnn, cVV, NVn}
2547 \cs_new_protected_nopar:Npn \spath_gshorten_at_end:Nn #1#2
2548 {
2549   \spath_gshorten_at_end:NVn #1#1{#2}
2550 }
2551 \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 This macro shortens a path from the start by a dimension.
\spath_shorten_at_start:Nn
\spath_gshorten_at_start:Nnn
\spath_gshorten_at_start:Nn
2552 \cs_new_protected_nopar:Npn \__spath_shorten_at_start:nn #1#2
2553 {
2554   \int_compare:nTF
2555   {

```

```

2556     \tl_count:n {#1} > 3
2557   }
2558   {
2559     \group_begin:
2560     \tl_set:Nn \l__spath_tmpa_tl {#1}
2561
2562     \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {4}}
2563
2564     \tl_clear:N \l__spath_tmpe_tl
2565
2566     \tl_if_eq:NNTF \l__spath_tmpb_tl \c_spath_curvetoa_tl
2567     {
2568       \int_set:Nn \l__spath_tmpe_int {3}
2569     }
2570     {
2571       \int_set:Nn \l__spath_tmpe_int {1}
2572     }
2573
2574     \tl_set_eq:NN \l__spath_tmpe_tl \c_spath_moveto_tl
2575     \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2576
2577     \prg_replicate:nn { \l__spath_tmpe_int }
2578     {
2579       \__spath_tl_put_right_braced:Nx
2580       \l__spath_tmpe_tl
2581       {\tl_item:Nn \l__spath_tmpe_tl {1}}
2582       \__spath_tl_put_right_braced:Nx
2583       \l__spath_tmpe_tl
2584       {\tl_item:Nn \l__spath_tmpe_tl {2}}
2585       \tl_put_right:Nx \l__spath_tmpe_tl {\tl_item:Nn \l__spath_tmpe_tl {3}}
2586
2587       \prg_replicate:nn {3}
2588       {
2589         \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
2590       }
2591     }
2592     \__spath_tl_put_right_braced:Nx
2593     \l__spath_tmpe_tl
2594     {\tl_item:Nn \l__spath_tmpe_tl {1}}
2595     \__spath_tl_put_right_braced:Nx
2596     \l__spath_tmpe_tl
2597     {\tl_item:Nn \l__spath_tmpe_tl {2}}
2598
2599     \fp_set:Nn \l__spath_tmpe_fp
2600     {
2601       \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {5}}
2602       -
2603       \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {2}}
2604     }
2605
2606     \fp_set:Nn \l__spath_tmpe_fp
2607     {
2608       \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {6}}
2609       -

```

```

2610   \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {3}}
2611 }
2612
2613 \fp_set:Nn \l__spath_tmpc_fp
2614 {
2615   sqrt(
2616     \l__spath_tmpe_tl * \l__spath_tmpe_tl
2617     +
2618     \l__spath_tmpe_tl * \l__spath_tmpe_tl
2619   )
2620   *
2621   \l__spath_tmpe_tl
2622 }
2623
2624 \fp_compare:nTF
2625 {
2626   \l__spath_tmpc_fp > #2
2627 }
2628 {
2629
2630   \fp_set:Nn \l__spath_tmpe_tl
2631   {
2632     #2/ \l__spath_tmpe_tl
2633   }
2634
2635   \tl_if_eq:NNTF \l__spath_tmpe_tl \c_spath_curvetoa_tl
2636   {
2637     \spath_split_curve:NNVV
2638     \l__spath_tmpe_tl
2639     \l__spath_tmpe_tl
2640     \l__spath_tmpe_tl
2641     \l__spath_tmpe_tl
2642   }
2643   {
2644     \spath_split_line:NNVV
2645     \l__spath_tmpe_tl
2646     \l__spath_tmpe_tl
2647     \l__spath_tmpe_tl
2648     \l__spath_tmpe_tl
2649   }
2650
2651   \prg_replicate:nn {2}
2652   {
2653     \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl}
2654   }
2655
2656   \tl_put_left:NV \l__spath_tmpe_tl \l__spath_tmpe_tl
2657
2658 }
2659 {
2660
2661   \tl_put_left:NV \l__spath_tmpe_tl \c_spath_moveto_tl
2662
2663   \int_compare:nT

```

```

2664     {
2665     \tl_count:N \l__spath_tmpa_tl > 3
2666     }
2667     {
2668     \dim_set:Nn \l__spath_tmpa_dim {\fp_to_dim:n {#2 - \l__spath_tmpc_fp } }
2669     \spath_shorten_at_start:NV \l__spath_tmpa_tl \l__spath_tmpa_dim
2670     }
2671   }
2672
2673   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
2674   \group_end:
2675   }
2676   {
2677     \tl_gset:Nn \g__spath_output_tl {#1}
2678   }
2679 }
2680 \cs_new_protected_nopar:Npn \spath_shorten_at_start:Nnn #1#2#3
2681 {
2682   \__spath_shorten_at_start:nn {#2}{#3}
2683   \tl_set_eq:NN #1 \g__spath_output_tl
2684   \tl_gclear:N \g__spath_output_tl
2685 }
2686 \cs_generate_variant:Nn \spath_shorten_at_start:Nnn {NVV, cnn, cVV, NVn}
2687 \cs_new_protected_nopar:Npn \spath_shorten_at_start:Nn #1#2
2688 {
2689   \spath_shorten_at_start:NVn #1#1{#2}
2690 }
2691 \cs_generate_variant:Nn \spath_shorten_at_start:Nn {cn, cV, NV}
2692 \cs_new_protected_nopar:Npn \spath_gshorten_at_start:Nnn #1#2#3
2693 {
2694   \__spath_shorten_at_start:nn {#2}{#3}
2695   \tl_gset_eq:NN #1 \g__spath_output_tl
2696   \tl_gclear:N \g__spath_output_tl
2697 }
2698 \cs_generate_variant:Nn \spath_gshorten_at_start:Nnn {NVV, cnn, cVV, NVn}
2699 \cs_new_protected_nopar:Npn \spath_gshorten_at_start:Nn #1#2
2700 {
2701   \spath_gshorten_at_start:NVn #1#1{#2}
2702 }
2703 \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` `\spath_gpoint_at:Nnn` Get the location of a point on a path, using the same location specification as the intersection library.

```

2704 \cs_new_protected_nopar:Npn \__spath_point_at:nn #1#2
2705 {
2706   \group_begin:
2707   \int_set:Nn \l__spath_tmpa_int {\fp_to_int:n {floor(#2) + 1}}
2708   \fp_set:Nn \l__spath_tmpa_fp {#2 - floor(#2)}
2709
2710   \spath_segments_to_seq:Nn \l__spath_tmpa_seq {#1}

```



```

2711
2712 \int_compare:nTF
2713 {
2714   \l__spath_tmpa_int < 1
2715 }
2716 {
2717   \spath_initialpoint:Nn \l__spath_tmpc_tl {#1}
2718 }
2719 {
2720   \int_compare:nTF
2721   {
2722     \l__spath_tmpa_int > \seq_count:N \l__spath_tmpa_seq
2723   }
2724   {
2725     \spath_finalpoint:Nn \l__spath_tmpc_tl {#1}
2726   }
2727   {
2728
2729     \tl_set:Nx
2730     \l__spath_tmpa_tl
2731     {\seq_item:Nn \l__spath_tmpa_seq { \l__spath_tmpa_int } }
2732
2733     \int_compare:nTF
2734     {
2735       \tl_count:N \l__spath_tmpa_tl > 3
2736     }
2737     {
2738       \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {4}}
2739     }
2740     {
2741       \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {1}}
2742     }
2743
2744     \tl_clear:N \l__spath_tmpc_tl
2745
2746     \tl_case:Nn \l__spath_tmpb_tl
2747     {
2748       \c_spath_moveto_tl
2749       {
2750         \tl_set:Nx \l__spath_tmpc_tl
2751         {
2752           {
2753             \tl_item:Nn \l__spath_tmpa_tl {2}
2754           }
2755           {
2756             \tl_item:Nn \l__spath_tmpa_tl {3}
2757           }
2758         }
2759       }
2760
2761       \c_spath_lineto_tl
2762       {
2763         \tl_set:Nx \l__spath_tmpc_tl
2764         {

```

```

2765     {\fp_to_dim:n
2766     {
2767         (1 - \l__spath_tmpa_fp) * ( \tl_item:Nn \l__spath_tmpa_tl {2} )
2768         +
2769         \l__spath_tmpa_fp * ( \tl_item:Nn \l__spath_tmpa_tl {5} )
2770     }
2771 }
2772 {\fp_to_dim:n
2773 {
2774     (1 - \l__spath_tmpa_fp) * ( \tl_item:Nn \l__spath_tmpa_tl {3} )
2775     +
2776     \l__spath_tmpa_fp * ( \tl_item:Nn \l__spath_tmpa_tl {6} )
2777 }
2778 }
2779 }
2780 }
2781
2782 \c_spath_closepath_tl
2783 {
2784     \tl_set:Nx \l__spath_tmpc_tl
2785     {
2786         {\fp_to_dim:n
2787         {
2788             (1 - \l__spath_tmpa_fp) * ( \tl_item:Nn \l__spath_tmpa_tl {2} )
2789             +
2790             \l__spath_tmpa_fp * ( \tl_item:Nn \l__spath_tmpa_tl {5} )
2791         }
2792         }
2793         {\fp_to_dim:n
2794         {
2795             (1 - \l__spath_tmpa_fp) * ( \tl_item:Nn \l__spath_tmpa_tl {3} )
2796             +
2797             \l__spath_tmpa_fp * ( \tl_item:Nn \l__spath_tmpa_tl {6} )
2798         }
2799         }
2800     }
2801 }
2802
2803 \c_spath_curvetoa_tl
2804 {
2805     \tl_set:Nx \l__spath_tmpc_tl
2806     {
2807         {\fp_to_dim:n
2808         {
2809             (1 - \l__spath_tmpa_fp)^3 * \tl_item:Nn \l__spath_tmpa_tl {2}
2810             + 3 * (1 - \l__spath_tmpa_fp)^2 * (\l__spath_tmpa_fp)
2811             * \tl_item:Nn \l__spath_tmpa_tl {5}
2812             + 3 * (1 - \l__spath_tmpa_fp) * (\l__spath_tmpa_fp)^2
2813             * \tl_item:Nn \l__spath_tmpa_tl {8}
2814             + (\l__spath_tmpa_fp)^3 * \tl_item:Nn \l__spath_tmpa_tl {11}
2815         }}
2816         {\fp_to_dim:n
2817         {
2818             (1 - \l__spath_tmpa_fp)^3 * \tl_item:Nn \l__spath_tmpa_tl {3}

```

```

2819         + 3 * (1 - \l__spath_tmpa_fp)^2 * (\l__spath_tmpa_fp)
2820         * \tl_item:Nn \l__spath_tmpa_tl {6}
2821         + 3 * (1 - \l__spath_tmpa_fp) * (\l__spath_tmpa_fp)^2
2822         * \tl_item:Nn \l__spath_tmpa_tl {9}
2823         + (\l__spath_tmpa_fp)^3 * \tl_item:Nn \l__spath_tmpa_tl {12}
2824     }}
2825   }
2826 }
2827 }
2828 }
2829 }
2830
2831 \tl_gclear:N \g__spath_output_tl
2832 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpe_tl
2833 \group_end:
2834 }
2835 \cs_new_protected_nopar:Npn \spath_point_at:Nnn #1#2#3
2836 {
2837   \__spath_point_at:nn {#2}{#3}
2838   \tl_set_eq:NN #1 \g__spath_output_tl
2839   \tl_gclear:N \g__spath_output_tl
2840 }
2841 \cs_generate_variant:Nn \spath_point_at:Nnn {NVn, NVV, NnV}
2842 \cs_new_protected_nopar:Npn \spath_gpoint_at:Nnn #1#2#3
2843 {
2844   \__spath_point_at:nn {#2}{#3}
2845   \tl_gset_eq:NN #1 \g__spath_output_tl
2846   \tl_gclear:N \g__spath_output_tl
2847 }
2848 \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` `\spath_gtangent_at:Nnn` Get the tangent at a point on a path, using the same location specification as the intersection library.

```

2849 \cs_new_protected_nopar:Npn \__spath_tangent_at:nn #1#2
2850 {
2851   \group_begin:
2852   \int_set:Nn \l__spath_tmpa_int {\fp_to_int:n {floor(#2) + 1}}
2853   \fp_set:Nn \l__spath_tmpa_fp {#2 - floor(#2)}
2854
2855   \spath_segments_to_seq:Nn \l__spath_tmpa_seq {#1}
2856
2857   \int_compare:nTF
2858   {
2859     \l__spath_tmpa_int < 1
2860   }
2861   {
2862     \spath_initialpoint:Nn \l__spath_tmpe_tl {#1}
2863   }
2864   {
2865     \int_compare:nTF
2866     {
2867       \l__spath_tmpa_int > \seq_count:N \l__spath_tmpa_seq

```

```

2868 }
2869 {
2870   \spath_finalpoint:Nn \l__spath_tmpc_tl {#1}
2871 }
2872 {
2873
2874   \tl_set:Nx
2875   \l__spath_tmpa_tl
2876   {\seq_item:Nn \l__spath_tmpa_seq { \l__spath_tmpa_int} }
2877
2878   \int_compare:nTF
2879   {
2880     \tl_count:N \l__spath_tmpa_tl > 3
2881   }
2882   {
2883     \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {4}}
2884   }
2885   {
2886     \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {1}}
2887   }
2888
2889   \tl_clear:N \l__spath_tmpc_tl
2890
2891   \tl_case:Nn \l__spath_tmpb_tl
2892   {
2893     \c_spath_moveto_tl
2894     {
2895       \tl_set:Nx \l__spath_tmpc_tl
2896       {
2897         {
2898           \tl_item:Nn \l__spath_tmpa_tl {2}
2899         }
2900         {
2901           \tl_item:Nn \l__spath_tmpa_tl {3}
2902         }
2903       }
2904     }
2905
2906     \c_spath_lineto_tl
2907     {
2908       \tl_set:Nx \l__spath_tmpc_tl
2909       {
2910         {\fp_to_dim:n
2911         {
2912           ( \tl_item:Nn \l__spath_tmpa_tl {5} )
2913           -
2914           ( \tl_item:Nn \l__spath_tmpa_tl {2} )
2915         }
2916         }
2917         {\fp_to_dim:n
2918         {
2919           ( \tl_item:Nn \l__spath_tmpa_tl {6} )
2920           -
2921           ( \tl_item:Nn \l__spath_tmpa_tl {3} )

```

```

2922     }
2923   }
2924 }
2925 }
2926
2927 \c_spath_closepath_tl
2928 {
2929   \tl_set:Nx \l__spath_tmpc_tl
2930   {
2931     {\fp_to_dim:n
2932       {
2933         ( \tl_item:Nn \l__spath_tmpa_tl {5} )
2934         -
2935         ( \tl_item:Nn \l__spath_tmpa_tl {2} )
2936       }
2937     }
2938     {\fp_to_dim:n
2939       {
2940         ( \tl_item:Nn \l__spath_tmpa_tl {6} )
2941         -
2942         ( \tl_item:Nn \l__spath_tmpa_tl {3} )
2943       }
2944     }
2945   }
2946 }
2947
2948 \c_spath_curvetoa_tl
2949 {
2950   \tl_set:Nx \l__spath_tmpc_tl
2951   {
2952     {\fp_to_dim:n
2953       {
2954         3*(1 - \l__spath_tmpa_fp)^2 * (\tl_item:Nn \l__spath_tmpa_tl {5}
2955         - \tl_item:Nn \l__spath_tmpa_tl {2})
2956         + 6 * (1 - \l__spath_tmpa_fp) * (\l__spath_tmpa_fp) *
2957         (\tl_item:Nn \l__spath_tmpa_tl {8}
2958         - \tl_item:Nn \l__spath_tmpa_tl {5})
2959         + 3*(\l__spath_tmpa_fp)^2 * (\tl_item:Nn \l__spath_tmpa_tl {11}
2960         - \tl_item:Nn \l__spath_tmpa_tl {8})
2961       }
2962     }
2963     {\fp_to_dim:n
2964       {
2965         3*(1 - \l__spath_tmpa_fp)^2 * (\tl_item:Nn \l__spath_tmpa_tl {6}
2966         - \tl_item:Nn \l__spath_tmpa_tl {3})
2967         + 6 * (1 - \l__spath_tmpa_fp) * (\l__spath_tmpa_fp) *
2968         (\tl_item:Nn \l__spath_tmpa_tl {9}
2969         - \tl_item:Nn \l__spath_tmpa_tl {6})
2970         + 3*(\l__spath_tmpa_fp)^2 * (\tl_item:Nn \l__spath_tmpa_tl {12}
2971         - \tl_item:Nn \l__spath_tmpa_tl {9})
2972       }
2973     }
2974   }
2975 }

```

```

2976     }
2977   }
2978
2979   \tl_gclear:N \g__spath_output_tl
2980   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpc_tl
2981   \group_end:
2982 }
2983 \cs_new_protected_nopar:Npn \spath_tangent_at:Nnn #1#2#3
2984 {
2985   \__spath_tangent_at:nn {#2}{#3}
2986   \tl_set_eq:NN #1 \g__spath_output_tl
2987   \tl_gclear:N \g__spath_output_tl
2988 }
2989 \cs_generate_variant:Nn \spath_tangent_at:Nnn {NVn, NVV, NnV}
2990 \cs_new_protected_nopar:Npn \spath_gtangent_at:Nnn #1#2#3
2991 {
2992   \__spath_tangent_at:nn {#2}{#3}
2993   \tl_gset_eq:NN #1 \g__spath_output_tl
2994   \tl_gclear:N \g__spath_output_tl
2995 }
2996 \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.
`\spath_gtransformation_at:Nnn`

```

2997 \cs_new_protected_nopar:Npn \__spath_transformation_at:nn #1#2
2998 {
2999   \group_begin:
3000   \tl_clear:N \l__spath_tmpa_tl
3001   \__spath_tangent_at:nn {#1}{#2}
3002   \tl_set_eq:NN \l__spath_tmpb_tl \g__spath_output_tl
3003   \fp_set:Nn \l__spath_tmpa_fp
3004   {
3005     sqrt(
3006       (\tl_item:Nn \l__spath_tmpb_tl {1})^2
3007       +
3008       (\tl_item:Nn \l__spath_tmpb_tl {2})^2
3009     )
3010   }
3011   \fp_compare:nTF {\l__spath_tmpa_fp = 0}
3012   {
3013     \fp_set:Nn \l__spath_tmpa_fp {1}
3014     \fp_set:Nn \l__spath_tmpb_fp {0}
3015   }
3016   {
3017     \fp_set:Nn \l__spath_tmpb_fp
3018     { (\tl_item:Nn \l__spath_tmpb_tl {2}) / \l__spath_tmpa_fp }
3019     \fp_set:Nn \l__spath_tmpa_fp
3020     { (\tl_item:Nn \l__spath_tmpb_tl {1}) / \l__spath_tmpa_fp }
3021   }
3022   \tl_set:Nx \l__spath_tmpa_tl
3023   {
3024     { \fp_to_decimal:n { \l__spath_tmpa_fp } }

```

```

3025     { \fp_to_decimal:n { \l__spath_tmpb_fp } }
3026     { \fp_to_decimal:n {- \l__spath_tmpb_fp } }
3027     { \fp_to_decimal:n { \l__spath_tmpa_fp } }
3028   }
3029   \__spath_point_at:nn {#1}{#2}
3030   \tl_put_right:NV \l__spath_tmpa_tl \g__spath_output_tl
3031   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
3032   \group_end:
3033 }
3034 \cs_new_protected_nopar:Npn \spath_transformation_at:Nnn #1#2#3
3035 {
3036   \__spath_transformation_at:nn {#2}{#3}
3037   \tl_set_eq:NN #1 \g__spath_output_tl
3038   \tl_gclear:N \g__spath_output_tl
3039 }
3040 \cs_generate_variant:Nn \spath_transformation_at:Nnn {NVn, NVV, NnV, NvV}
3041 \cs_new_protected_nopar:Npn \spath_gtransformation_at:Nnn #1#2#3
3042 {
3043   \__spath_transformation_at:nn {#2}{#3}
3044   \tl_gset_eq:NN #1 \g__spath_output_tl
3045   \tl_gclear:N \g__spath_output_tl
3046 }
3047 \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
3048 \cs_new_protected_nopar:Npn \spath_intersect:NN #1#2
3049 {
3050   \pgfintersectionofpaths%
3051   {%
3052     \pgfsetpath #1
3053   }{%
3054     \pgfsetpath #2
3055   }
3056 }
3057 \cs_new_protected_nopar:Npn \spath_intersect:nn #1#2
3058 {
3059   \tl_set:Nn \l__spath_intersecta_tl {#1}
3060   \tl_set:Nn \l__spath_intersectb_tl {#2}
3061   \spath_intersect:NN \l__spath_intersecta_tl \l__spath_intersectb_tl
3062 }

```

(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.

```

3063 \cs_new_protected_nopar:Npn \__spath_split_component_at_intersections:nn #1#2
3064 {
3065   \group_begin:
3066   \tl_clear:N \l__spath_tmpe_tl
3067   \seq_clear:N \l__spath_tmpb_seq
3068
3069   % Find the intersections of these segments
3070   \tl_set:Nn \l__spath_tmpb_tl {#1}
3071   \tl_set:Nn \l__spath_tmpc_tl {#2}
3072
3073   \spath_reallength:Nn \l__spath_tmpa_int {#1}
3074
3075   % Remember if the component is closed
3076   \spath_finalaction:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
3077
3078   \bool_set:Nn \l__spath_closed_bool
3079   {
3080     \tl_if_eq_p:NN \l__spath_tmpa_tl \c_spath_closepath_tl
3081   }
3082
3083   % Open it
3084   \spath_open:N \l__spath_tmpb_tl
3085
3086   % Sort intersections along the component
3087   \pgfintersectionsortbyfirstpath
3088   \spath_intersect:NN \l__spath_tmpb_tl \l__spath_tmpc_tl
3089
3090   % If we get intersections
3091   \int_compare:nT {\pgfintersectionsolutions > 0}
3092   {
3093     % Find the times of the intersections on the component
3094     \int_step_inline:nnnn {1} {1} {\pgfintersectionsolutions}
3095     {
3096       \pgfintersectiongetsolutiontimes{##1}{\l__spath_tmph_tl}{\l__spath_tmpl_tl}
3097       \seq_put_left:NV \l__spath_tmpb_seq \l__spath_tmph_tl
3098     }
3099
3100     \seq_get_left:NN \l__spath_tmpb_seq \l__spath_tmpa_tl
3101     \fp_compare:nT
3102     {
3103       \l__spath_tmpa_tl > \l__spath_tmpa_int - .01
3104     }
3105     {
3106       \bool_set_false:N \l__spath_closed_bool
3107     }
3108     \seq_get_right:NN \l__spath_tmpb_seq \l__spath_tmpa_tl
3109     \fp_compare:nT
3110     {
3111       \l__spath_tmpa_tl < .01
3112     }
3113     {
3114       \bool_set_false:N \l__spath_closed_bool
3115     }
3116

```



```

3117
3118 \tl_set:Nn \l__spath_tmpg_tl {-1}
3119
3120 \seq_map_inline:Nn \l__spath_tmpb_seq
3121 {
3122   \tl_set:Nn \l__spath_tmph_tl {##1}
3123
3124   \tl_set_eq:NN \l__spath_tmpa_tl \l__spath_tmph_tl
3125   \int_compare:nT
3126   {
3127     \fp_to_int:n {floor( \l__spath_tmph_tl ) }
3128     =
3129     \fp_to_int:n {floor( \l__spath_tmpg_tl ) }
3130   }
3131   {
3132     \tl_set:Nx \l__spath_tmph_tl
3133     {
3134       \fp_eval:n {
3135         floor( \l__spath_tmph_tl )
3136         +
3137         ( \l__spath_tmph_tl - floor( \l__spath_tmph_tl ) )
3138         /
3139         ( \l__spath_tmpg_tl - floor( \l__spath_tmpg_tl ) )
3140       }
3141     }
3142   }
3143   \tl_set_eq:NN \l__spath_tmpg_tl \l__spath_tmpa_tl
3144
3145   \spath_split_at:NNVV
3146   \l__spath_tmpd_tl
3147   \l__spath_tmpe_tl
3148   \l__spath_tmpe_tl
3149   \l__spath_tmpe_tl
3150
3151   \tl_put_left:NV \l__spath_tmpe_tl \l__spath_tmpe_tl
3152   \tl_set_eq:NN \l__spath_tmpe_tl \l__spath_tmpe_tl
3153
3154 }
3155
3156
3157 \tl_put_left:NV \l__spath_tmpe_tl \l__spath_tmpe_tl
3158
3159 \spath_remove_empty_components:N \l__spath_tmpe_tl
3160
3161 \tl_set_eq:NN \l__spath_tmpe_tl \l__spath_tmpe_tl
3162 }
3163
3164 \bool_if:NT \l__spath_closed_bool
3165 {
3166   \spath_join_component:Nn \l__spath_tmpe_tl {1}
3167 }
3168
3169 \tl_gclear:N \g__spath_output_tl
3170 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpe_tl

```

```

3171
3172   \group_end:
3173 }
3174 \cs_new_protected_nopar:Npn \spath_split_component_at_intersections:Nnn #1#2#3
3175 {
3176   \__spath_split_component_at_intersections:nn {#2}{#3}
3177   \tl_set_eq:NN #1 \g__spath_output_tl
3178   \tl_gclear:N \g__spath_output_tl
3179 }
3180 \cs_generate_variant:Nn \spath_split_component_at_intersections:Nnn {NVn, NVV}
3181 \cs_new_protected_nopar:Npn \spath_split_component_at_intersections:Nn #1#2
3182 {
3183   \spath_split_component_at_intersections:NVn #1#1{#2}
3184 }
3185 \cs_generate_variant:Nn \spath_split_component_at_intersections:Nn {cn, cv}
3186 \cs_new_protected_nopar:Npn \spath_gsplit_component_at_intersections:Nnn #1#2#3
3187 {
3188   \__spath_split_component_at_intersections:nn {#2}{#3}
3189   \tl_gset_eq:NN #1 \g__spath_output_tl
3190   \tl_gclear:N \g__spath_output_tl
3191 }
3192 \cs_generate_variant:Nn \spath_gsplit_component_at_intersections:Nnn {NVn, NVV}
3193 \cs_new_protected_nopar:Npn \spath_gsplit_component_at_intersections:Nn #1#2
3194 {
3195   \spath_gsplit_component_at_intersections:NVn #1#1{#2}
3196 }
3197 \cs_generate_variant:Nn \spath_gsplit_component_at_intersections:Nn {cn, cv}

```

(End definition for \spath_split_component_at_intersections:Nnn.)

`\spath_split_path_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: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
3198 \cs_new_protected_nopar:Npn \__spath_split_path_at_intersections:nn #1#2
3199 {
3200   \group_begin:
3201
3202   \seq_clear:N \l__spath_tmpa_seq
3203   \seq_clear:N \l__spath_tmpb_seq
3204
3205   \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
3206   \seq_map_inline:Nn \l__spath_tmpa_seq
3207   {
3208     \spath_split_component_at_intersections:Nnn \l__spath_tmpa_tl {##1} {#2}
3209     \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
3210   }
3211
3212   \tl_gclear:N \g__spath_output_tl
3213   \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpb_seq {}}
3214   \group_end:
3215 }
3216 \cs_new_protected_nopar:Npn \spath_split_path_at_intersections:Nnn #1#2#3
3217 {
3218   \__spath_split_path_at_intersections:nn {#2}{#3}
3219   \tl_set_eq:NN #1 \g__spath_output_tl

```

```

3220 \tl_gclear:N \g__spath_output_tl
3221 }
3222 \cs_generate_variant:Nn \spath_split_path_at_intersections:Nnn
3223 {NVn, NVV, cVn, cVV, cvn, cvv}
3224 \cs_new_protected_nopar:Npn \spath_split_path_at_intersections:Nn #1#2
3225 {
3226 \spath_split_path_at_intersections:NVn #1#1{#2}
3227 }
3228 \cs_generate_variant:Nn \spath_split_path_at_intersections:Nn {cv, NV}
3229 \cs_new_protected_nopar:Npn \spath_gsplit_path_at_intersections:Nnn #1#2#3
3230 {
3231 \__spath_split_path_at_intersections:nn {#2}{#3}
3232 \tl_gset_eq:NN #1 \g__spath_output_tl
3233 \tl_gclear:N \g__spath_output_tl
3234 }
3235 \cs_generate_variant:Nn \spath_gsplit_path_at_intersections:Nnn
3236 {NVn, NVV, cVn, cVV, cvn, cvv}
3237 \cs_new_protected_nopar:Npn \spath_gsplit_path_at_intersections:Nn #1#2
3238 {
3239 \spath_gsplit_path_at_intersections:NVn #1#1{#2}
3240 }
3241 \cs_generate_variant:Nn \spath_gsplit_path_at_intersections:Nn {cv, NV}
3242 \cs_new_protected_nopar:Npn \spath_split_at_intersections:NNnn #1#2#3#4
3243 {
3244 \__spath_split_path_at_intersections:nn {#3}{#4}
3245 \tl_set_eq:NN #1 \g__spath_output_tl
3246 \__spath_split_path_at_intersections:nn {#4}{#3}
3247 \tl_set_eq:NN #2 \g__spath_output_tl
3248 \tl_gclear:N \g__spath_output_tl
3249 }
3250 \cs_generate_variant:Nn \spath_split_at_intersections:NNnn
3251 {NNVn, NNVV, ccVn, ccVV, ccvn, ccvv}
3252 \cs_new_protected_nopar:Npn \spath_split_at_intersections:NN #1#2
3253 {
3254 \spath_split_at_intersections:NNVV #1#2#1#2
3255 }
3256 \cs_generate_variant:Nn \spath_split_at_intersections:NN {cc}
3257 \cs_new_protected_nopar:Npn \spath_gsplit_at_intersections:NNnn #1#2#3#4
3258 {
3259 \__spath_split_path_at_intersections:nn {#3}{#4}
3260 \tl_gset_eq:NN #1 \g__spath_output_tl
3261 \__spath_split_path_at_intersections:nn {#4}{#3}
3262 \tl_gset_eq:NN #2 \g__spath_output_tl
3263 \tl_gclear:N \g__spath_output_tl
3264 }
3265 \cs_generate_variant:Nn \spath_gsplit_at_intersections:NNnn
3266 {NNVn, NNVV, ccVn, ccVV, ccvn, ccvv}
3267 \cs_new_protected_nopar:Npn \spath_gsplit_at_intersections:NN #1#2
3268 {
3269 \spath_gsplit_at_intersections:NNVV #1#2#1#2
3270 }
3271 \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.

```

3272 \cs_new_protected_nopar:Npn \__spath_split_component_at_self_intersections:n #1
3273 {
3274   \group_begin:
3275   \tl_set:Nn \l__spath_tmpe_tl {#1}
3276
3277   % Remember if the component is closed
3278   \spath_finalaction:NV \l__spath_tmpe_tl \l__spath_tmpe_tl
3279
3280   \bool_set:Nn \l__spath_closed_bool
3281   {
3282     \tl_if_eq_p:NN \l__spath_tmpe_tl \c_spath_closepath_tl
3283   }
3284
3285   % Copy the path
3286   \tl_set:Nn \l__spath_tmpe_tl {#1}
3287
3288   % Open the path
3289   \spath_open:N \l__spath_tmpe_tl
3290   % Ensure beziers don't self-intersect
3291   \spath_split_curves:N \l__spath_tmpe_tl
3292
3293   % Make a copy for later
3294   \tl_set_eq:NN \l__spath_tmpg_tl \l__spath_tmpe_tl
3295
3296   % Clear some token lists and sequences
3297   \tl_clear:N \l__spath_tmpd_tl
3298   \seq_clear:N \l__spath_tmpb_seq
3299   \int_zero:N \l__spath_tmpe_int
3300
3301   \pgfintersectionsorthbyfirstpath
3302
3303   % Split the path into a sequence of segments
3304   \spath_segments_to_seq:NV \l__spath_tmpe_tl \l__spath_tmpe_tl
3305
3306   \seq_map_indexed_inline:Nn \l__spath_tmpe_tl
3307   {
3308     \seq_map_indexed_inline:Nn \l__spath_tmpe_tl
3309     {
3310       % Don't intersect a segment with itself
3311       \int_compare:nF
3312       {
3313         ##1 == ####1
3314       }
3315       {
3316         \spath_intersect:nn {##2} {####2}
3317
3318         \int_compare:nT {\pgfintersectionsolutions > 0}
3319         {
3320           % Find the times of the intersections on each path
3321           \int_step_inline:nnnn {1} {1} {\pgfintersectionsolutions}
3322           {
3323             \pgfintersectiongetsolutiontimes
3324             {#####1}{\l__spath_tmpb_tl}{\l__spath_tmpe_tl}

```

```

3325
3326 \bool_if:nT
3327 {
3328   !(
3329     \fp_compare_p:n { \l__spath_tmpb_tl > .99 }
3330     &&
3331     \int_compare_p:n {##1 + 1 == ###1}
3332     )
3333     &&
3334     !(
3335       \fp_compare_p:n { \l__spath_tmpb_tl < .01 }
3336       &&
3337       \int_compare_p:n {##1 - 1 == ###1}
3338       )
3339       &&
3340       !(
3341         \l__spath_closed_bool
3342         &&
3343         \fp_compare_p:n { \l__spath_tmpb_tl < .01 }
3344         &&
3345         \int_compare_p:n {##1 == 1}
3346         &&
3347         \int_compare_p:n {\seq_count:N \l__spath_tmpa_seq == ###1}
3348         )
3349         &&
3350         !(
3351           \l__spath_closed_bool
3352           &&
3353           \fp_compare_p:n { \l__spath_tmpb_tl > .99 }
3354           &&
3355           \int_compare_p:n {####1 == 1}
3356           &&
3357           \int_compare_p:n {\seq_count:N \l__spath_tmpa_seq == ##1}
3358           )
3359       }
3360     {
3361       \tl_set:Nx \l__spath_tmpa_tl
3362       {\fp_to_decimal:n {\l__spath_tmpb_tl + ##1 - 1}}
3363       \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
3364     }
3365   }
3366 }
3367 }
3368 }
3369 }
3370
3371 % Sort the sequence by reverse order along the path
3372 \seq_sort:Nn \l__spath_tmpb_seq
3373 {
3374   \fp_compare:nNnTF { ##1 } < { ##2 }
3375   { \sort_return_swapped: }
3376   { \sort_return_same: }
3377 }
3378

```

```

3379 \seq_get_left:NN \l__spath_tmpb_seq \l__spath_tmpa_tl
3380 \fp_compare:nT
3381 {
3382   \l__spath_tmpa_tl > \seq_count:N \l__spath_tmpa_seq - .01
3383 }
3384 {
3385   \bool_set_false:N \l__spath_closed_bool
3386 }
3387 \seq_get_right:NN \l__spath_tmpb_seq \l__spath_tmpa_tl
3388 \fp_compare:nT
3389 {
3390   \l__spath_tmpa_tl < .01
3391 }
3392 {
3393   \bool_set_false:N \l__spath_closed_bool
3394 }
3395
3396 % Restore the original copy of the path
3397 \tl_set_eq:NN \l__spath_tmpe_tl \l__spath_tmpg_tl
3398
3399 % Clear the token lists
3400 \tl_clear:N \l__spath_tmpf_tl
3401 \tl_clear:N \l__spath_tmph_tl
3402 \tl_clear:N \l__spath_tmpg_tl
3403
3404 \tl_set:Nn \l__spath_tmpe_tl {-1}
3405
3406 \seq_map_inline:Nn \l__spath_tmpb_seq
3407 {
3408   \tl_set:Nn \l__spath_tmpe_tl {##1}
3409   \tl_set_eq:NN \l__spath_tmpe_tl \l__spath_tmpe_tl
3410   \int_compare:nT
3411   {
3412     \fp_to_int:n {floor( \l__spath_tmpe_tl ) }
3413     =
3414     \fp_to_int:n {floor( \l__spath_tmpe_tl ) }
3415   }
3416   {
3417     \tl_set:Nx \l__spath_tmpe_tl
3418     {
3419       \fp_eval:n {
3420         floor( \l__spath_tmpe_tl )
3421         +
3422         ( \l__spath_tmpe_tl - floor( \l__spath_tmpe_tl ) )
3423         /
3424         ( \l__spath_tmpe_tl - floor( \l__spath_tmpe_tl ) )
3425       }
3426     }
3427   }
3428   \tl_set_eq:NN \l__spath_tmpe_tl \l__spath_tmpe_tl
3429
3430   \spath_split_at:NNVV
3431   \l__spath_tmpe_tl
3432   \l__spath_tmpe_tl

```

```

3433     \l__spath_tmpe_tl
3434     \l__spath_tmpe_tl
3435
3436     \tl_put_left:NV \l__spath_tmpe_tl \l__spath_tmpe_tl
3437     \tl_set_eq:NN \l__spath_tmpe_tl \l__spath_tmpe_tl
3438
3439 }
3440
3441 \tl_put_left:NV \l__spath_tmpe_tl \l__spath_tmpe_tl
3442
3443 \tl_if_empty:NT \l__spath_tmpe_tl
3444 {
3445     \tl_set_eq:NN \l__spath_tmpe_tl \l__spath_tmpe_tl
3446 }
3447
3448 \spath_remove_empty_components:N \l__spath_tmpe_tl
3449
3450 % Do something with closed
3451 \bool_if:NT \l__spath_closed_bool
3452 {
3453     \spath_join_component:Nn \l__spath_tmpe_tl {1}
3454 }
3455
3456 \tl_gclear:N \g__spath_output_tl
3457 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpe_tl
3458 \group_end:
3459 }
3460 \cs_new_protected_nopar:Npn \spath_split_component_at_self_intersections:Nn #1#2
3461 {
3462     \__spath_split_component_at_self_intersections:n {#2}
3463     \tl_set_eq:NN #1 \g__spath_output_tl
3464     \tl_gclear:N \g__spath_output_tl
3465 }
3466 \cs_generate_variant:Nn \spath_split_component_at_self_intersections:Nn {NV}
3467 \cs_new_protected_nopar:Npn \spath_split_component_at_self_intersections:N #1
3468 {
3469     \spath_split_component_at_self_intersections:NV #1#1
3470 }
3471 \cs_generate_variant:Nn \spath_split_component_at_self_intersections:N {c}
3472 \cs_new_protected_nopar:Npn \spath_gsplit_component_at_self_intersections:Nn #1#2
3473 {
3474     \__spath_split_component_at_self_intersections:n {#2}
3475     \tl_gset_eq:NN #1 \g__spath_output_tl
3476     \tl_gclear:N \g__spath_output_tl
3477 }
3478 \cs_generate_variant:Nn \spath_gsplit_component_at_self_intersections:Nn {NV}
3479 \cs_new_protected_nopar:Npn \spath_gsplit_component_at_self_intersections:N #1
3480 {
3481     \spath_gsplit_component_at_self_intersections:NV #1#1
3482 }
3483 \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
`\spath_split_at_self_intersections:N`
`\spath_gsplit_at_self_intersections:Nn`
`\spath_gsplit_at_self_intersections:N`

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.

```

3484 \cs_new_protected_nopar:Npn \__spath_split_at_self_intersections:n #1
3485 {
3486   \group_begin:
3487   \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
3488   \seq_clear:N \l__spath_tmpb_seq
3489   \seq_clear:N \l__spath_tmpc_seq
3490
3491   % Iterate over the components of the original path.
3492   \bool_do_until:nn
3493   {
3494     \seq_if_empty_p:N \l__spath_tmpa_seq
3495   }
3496   {
3497     % Get the next component
3498     \seq_pop_left:NN \l__spath_tmpa_seq \l__spath_tmpa_tl
3499     % Copy for later
3500     \tl_set_eq:NN \l__spath_tmpc_tl \l__spath_tmpa_tl
3501     \int_compare:nT
3502     {
3503       \tl_count:N \l__spath_tmpa_tl > 3
3504     }
3505     {
3506
3507       % Split against itself
3508       \spath_split_component_at_self_intersections:N \l__spath_tmpa_tl
3509       % Grab the rest of the path
3510       \tl_set:Nx \l__spath_tmpb_tl
3511       {
3512         \seq_use:Nn \l__spath_tmpb_seq {}
3513         \seq_use:Nn \l__spath_tmpa_seq {}
3514       }
3515       % Split against the rest of the path
3516       \spath_split_path_at_intersections:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
3517     }
3518     % Save the split path
3519     \seq_put_right:NV \l__spath_tmpc_seq \l__spath_tmpa_tl
3520     % Add the original copy to the sequence of processed components
3521     \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpc_tl
3522   }
3523
3524   \tl_gclear:N \g__spath_output_tl
3525   \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpc_seq {} }
3526   \group_end:
3527 }
3528 \cs_generate_variant:Nn \__spath_split_at_self_intersections:n {V, v}
3529 \cs_new_protected_nopar:Npn \spath_split_at_self_intersections:Nn #1#2
3530 {
3531   \__spath_split_at_self_intersections:n {#2}
3532   \tl_set_eq:NN #1 \g__spath_output_tl
3533   \tl_gclear:N \g__spath_output_tl
3534 }
3535 \cs_generate_variant:Nn \spath_split_at_self_intersections:Nn {NV, cn, cV, cv}

```



```

3536 \cs_new_protected_nopar:Npn \spath_split_at_self_intersections:N #1
3537 {
3538   \spath_split_at_self_intersections:NV #1#1
3539 }
3540 \cs_generate_variant:Nn \spath_split_at_self_intersections:N {c}
3541 \cs_new_protected_nopar:Npn \spath_gsplit_at_self_intersections:Nn #1#2
3542 {
3543   \__spath_split_at_self_intersections:n {#2}
3544   \tl_gset_eq:NN #1 \g__spath_output_tl
3545   \tl_gclear:N \g__spath_output_tl
3546 }
3547 \cs_generate_variant:Nn \spath_gsplit_at_self_intersections:Nn {NV, cn, cV, cv}
3548 \cs_new_protected_nopar:Npn \spath_gsplit_at_self_intersections:N #1
3549 {
3550   \spath_gsplit_at_self_intersections:NV #1#1
3551 }
3552 \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
\spath_join_component:Nn
\spath_gjoin_component:Nnn
\spath_gjoin_component:Nn

```

Join the specified component of the spath to its predecessor.

```

3553 \cs_new_protected_nopar:Npn \__spath_join_component:nn #1#2
3554 {
3555   \group_begin:
3556   \spath_numberofcomponents:Nn \l__spath_tmpa_int {#1}
3557
3558   \bool_if:nTF
3559   {
3560     \int_compare_p:n { #2 >= 1 }
3561     &&
3562     \int_compare_p:n { #2 <= \l__spath_tmpa_int }
3563   }
3564   {
3565     \int_compare:nTF
3566     {
3567       #2 == 1
3568     }
3569     {
3570       \int_compare:nTF
3571       {
3572         \l__spath_tmpa_int == 1
3573       }
3574       {
3575         \tl_set:Nn \l__spath_tmpa_tl {#1}
3576         \spath_initialpoint:Nn \l__spath_tmpb_tl {#1}
3577         \tl_put_right:NV \l__spath_tmpa_tl \c_spath_closepath_tl
3578         \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
3579         \tl_gclear:N \g__spath_output_tl
3580         \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
3581       }
3582     }
3583     \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
3584     \seq_pop_left:NN \l__spath_tmpa_seq \l__spath_tmpa_tl
3585

```

```

3586     \prg_replicate:nn {3}
3587     {
3588       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
3589     }
3590
3591     \seq_put_right:NV \l__spath_tmpa_seq \l__spath_tmpa_tl
3592
3593     \tl_gclear:N \g__spath_output_tl
3594     \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpa_seq {}}
3595   }
3596 }
3597 {
3598   \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
3599
3600   \seq_clear:N \l__spath_tmpb_seq
3601   \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
3602   {
3603     \tl_set:Nn \l__spath_tmpa_tl {##2}
3604     \int_compare:nT {##1 = #2}
3605     {
3606       \prg_replicate:nn {3}
3607       {
3608         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
3609       }
3610     }
3611     \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
3612   }
3613
3614   \tl_gclear:N \g__spath_output_tl
3615   \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpb_seq {}}
3616 }
3617 }
3618 {
3619   \tl_gclear:N \g__spath_output_tl
3620   \tl_gset:Nn \g__spath_output_tl {#1}
3621 }
3622
3623 \group_end:
3624 }
3625 \cs_new_protected_nopar:Npn \spath_join_component:Nnn #1#2#3
3626 {
3627   \__spath_join_component:nn {#2}{#3}
3628   \tl_set_eq:NN #1 \g__spath_output_tl
3629   \tl_gclear:N \g__spath_output_tl
3630 }
3631 \cs_generate_variant:Nn \spath_join_component:Nnn {NVn, NVV}
3632 \cs_new_protected_nopar:Npn \spath_join_component:Nn #1#2
3633 {
3634   \spath_join_component:NVn #1#1{#2}
3635 }
3636 \cs_generate_variant:Nn \spath_join_component:Nn {cn, NV, cV}
3637 \cs_new_protected_nopar:Npn \spath_gjoin_component:Nnn #1#2#3
3638 {
3639   \__spath_join_component:nn {#2}{#3}

```

```

3640 \tl_gset_eq:NN #1 \g__spath_output_tl
3641 \tl_gclear:N \g__spath_output_tl
3642 }
3643 \cs_generate_variant:Nn \spath_gjoin_component:Nnn {NVn, NVV}
3644 \cs_new_protected_nopar:Npn \spath_gjoin_component:Nn #1#2
3645 {
3646   \spath_gjoin_component:NVn #1#1{#2}
3647 }
3648 \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
3649 \cs_new_protected_nopar:Npn \__spath_spot_weld_components:n #1
3650 {
3651   \group_begin:
3652   \dim_zero:N \l__spath_move_x_dim
3653   \dim_zero:N \l__spath_move_y_dim
3654
3655   \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
3656   \seq_clear:N \l__spath_tmpb_seq
3657   \dim_set:Nn \l__spath_move_x_dim {\tl_item:nn {#1} {2} + 10 pt}
3658   \dim_set:Nn \l__spath_move_y_dim {\tl_item:nn {#1} {3} + 10 pt}
3659
3660   \int_set:Nn \l__spath_tmpa_int {\seq_count:N \l__spath_tmpa_seq}
3661
3662   \seq_map_inline:Nn \l__spath_tmpa_seq
3663   {
3664     \tl_set:Nn \l__spath_tmpa_tl {##1}
3665     \bool_if:nT
3666     {
3667       \dim_compare_p:n
3668       {
3669         \dim_abs:n
3670         {\l__spath_move_x_dim - \tl_item:Nn \l__spath_tmpa_tl {2} }
3671         < 0.01pt
3672       }
3673       &&
3674       \dim_compare_p:n
3675       {
3676         \dim_abs:n
3677         {\l__spath_move_y_dim - \tl_item:Nn \l__spath_tmpa_tl {3} }
3678         < 0.01pt
3679       }
3680     }
3681     {
3682       \prg_replicate:nn {3}
3683       {
3684         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
3685       }
3686       \int_decr:N \l__spath_tmpa_int
3687     }
3688     \tl_reverse:N \l__spath_tmpa_tl

```

```

3689 \dim_set:Nn \l__spath_move_x_dim {\tl_item:Nn \l__spath_tmpa_tl {2}}
3690 \dim_set:Nn \l__spath_move_y_dim {\tl_item:Nn \l__spath_tmpa_tl {1}}
3691 \tl_reverse:N \l__spath_tmpa_tl
3692 \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
3693 }
3694
3695 \tl_set:Nx \l__spath_tmpa_tl {\seq_use:Nn \l__spath_tmpb_seq {}}
3696 \spath_components_to_seq:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
3697
3698
3699 \spath_initialpoint:Nn \l__spath_tmpa_tl {#1}
3700 \spath_finalpoint:Nn \l__spath_tmpb_tl {#1}
3701
3702 \bool_if:nT
3703 {
3704   \dim_compare_p:n
3705   {
3706     \dim_abs:n
3707     {
3708       \tl_item:Nn \l__spath_tmpa_tl {1} - \tl_item:Nn \l__spath_tmpb_tl {1}
3709     }
3710     <
3711     0.01pt
3712   }
3713   &&
3714   \dim_compare_p:n
3715   {
3716     \dim_abs:n
3717     {
3718       \tl_item:Nn \l__spath_tmpa_tl {2} - \tl_item:Nn \l__spath_tmpb_tl {2}
3719     }
3720     <
3721     0.01pt
3722   }
3723 }
3724 {
3725   \int_compare:nTF
3726   {
3727     \seq_count:N \l__spath_tmpb_seq > 1
3728   }
3729   {
3730     \seq_pop_left:NN \l__spath_tmpb_seq \l__spath_tmpb_tl
3731
3732     \prg_replicate:nn {3}
3733     {
3734       \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
3735     }
3736     \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpb_tl
3737   }
3738   {
3739     \tl_set:NV \l__spath_tmpb_tl \c_spath_closepath_tl
3740     \tl_put_right:Nx \l__spath_tmpb_tl
3741     {
3742       \tl_item:Nn \l__spath_tmpa_tl {1}
3743     }

```

```

3743     { \tl_item:Nn \l__spath_tmpa_tl {2} }
3744   }
3745   \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpb_tl
3746 }
3747 }
3748
3749 \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpb_seq {}}
3750 \group_end:
3751 }
3752 \cs_new_protected_nopar:Npn \spath_spot_weld_components:Nn #1#2
3753 {
3754   \__spath_spot_weld_components:n {#2}
3755   \tl_set_eq:NN #1 \g__spath_output_tl
3756   \tl_gclear:N \g__spath_output_tl
3757 }
3758 \cs_generate_variant:Nn \spath_spot_weld_components:Nn {NV, cV, cn}
3759 \cs_new_protected_nopar:Npn \spath_spot_weld_components:N #1
3760 {
3761   \spath_spot_weld_components:NV #1#1
3762 }
3763 \cs_generate_variant:Nn \spath_spot_weld_components:N {c}
3764 \cs_new_protected_nopar:Npn \spath_spot_gweld_components:Nn #1#2
3765 {
3766   \__spath_spot_weld_components:n {#2}
3767   \tl_gset_eq:NN #1 \g__spath_output_tl
3768   \tl_gclear:N \g__spath_output_tl
3769 }
3770 \cs_generate_variant:Nn \spath_spot_gweld_components:Nn {NV, cV, cn}
3771 \cs_new_protected_nopar:Npn \spath_spot_gweld_components:N #1
3772 {
3773   \spath_spot_gweld_components:NV #1#1
3774 }
3775 \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`
`\spath_gconvert_to_svg:Nn`

Convert the soft path to an SVG document.

```

3776 \cs_new_protected_nopar:Npn \__spath_convert_to_svg:n #1
3777 {
3778   \group_begin:
3779   \tl_clear:N \l__spath_tmpa_tl
3780   \tl_put_right:Nn \l__spath_tmpa_tl
3781   {
3782     <?xml~ version="1.0"~ standalone="no"?>
3783     \iow_newline:
3784     <!DOCTYPE~ svg~ PUBLIC~ "-//W3C//DTD SVG 1.1//EN"~
3785     "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
3786     \iow_newline:
3787     <svg~ xmlns="http://www.w3.org/2000/svg"~ version="1.1"~viewBox="
3788   }
3789
3790   \spath_minbb:Nn \l__spath_tmpb_tl {#1}

```

```

3791 \spath_maxbb:Nn \l__spath_tmpc_tl {#1}
3792 \tl_put_right:Nx \l__spath_tmpa_tl
3793 {
3794   \dim_to_decimal:n
3795   {
3796     \tl_item:Nn \l__spath_tmpb_tl {1} - 10pt
3797   }
3798   \exp_not:n {~}
3799   \dim_to_decimal:n
3800   {
3801     \tl_item:Nn \l__spath_tmpb_tl {2} - 10pt
3802   }
3803   \exp_not:n {~}
3804   \dim_to_decimal:n
3805   {
3806     \tl_item:Nn \l__spath_tmpc_tl {1}
3807     -
3808     \tl_item:Nn \l__spath_tmpb_tl {1}
3809     + 20pt
3810   }
3811   \exp_not:n {~}
3812   \dim_to_decimal:n
3813   {
3814     \tl_item:Nn \l__spath_tmpc_tl {2}
3815     -
3816     \tl_item:Nn \l__spath_tmpb_tl {2}
3817     + 20pt
3818   }
3819 }
3820
3821 \tl_put_right:Nn \l__spath_tmpa_tl
3822 {
3823   ">
3824   \iow_newline:
3825   <path~ d="
3826 }
3827 \tl_set:Nn \l__spath_tmpc_tl {use:n}
3828 \tl_map_inline:nn {#1}
3829 {
3830   \tl_set:Nn \l__spath_tmpb_tl {##1}
3831   \tl_case:NnF \l__spath_tmpb_tl
3832   {
3833     \c_spath_moveto_tl
3834     {
3835       \tl_put_right:Nn \l__spath_tmpa_tl {M~}
3836       \tl_set:Nn \l__spath_tmpc_tl {use:n}
3837     }
3838     \c_spath_lineto_tl
3839     {
3840       \tl_put_right:Nn \l__spath_tmpa_tl {L~}
3841       \tl_set:Nn \l__spath_tmpc_tl {use:n}
3842     }
3843     \c_spath_closepath_tl
3844     {

```

```

3845     \tl_put_right:Nn \l__spath_tmpa_tl {Z~}
3846     \tl_set:Nn \l__spath_tmpc_tl {use:none}
3847   }
3848   \c_spath_curvetoa_tl
3849   {
3850     \tl_put_right:Nn \l__spath_tmpa_tl {C~}
3851     \tl_set:Nn \l__spath_tmpc_tl {use:n}
3852   }
3853   \c_spath_curvetob_tl {
3854     \tl_set:Nn \l__spath_tmpc_tl {use:n}
3855   }
3856   \c_spath_curveto_tl {
3857     \tl_set:Nn \l__spath_tmpc_tl {use:n}
3858   }
3859 }
3860 {
3861   \tl_put_right:Nx
3862   \l__spath_tmpa_tl
3863   {\use:c { \l__spath_tmpc_tl } {\dim_to_decimal:n {##1}} ~}
3864 }
3865 }
3866 \tl_put_right:Nn \l__spath_tmpa_tl
3867 {
3868   "~ fill="none"~ stroke="black"~ />
3869   \iow_newline:
3870   </svg>
3871   \iow_newline:
3872 }
3873 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
3874 \group_end:
3875 }
3876 \cs_new_protected_nopar:Npn \spath_convert_to_svg:Nn #1#2
3877 {
3878   \__spath_convert_to_svg:n {#2}
3879   \tl_set_eq:NN #1 \g__spath_output_tl
3880   \tl_gclear:N \g__spath_output_tl
3881 }
3882 \cs_new_protected_nopar:Npn \spath_gconvert_to_svg:Nn #1#2
3883 {
3884   \__spath_convert_to_svg:n {#2}
3885   \tl_gset_eq:NN #1 \g__spath_output_tl
3886   \tl_gclear:N \g__spath_output_tl
3887 }

```

(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.

```

3888 \iow_new:N \g__spath_stream
3889 \cs_new_protected_nopar:Npn \spath_export_to_svg:nn #1#2
3890 {
3891   \group_begin:
3892   \spath_convert_to_svg:Nn \l__spath_tmpa_tl {#2}
3893   \iow_open:Nn \g__spath_stream {#1 .svg}
3894   \iow_now:Nx \g__spath_stream

```

```

3895 {
3896   \tl_use:N \l__spath_tmpa_tl
3897 }
3898 \iow_close:N \g__spath_stream
3899 \group_end:
3900 }
3901 \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.

```

3902 \cs_new_protected_nopar:Npn \spath_show:n #1
3903 {
3904   \int_step_inline:nnnn {1} {3} {\tl_count:n {#1}}
3905   {
3906     \iow_term:x {
3907       \tl_item:nn {#1} {##1}
3908       {\tl_item:nn {#1} {##1+1}}
3909       {\tl_item:nn {#1} {##1+2}}
3910     }
3911   }
3912 }
3913 \cs_generate_variant:Nn \spath_show:n {V, v}

```

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

3.9 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
3914 \cs_new_protected_nopar:Npn \spath_get_current_path:N #1
3915 {
3916   \pgfsyssoftpath@getcurrentpath #1
3917 }
3918 \cs_new_protected_nopar:Npn \spath_gget_current_path:N #1
3919 {
3920   \pgfsyssoftpath@getcurrentpath #1
3921   \tl_gset_eq:NN #1 #1
3922 }

```

(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.

```

3923 \cs_new_protected_nopar:Npn \spath_protocol_path:n #1
3924 {
3925   \spath_minbb:Nn \l__spath_tmpa_tl {#1}
3926   \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl
3927
3928   \spath_maxbb:Nn \l__spath_tmpa_tl {#1}
3929   \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl
3930 }
3931 \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 3932 \cs_new_protected_nopar:Npn \spath_set_current_path:n #1
3933 {
3934   \spath_protocol_path:n {#1}
3935   \tl_set:Nn \l__spath_tmpa_tl {#1}
3936   \pgfsyssoftpath@setcurrentpath\l__spath_tmpa_tl
3937 }
3938 \cs_new_protected_nopar:Npn \spath_set_current_path:N #1
3939 {
3940   \spath_protocol_path:V #1
3941   \pgfsyssoftpath@setcurrentpath #1
3942 }
3943 \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.

```
3944 \cs_new_protected_nopar:Npn \spath_use_path:nn #1#2
3945 {
3946   \spath_set_current_path:n {#1}
3947   \pgfusepath{#2}
3948 }
```

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

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

```
3949 \cs_new_protected_nopar:Npn \spath_tikz_path:nn #1#2
3950 {
3951   \path[#1] \pgfextra{
3952     \spath_set_current_path:n {#2}
3953     \tl_put_right:Nn \tikz@preactions {\def\tikz@actions@path{#2}}
3954   };
3955 }
3956 \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.

```
3957 \cs_new_protected_nopar:Npn \spath_set_tikz_data:n #1
3958 {
3959   \spath_finalpoint:Nn \l__spath_tmpa_tl {#1}
3960   \tl_set:Nx \l__spath_tmpa_tl
3961   {
3962     \exp_not:c {pgf@x}=\tl_item:Nn \l__spath_tmpa_tl {1}
3963     \exp_not:c {pgf@y}=\tl_item:Nn \l__spath_tmpa_tl {2}
3964   }
3965   \use:c {pgf@process}{%
3966     \tl_use:N \l__spath_tmpa_tl
3967     \pgftransforminvert
3968     \use:c {pgf@pos@transform@glob}
3969   }
3970   \tl_set:Nx \l__spath_tmpa_tl
3971   {
```

```

3972 \exp_not:c {tikz@lastx}=\exp_not:c {pgf@x}
3973 \exp_not:c {tikz@lasty}=\exp_not:c {pgf@y}
3974 \exp_not:c {tikz@lastxsaved}=\exp_not:c {pgf@x}
3975 \exp_not:c {tikz@lastysaved}=\exp_not:c {pgf@y}
3976 }
3977 \tl_use:N \l__spath_tmpa_tl
3978 \spath_finalmovepoint:Nn \l__spath_tmpa_tl {#1}
3979 \ifpgfsyssoftpathmovetorelevant%
3980 \tl_gset_eq:cN {pgfsyssoftpath@lastmoveto} \l__spath_tmpa_tl
3981 \fi
3982 \tl_set:Nx \l__spath_tmpa_tl
3983 {
3984 \exp_not:c {pgf@x}=\tl_item:Nn \l__spath_tmpa_tl {1}
3985 \exp_not:c {pgf@y}=\tl_item:Nn \l__spath_tmpa_tl {2}
3986 }
3987 \use:c {pgf@process}{%
3988 \tl_use:N \l__spath_tmpa_tl
3989 \pgftransforminvert
3990 \use:c {pgf@pos@transform@glob}
3991 }
3992 \tl_set:Nx \l__spath_tmpa_tl
3993 {
3994 \exp_not:c {tikz@lastmovetox}=\exp_not:c {pgf@x}
3995 \exp_not:c {tikz@lastmovetoy}=\exp_not:c {pgf@y}
3996 }
3997 \tl_use:N \l__spath_tmpa_tl
3998 \tl_clear_new:c {tikz@timer}
3999 \tl_set:cn {tikz@timer}
4000 {
4001 \pgftransformreset
4002 \spath_reallength:Nn \l__spath_tmpa_int {#1}
4003 \tl_set_eq:Nc \l__spath_tmpb_tl {tikz@time}
4004 \tl_set:Nx \l__spath_tmpb_tl
4005 { \fp_to_decimal:n {(\l__spath_tmpb_tl) * (\l__spath_tmpa_int)}}
4006 \spath_transformation_at:NnV \l__spath_tmpc_tl {#1} \l__spath_tmpb_tl
4007
4008 \tl_set:Nx \l__spath_tmpa_tl
4009 {
4010 \exp_not:N \pgfpoint
4011 { \tl_item:Nn \l__spath_tmpc_tl {5} }
4012 { \tl_item:Nn \l__spath_tmpc_tl {6} }
4013 }
4014 \exp_args:NV \pgftransformshift \l__spath_tmpa_tl
4015
4016 \ifpgfresetnontranslationatime
4017 \pgftransformresetnontranslations
4018 \fi
4019
4020 \ifpgfslopedatime
4021
4022 \tl_set:Nx \l__spath_tmpa_tl
4023 {
4024 { \tl_item:Nn \l__spath_tmpc_tl {1} }
4025 { \tl_item:Nn \l__spath_tmpc_tl {2} }

```

```

4026     { \tl_item:Nn \l__spath_tmpc_tl {3} }
4027     { \tl_item:Nn \l__spath_tmpc_tl {4} }
4028   }
4029   \ifpgfallowupsidedownattime
4030   \else
4031   \fp_compare:nT { \tl_item:Nn \l__spath_tmpc_tl {4} < 0}
4032   {
4033     \tl_set:Nx \l__spath_tmpa_tl
4034     {
4035       { \fp_eval:n { - (\tl_item:Nn \l__spath_tmpc_tl {1})} }
4036       { \fp_eval:n { - (\tl_item:Nn \l__spath_tmpc_tl {2})} }
4037       { \fp_eval:n { - (\tl_item:Nn \l__spath_tmpc_tl {3})} }
4038       { \fp_eval:n { - (\tl_item:Nn \l__spath_tmpc_tl {4})} }
4039     }
4040   }
4041   \fi
4042   \tl_put_right:Nn \l__spath_tmpa_tl {{\pgfpointorigin}}
4043   \exp_last_unbraced:NV \pgftransformcm \l__spath_tmpa_tl
4044   \fi
4045   }
4046 }
4047 \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.

```

4048 \RequirePackage{spath3}
4049 \RequirePackage{expl3}
4050 \ExplSyntaxOn
4051
4052 \tl_new:N \l__spath_current_tl
4053 \tl_new:N \l__spath_reverse_tl
4054 \tl_new:N \l__spath_prefix_tl
4055 \tl_new:N \l__spath_suffix_tl
4056 \tl_new:N \g__spath_smuggle_tl
4057 \seq_new:N \g__spath_tmpa_seq
4058 \seq_new:N \g__spath_tmpb_seq
4059 \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.

```

4060 \msg_new:nnn { spath3 } { missing soft path } { Soft~ path~ #1~ doesn't~ exist }
4061 \msg_new:nnnn { spath3 } { empty soft path } { Soft~ path~ #1~ is~ empty}
4062 {If~ it~ was~ defined~ inside~ a~ group,~ try~ using~ "save~ global". }
4063 \msg_new:nnn { spath3 } { load intersections }
4064 { You~ need~ to~ load~ the~ "intersections"~ library~ to~ work~ with~ intersections }

```

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.

```

4065 \tl_set:Nn \l__spath_prefix_tl {tikz@intersect@path@name@}
4066 \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.

```

4067 \tl_new:N \g__spath_tikzfinish_tl
4068 \cs_new_protected_nopar:Npn \spath_at_end_of_path:
4069 {
4070   \tl_use:N \g__spath_tikzfinish_tl
4071   \tl_gclear:N \g__spath_tikzfinish_tl
4072 }
4073 \tl_put_right:Nn \tikz@finish {\spath_at_end_of_path:}
4074
4075 \cs_new_protected_nopar:Npn \spath_save_path:Nn #1#2
4076 {
4077   \tl_gput_right:Nn \g__spath_tikzfinish_tl
4078   {
4079     \tl_clear_new:N #1
4080     \tl_set:Nn #1 {#2}
4081   }
4082 }
4083 \cs_generate_variant:Nn \spath_save_path:Nn {cn, NV, cV}
4084
4085 \cs_new_protected_nopar:Npn \spath_gsave_path:Nn #1#2
4086 {
4087   \tl_gput_right:Nn \g__spath_tikzfinish_tl
4088   {
4089     \tl_gclear_new:N #1
4090     \tl_gset:Nn #1 {#2}
4091   }
4092 }
4093 \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.

```

4094 \cs_new_protected_nopar:Npn \__spath_process_tikz_point:Nn #1#2
4095 {
4096   \group_begin:
4097   \use:c {tikz@scan@one@point} \use:n #2 \scan_stop:
4098   \tl_gset:Nx \g__spath_output_tl
4099   {
4100     { \dim_use:c {pgf@x} }
4101     { \dim_use:c {pgf@y} }
4102   }
4103   \group_end:
4104   \tl_set_eq:NN #1 \g__spath_output_tl
4105   \tl_gclear:N \g__spath_output_tl
4106 }

```

(End definition for __spath_process_tikz_point:Nn.)

`__spath_tikzset:n` Wrapper around `\tikzset` for expansion.

```

4107 \cs_set_eq:NN \__spath_tikzset:n \tikzset
4108 \cs_generate_variant:Nn \__spath_tikzset:n {V, v}

```

(End definition for `_spath_tikzset:n`.)

When joining two paths we provide a set of options for how to process the second path.

```
4109 \bool_new:N \l__spath_reverse_bool
4110 \bool_new:N \l__spath_weld_bool
4111 \bool_new:N \l__spath_move_bool
4112 \bool_new:N \l__spath_global_bool
4113 \tl_new:N \l__spath_joinpath_tl
4114 \tl_new:N \l__spath_transformation_tl
4115
4116 \cs_new_protected_nopar:Npn \__spath_set_bool:Nn #1#2
4117 {
4118   \tl_if_eq:nnTF {#2}{false}
4119   {
4120     \bool_set_false:N #1
4121   }
4122   {
4123     \bool_set_true:N #1
4124   }
4125 }
4126 \tikzset {
4127   spath/join/.is~ family,
4128   spath/join/.cd,
4129   reverse/.code = {
4130     \__spath_set_bool:Nn \l__spath_reverse_bool {#1}
4131   },
4132   reverse/.default = true,
4133   weld/.code = {
4134     \__spath_set_bool:Nn \l__spath_weld_bool {#1}
4135   },
4136   weld/.default = true,
4137   no~ weld/.code = {
4138     \__spath_set_bool:Nn \l__spath_weld_bool {#1}
4139     \bool_set:Nn \l__spath_weld_bool {! \l__spath_weld_bool}
4140   },
4141   no~ weld/.default = true,
4142   move/.code = {
4143     \__spath_set_bool:Nn \l__spath_move_bool {#1}
4144   },
4145   move/.default = true,
4146   no~ move/.code = {
4147     \__spath_set_bool:Nn \l__spath_move_bool {#1}
4148     \bool_set:Nn \l__spath_move_bool {! \l__spath_move_bool}
4149   },
4150   no~ move/.default = true,
4151   global/.code = {
4152     \__spath_set_bool:Nn \l__spath_global_bool {#1}
4153   },
4154   global/.default = true,
4155   transform/.store-in=\l__spath_transformation_tl,
4156   .unknown/.code = {
4157     \tl_set_eq:NN \l__spath_joinpath_tl \pgfkeyscurrentname
4158   }
4159 }
```

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).

```
4160 \cs_set_eq:NN \GetComponentOf \clist_item:Nn
```

Now we define all of our keys.

```
4161 \tikzset{
```

We're in the `spath` key family.

```
4162   spath/.is~family,
```

```
4163   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.

```
4164   set~ prefix/.store~ in=\l__spath_prefix_tl,
4165   prefix/.is~choice,
4166   prefix/default/.style={
4167     /tikz/spath/set~ prefix=tikz@intersect@path@name@
4168   },
4169   set~ suffix/.store~ in=\l__spath_suffix_tl,
4170   suffix/.is~choice,
4171   suffix/default/.style={
4172     /tikz/spath/set~ suffix={}
4173   },
4174   set~ name/.style={
4175     /tikz/spath/prefix=#1,
4176     /tikz/spath/suffix=#1
4177   },
```

Keys for saving and cloning a soft path.

```
4178   save/.code={
4179     \tikz@addmode{
4180       \spath_get_current_path:N \l__spath_tmpa_tl
4181       \spath_bake_round:NV \l__spath_tmpa_tl \l__spath_tmpa_tl
4182       \spath_save_path:cV
4183       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4184       \l__spath_tmpa_tl
4185     }
4186   },
4187   save~ global/.code={
4188     \tikz@addmode{
4189       \spath_get_current_path:N \l__spath_tmpa_tl
4190       \spath_bake_round:NV \l__spath_tmpa_tl \l__spath_tmpa_tl
4191       \spath_gsave_path:cV
4192       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4193       \l__spath_tmpa_tl
4194     }
4195   },
4196   clone/.code~ 2~ args={
4197     \tl_if_exist:cTF
4198     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4199     {
```

```

4200     \tl_clear_new:c
4201     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4202     \tl_set_eq:cc
4203     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4204     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4205   }
4206   {
4207     \msg_warning:nnn { spath3 } { missing soft path } { #2 }
4208   }
4209 },
4210 clone~ global/.code~ 2~ args={
4211   \tl_if_exist:cTF
4212   {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4213   {
4214     \tl_gclear_new:c
4215     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4216     \tl_gset_eq:cc
4217     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4218     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4219   }
4220   {
4221     \msg_warning:nnn { spath3 } { missing soft path } { #2 }
4222   }
4223 },

```

Saves a soft path to the aux file.

```

4224 save~ to~ aux/.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_save_to_aux: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 },

```

Exports the path as an SVG file.

```

4235 export~ to~ svg/.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_export_to_svg:nv {#1}
4240     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4241   }
4242   {
4243     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4244   }
4245 },

```

Inserts the named path at the current point in the path, with options for how this is accomplished. The inserted path can be transformed, reversed, moved to the current

point, and welded to the current path. If this is used before the path has been started then it becomes the start of the path (and the “current point” is taken as the origin).

```

4246 use/.code={
4247   \bool_set_false:N \l__spath_reverse_bool
4248   \bool_set_false:N \l__spath_weld_bool
4249   \bool_set_false:N \l__spath_move_bool
4250   \tl_clear:N \l__spath_joinpath_tl
4251   \tl_clear:N \l__spath_transformation_tl
4252   \tikzset{
4253     spath/join/.cd,
4254     #1
4255   }
4256
4257   \tl_if_exist:cTF
4258   {
4259     \tl_use:N \l__spath_prefix_tl
4260     \tl_use:N \l__spath_joinpath_tl
4261     \tl_use:N \l__spath_suffix_tl
4262   }
4263   {
4264     \tl_if_empty:cT
4265     {
4266       \tl_use:N \l__spath_prefix_tl
4267       \tl_use:N \l__spath_joinpath_tl
4268       \tl_use:N \l__spath_suffix_tl
4269     }
4270     {
4271       \msg_warning:nnn { spath3 } { empty soft path } { #1 }
4272     }
4273     \tl_set_eq:Nc \l__spath_joinpath_tl
4274     {
4275       \tl_use:N \l__spath_prefix_tl
4276       \tl_use:N \l__spath_joinpath_tl
4277       \tl_use:N \l__spath_suffix_tl
4278     }
4279     \spath_get_current_path:N \l__spath_current_tl
4280
4281     \bool_if:NT \l__spath_reverse_bool
4282     {
4283       \spath_reverse:N \l__spath_joinpath_tl
4284     }
4285
4286     \tl_if_empty:NF \l__spath_transformation_tl
4287     {
4288       \group_begin:
4289       \pgftransformreset
4290       \__spath_tikzset:V \l__spath_transformation_tl
4291       \pgfgettransform \l__spath_tmpa_tl
4292       \tl_gset:Nn \g__spath_smuggle_tl
4293       {
4294         \spath_transform:Nnnnnn
4295         \l__spath_joinpath_tl
4296       }
4297       \tl_gput_right:NV \g__spath_smuggle_tl \l__spath_tmpa_tl

```



```

4298     \group_end:
4299     \tl_use:N \g__spath_smuggle_tl
4300 }
4301
4302 \bool_if:NT \l__spath_move_bool
4303 {
4304     \tl_if_empty:NTF \l__spath_current_tl
4305     {
4306         \tl_set:Nn \l__spath_tmpc_tl { {Opt} {Opt} }
4307     }
4308     {
4309         \spath_finalpoint:NV
4310         \l__spath_tmpc_tl
4311         \l__spath_current_tl
4312     }
4313     \spath_translate_to:NV \l__spath_joinpath_tl \l__spath_tmpc_tl
4314 }
4315
4316 \tl_if_empty:NTF \l__spath_current_tl
4317 {
4318     \tl_if_empty:NTF \l__spath_joinpath_tl
4319     {
4320         \tl_set_eq:NN \l__spath_current_tl \c_spath_moveto_tl
4321         \tl_put_right:Nn \l__spath_current_tl {{0pt}{0pt}}
4322     }
4323     {
4324         \tl_set_eq:NN \l__spath_current_tl \l__spath_joinpath_tl
4325     }
4326 }
4327 {
4328
4329     \tl_clear:N \l__spath_tmpa_tl
4330     \tl_set:Nn \l__spath_tmpa_tl {spath_}
4331
4332     \tl_put_right:Nn \l__spath_tmpa_tl {append}
4333
4334     \bool_if:NT \l__spath_weld_bool
4335     {
4336         \tl_put_right:Nn \l__spath_tmpa_tl {_no_move}
4337     }
4338     \tl_put_right:Nn \l__spath_tmpa_tl {:NV}
4339
4340     \use:c {\tl_use:N \l__spath_tmpa_tl }
4341     \l__spath_current_tl
4342     \l__spath_joinpath_tl
4343 }
4344
4345 \spath_set_current_path:N \l__spath_current_tl
4346 \spath_set_tikz_data:V \l__spath_joinpath_tl
4347 }
4348 {
4349     \msg_warning:nnx
4350     { spath3 }
4351     { missing soft path }

```

```

4352     {\tl_use:N \l__spath_joinpath_tl }
4353   }
4354 },

```

Some aliases for the above.

```

4355 restore/.style={/tikz/spath/use={#1}},
4356 restore~ reverse/.style={/tikz/spath/use={reverse, #1}},
4357 append/.style={/tikz/spath/use={move, weld, #1}},
4358 append~ no~ move/.style={/tikz/spath/use={weld, #1}},
4359 append~ reverse/.style={/tikz/spath/use={move, weld, reverse, #1}},
4360 append~ reverse~ no~ move/.style={/tikz/spath/use={weld, reverse, #1}},
4361 insert/.style={/tikz/spath/use={#1}},
4362 insert~ reverse/.style={/tikz/spath/use={reverse, #1}},

```

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

```

4363 show~current~path/.code={
4364   \tikz@addmode{
4365     \pgfsyssoftpath@getcurrentpath\l__spath_tmpa_tl
4366     \iow_term:n {---- current~ soft~ path~ ---}
4367     \spath_show:V \l__spath_tmpa_tl
4368   }
4369 },

```

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

```

4370 show/.code={
4371   \tl_if_exist:cTF
4372   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4373   {
4374     \tl_if_empty:cTF
4375     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4376     {
4377       \msg_warning:nnn { spath3 } { empty soft path } { #1 }
4378     }
4379     {
4380       \iow_term:n {---- soft~ path~ #1~ ---}
4381       \spath_show:v
4382       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4383     }
4384   }
4385   {
4386     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4387   }
4388 },

```

This joins a path on to an existing path, possibly modifying it first. The possible options are the same as those for use. It is possible to specify the same path both for the initial and the joining path as a copy is made internally first.

```

4389 join~ with/.code~ 2~ args={
4390   \tl_if_exist:cTF
4391   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4392   {
4393     \bool_set_false:N \l__spath_reverse_bool
4394     \bool_set_false:N \l__spath_weld_bool
4395     \bool_set_false:N \l__spath_move_bool

```

```

4396 \bool_set_false:N \l__spath_global_bool
4397 \tl_clear:N \l__spath_joinpath_tl
4398 \tl_clear:N \l__spath_transformation_tl
4399 \tikzset{
4400   spath/join/.cd,
4401   #2
4402 }
4403
4404 \tl_if_exist:cTF
4405 {
4406   \tl_use:N \l__spath_prefix_tl
4407   \tl_use:N \l__spath_joinpath_tl
4408   \tl_use:N \l__spath_suffix_tl
4409 }
4410 {
4411   \tl_set_eq:Nc \l__spath_joinpath_tl
4412   {
4413     \tl_use:N \l__spath_prefix_tl
4414     \tl_use:N \l__spath_joinpath_tl
4415     \tl_use:N \l__spath_suffix_tl
4416   }
4417
4418   \bool_if:NT \l__spath_reverse_bool
4419   {
4420     \spath_reverse:N \l__spath_joinpath_tl
4421   }
4422
4423   \tl_if_empty:NF \l__spath_transformation_tl
4424   {
4425     \group_begin:
4426     \pgftransformreset
4427     \__spath_tikzset:V \l__spath_transformation_tl
4428     \pgfgettransform \l__spath_tmpa_tl
4429     \tl_gset:Nn \g__spath_smuggle_tl
4430     {
4431       \spath_transform:Nnnnnnn
4432       \l__spath_joinpath_tl
4433     }
4434     \tl_gput_right:NV \g__spath_smuggle_tl \l__spath_tmpa_tl
4435     \group_end:
4436     \tl_use:N \g__spath_smuggle_tl
4437   }
4438
4439   \bool_if:NT \l__spath_move_bool
4440   {
4441     \spath_finalpoint:Nv
4442     \l__spath_tmpc_tl
4443     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4444     \spath_translate_to:NV \l__spath_joinpath_tl \l__spath_tmpc_tl
4445   }
4446
4447   \tl_clear:N \l__spath_tmpa_tl
4448   \tl_set:Nn \l__spath_tmpa_tl {spath_}
4449

```

```

4450     \bool_if:NT \l__spath_global_bool
4451     {
4452       \tl_put_right:Nn \l__spath_tmpa_tl {g}
4453     }
4454
4455     \tl_put_right:Nn \l__spath_tmpa_tl {append}
4456
4457     \bool_if:NT \l__spath_weld_bool
4458     {
4459       \tl_put_right:Nn \l__spath_tmpa_tl {_no_move}
4460     }
4461     \tl_put_right:Nn \l__spath_tmpa_tl {:cV}
4462
4463     \cs_if_exist:cF {\tl_use:N \l__spath_tmpa_tl}
4464     {
4465       \tl_show:N \l__spath_tmpa_tl
4466     }
4467
4468     \use:c {\tl_use:N \l__spath_tmpa_tl }
4469     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4470     \l__spath_joinpath_tl
4471   }
4472   {
4473     \msg_warning:nxx
4474     { spath3 }
4475     { missing soft path }
4476     {\tl_use:N \l__spath_joinpath_tl }
4477   }
4478 }
4479 {
4480   \msg_warning:nxx { spath3 } { missing soft path } { #1 }
4481 }
4482 },

```

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

```

4483 spot~ weld/.code={
4484   \tl_if_exist:cTF
4485   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4486   {
4487     \spath_spot_weld_components:c
4488     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4489   }
4490   {
4491     \msg_warning:nxx { spath3 } { missing soft path } { #1 }
4492   }
4493 },
4494 spot~ weld~ globally/.code={
4495   \tl_if_exist:cTF
4496   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4497   {
4498     \spath_spot_gweld_components:c
4499     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4500   }

```

```

4501     {
4502     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4503     }
4504 },

```

Reverses the named path.

```

4505 reverse/.code={
4506   \tl_if_exist:cTF
4507   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4508   {
4509     \spath_reverse:c
4510     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4511   }
4512   {
4513     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4514   }
4515 },
4516 reverse~ globally/.code={
4517   \tl_if_exist:cTF
4518   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4519   {
4520     \spath_reverse:c
4521     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4522   }
4523   {
4524     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4525   }
4526 },

```

Adjust a path to span between two points.

```

4527 span/.code ~n~ args={3}{
4528   \tl_if_exist:cTF
4529   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4530   {
4531     \__spath_process_tikz_point:Nn \l__spath_tmpa_tl {#2}
4532     \__spath_process_tikz_point:Nn \l__spath_tmpb_tl {#3}
4533     \spath_span:cVV
4534     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4535     \l__spath_tmpa_tl \l__spath_tmpb_tl
4536   }
4537   {
4538     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4539   }
4540 },
4541 span~ global/.code ~n~ args={3}{
4542   \tl_if_exist:cTF
4543   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4544   {
4545     \__spath_process_tikz_point:Nn \l__spath_tmpa_tl {#2}
4546     \__spath_process_tikz_point:Nn \l__spath_tmpb_tl {#3}
4547     \spath_gspan:cVV
4548     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4549     \l__spath_tmpa_tl \l__spath_tmpb_tl
4550   }

```

```

4551     {
4552     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4553     }
4554 },

```

Defines a to path

```

4555 to/.style={
4556   to~path={
4557     [
4558       spath/span={#1}{(\tikztostart)}{(\tikztotarget)},
4559       spath/append~no~move={#1},
4560     ]
4561     \tikztonodes
4562   }
4563 },

```

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

```

4564 splice/.code ~n~ args={3}{
4565   \tl_if_exist:cTF
4566   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4567   {
4568     \tl_if_exist:cTF
4569     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4570     {
4571       \tl_if_exist:cTF
4572       {\tl_use:N \l__spath_prefix_tl #3 \tl_use:N \l__spath_suffix_tl}
4573       {
4574         \spath_splice_between:cvv
4575         {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4576         {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4577         {\tl_use:N \l__spath_prefix_tl #3 \tl_use:N \l__spath_suffix_tl}
4578       }
4579     }
4580     \msg_warning:nnn { spath3 } { missing soft path } { #3 }
4581   }
4582   {
4583     \msg_warning:nnn { spath3 } { missing soft path } { #2 }
4584   }
4585 }
4586 }
4587 {
4588   \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4589 }
4590 },
4591 splice~ global/.code ~n~ args={3}{
4592   \tl_if_exist:cTF
4593   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4594   {
4595     \tl_if_exist:cTF
4596     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4597     {
4598       \tl_if_exist:cTF
4599       {\tl_use:N \l__spath_prefix_tl #3 \tl_use:N \l__spath_suffix_tl}

```

```

4600     {
4601       \spath_gsplice_between:cvv
4602       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4603       {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
4604       {\tl_use:N \l__spath_prefix_tl #3 \tl_use:N \l__spath_suffix_tl}
4605     }
4606     {
4607       \msg_warning:nnn { spath3 } { missing soft path } { #3 }
4608     }
4609   }
4610   {
4611     \msg_warning:nnn { spath3 } { missing soft path } { #2 }
4612   }
4613 }
4614 {
4615   \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4616 }
4617 },

```

Join the components of a path by splicing in the second path whenever the components are sufficiently far apart. The third argument is a list of components to splice after, if it is empty then all components are used and a spot weld is done first so that the splicing only happens if there is an actual gap.

The upright versions will join with the reflection of the splice path if it detects that the gap is “upside-down”.

```

4618 join~ components~ with/.code~2~args={
4619   \tl_if_exist:cTF
4620   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4621   {
4622     \tl_if_head_is_group:nTF {#2}
4623     {
4624       \tl_set:Nx \l__spath_tmpc_tl { \tl_item:nn {#2} {1} }
4625       \tl_set:Nx \l__spath_tmpd_tl { \tl_item:nn {#2} {2} }
4626     }
4627     {
4628       \tl_set:Nn \l__spath_tmpc_tl {#2}
4629       \tl_clear:N \l__spath_tmpd_tl
4630     }
4631   }
4632   \tl_if_exist:cTF
4633   {
4634     \tl_use:N \l__spath_prefix_tl
4635     \tl_use:N \l__spath_tmpc_tl
4636     \tl_use:N \l__spath_suffix_tl
4637   }
4638   {
4639     \tl_if_empty:NT \l__spath_tmpd_tl
4640     {
4641       \spath_spot_weld_components:c
4642       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4643     }
4644     \spath_components_to_seq:Nv
4645     \l__spath_tmpa_seq
4646     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}

```

```

4647
4648 \seq_gclear:N \g__spath_tmpa_seq
4649
4650 \tl_if_empty:NTF \l__spath_tmpd_tl
4651 {
4652   \int_step_inline:nnnn {1}{1} {\seq_count:N \l__spath_tmpa_seq}
4653   {
4654     \seq_gput_right:Nn \g__spath_tmpa_seq {##1}
4655   }
4656 }
4657 {
4658   \foreach \l__spath_tmpa_tl in \l__spath_tmpd_tl
4659   {
4660     \seq_gput_right:NV \g__spath_tmpa_seq \l__spath_tmpa_tl
4661   }
4662   \seq_gsort:Nn \g__spath_tmpa_seq
4663   {
4664     \int_compare:nNnTF {##1} < {##2}
4665     { \sort_return_same: }
4666     { \sort_return_swapped: }
4667   }
4668 }
4669
4670 \seq_pop_left:NN \l__spath_tmpa_seq \l__spath_tmpa_tl
4671 \seq_gpop_left:NN \g__spath_tmpa_seq \l__spath_tmpb_tl
4672
4673 \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
4674 {
4675   \int_compare:nTF
4676   {
4677     ##1 == \l__spath_tmpb_tl
4678   }
4679   {
4680     \seq_gpop_left:NNF \g__spath_tmpa_seq \l__spath_tmpb_tl
4681     {
4682       \tl_set:Nn \l__spath_tmpb_tl {-1}
4683     }
4684     \spath_splice_between:Nvn \l__spath_tmpa_tl
4685     {
4686       \tl_use:N \l__spath_prefix_tl
4687       \tl_use:N \l__spath_tmpc_tl
4688       \tl_use:N \l__spath_suffix_tl
4689     }
4690     {##2}
4691   }
4692   {
4693     \tl_put_right:Nn \l__spath_tmpa_tl {##2}
4694   }
4695 }
4696 \tl_set_eq:cN
4697 {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4698 \l__spath_tmpa_tl
4699 }
4700 {

```



```

4701     \msg_warning:nxx
4702     { spath3 }
4703     { missing soft path }
4704     { \tl_use:N \l__spath_tmpc_tl }
4705   }
4706 }
4707 {
4708   \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4709 }
4710 },
4711 join~ components~ globally~ with/.code~2-args={
4712   \tl_if_exist:cTF
4713   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4714   {
4715     \tl_if_head_is_group:nTF {#2}
4716     {
4717       \tl_set:Nx \l__spath_tmpc_tl { \tl_item:nn {#2} {1} }
4718       \tl_set:Nx \l__spath_tmpd_tl { \tl_item:nn {#2} {2} }
4719     }
4720     {
4721       \tl_set:Nn \l__spath_tmpc_tl {#2}
4722       \tl_clear:N \l__spath_tmpd_tl
4723     }
4724     \tl_if_exist:cTF
4725     {
4726       \tl_use:N \l__spath_prefix_tl
4727       \tl_use:N \l__spath_tmpc_tl
4728       \tl_use:N \l__spath_suffix_tl
4729     }
4730     {
4731       \tl_if_empty:NT \l__spath_tmpd_tl
4732       {
4733         \spath_spot_weld_components:c
4734         {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4735       }
4736     }
4737     \spath_components_to_seq:Nv
4738     \l__spath_tmpa_seq
4739     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4740
4741     \seq_gclear:N \g__spath_tmpa_seq
4742
4743     \tl_if_empty:NTF \l__spath_tmpd_tl
4744     {
4745       \int_step_inline:nnnn {1}{1} {\seq_count:N \l__spath_tmpa_seq}
4746       {
4747         \seq_gput_right:Nn \g__spath_tmpa_seq {##1}
4748       }
4749     }
4750     {
4751       \foreach \l__spath_tmpa_tl in \l__spath_tmpd_tl
4752       {
4753         \seq_gput_right:NV \g__spath_tmpa_seq \l__spath_tmpa_tl
4754       }

```

```

4755     \seq_gsort:Nn \g__spath_tmpa_seq
4756     {
4757         \int_compare:nNnTF {##1} < {##2}
4758         { \sort_return_same: }
4759         { \sort_return_swapped: }
4760     }
4761 }
4762
4763 \seq_pop_left:NN \l__spath_tmpa_seq \l__spath_tmpa_tl
4764 \seq_gpop_left:NN \g__spath_tmpa_seq \l__spath_tmpb_tl
4765
4766 \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
4767 {
4768     \int_compare:nTF
4769     {
4770         ##1 == \l__spath_tmpb_tl
4771     }
4772     {
4773         \seq_gpop_left:NnF \g__spath_tmpa_seq \l__spath_tmpb_tl
4774         {
4775             \tl_set:Nn \l__spath_tmpb_tl {-1}
4776         }
4777         \spath_splice_between:Nvn \l__spath_tmpa_tl
4778         {
4779             \tl_use:N \l__spath_prefix_tl
4780             \tl_use:N \l__spath_tmpc_tl
4781             \tl_use:N \l__spath_suffix_tl
4782         }
4783         {##2}
4784     }
4785     {
4786         \tl_put_right:Nn \l__spath_tmpa_tl {##2}
4787     }
4788 }
4789 \tl_gset_eq:cN
4790 {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4791 \l__spath_tmpa_tl
4792 }
4793 {
4794     \msg_warning:nxx
4795     { spath3 }
4796     { missing soft path }
4797     { \tl_use:N \l__spath_tmpc_tl }
4798 }
4799 }
4800 {
4801     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4802 }
4803 },
4804 join~ components~ upright~ with/.code~2~args={
4805     \tl_if_exist:cTF
4806     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4807     {
4808         \tl_if_head_is_group:nTF {#2}

```

```

4809 {
4810 \tl_set:Nx \l__spath_tmpc_tl { \tl_item:nn {#2} {1} }
4811 \tl_set:Nx \l__spath_tmpd_tl { \tl_item:nn {#2} {2} }
4812 }
4813 {
4814 \tl_set:Nn \l__spath_tmpc_tl {#2}
4815 \tl_clear:N \l__spath_tmpd_tl
4816 }
4817 \tl_if_exist:cTF
4818 {
4819 \tl_use:N \l__spath_prefix_tl
4820 \tl_use:N \l__spath_tmpc_tl
4821 \tl_use:N \l__spath_suffix_tl
4822 }
4823 {
4824 \tl_if_empty:NT \l__spath_tmpd_tl
4825 {
4826 \spath_spot_weld_components:c
4827 {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4828 }
4829
4830 \spath_components_to_seq:Nv
4831 \l__spath_tmpa_seq
4832 {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4833
4834 \seq_gclear:N \g__spath_tmpa_seq
4835
4836 \tl_if_empty:NTF \l__spath_tmpd_tl
4837 {
4838 \int_step_inline:nnnn {1}{1} {\seq_count:N \l__spath_tmpa_seq}
4839 {
4840 \seq_gput_right:Nn \g__spath_tmpa_seq {##1}
4841 }
4842 }
4843 {
4844 \foreach \l__spath_tmpa_tl in \l__spath_tmpd_tl
4845 {
4846 \seq_gput_right:Nv \g__spath_tmpa_seq \l__spath_tmpa_tl
4847 }
4848 \seq_gsort:Nn \g__spath_tmpa_seq
4849 {
4850 \int_compare:nNnTF {##1} < {##2}
4851 { \sort_return_same: }
4852 { \sort_return_swapped: }
4853 }
4854 }
4855
4856 \seq_pop_left:NN \l__spath_tmpa_seq \l__spath_tmpa_tl
4857 \seq_gpop_left:NN \g__spath_tmpa_seq \l__spath_tmpb_tl
4858
4859 \tl_set_eq:Nc \l__spath_tmpc_tl
4860 {
4861 \tl_use:N \l__spath_prefix_tl
4862 \tl_use:N \l__spath_tmpc_tl

```

```

4863         \tl_use:N \l__spath_suffix_tl
4864     }
4865     \spath_transform:NVnnnnnn \l__spath_tmpd_tl \l__spath_tmpe_tl {1}{0}{0}{-
1}{Opt}{Opt}
4866
4867     \seq_map_indexed_inline:Nn \l__spath_tmpe_seq
4868     {
4869         \int_compare:nTF
4870         {
4871             ##1 == \l__spath_tmpe_tl
4872         }
4873         {
4874             \seq_gpop_left:NNF \g__spath_tmpe_seq \l__spath_tmpe_tl
4875             {
4876                 \tl_set:Nn \l__spath_tmpe_tl {-1}
4877             }
4878             \spath_finalpoint:NV \l__spath_tmpe_tl \l__spath_tmpe_tl
4879             \spath_initialpoint:Nn \l__spath_tmpe_tl {##2}
4880
4881             \dim_compare:nTF
4882             {
4883                 \tl_item:Nn \l__spath_tmpe_tl {1}
4884                 >
4885                 \tl_item:Nn \l__spath_tmpe_tl {1}
4886             }
4887             {
4888                 \spath_splice_between:NVn
4889                 \l__spath_tmpe_tl
4890                 \l__spath_tmpe_tl
4891                 {##2}
4892             }
4893             {
4894                 \spath_splice_between:NVn
4895                 \l__spath_tmpe_tl
4896                 \l__spath_tmpe_tl
4897                 {##2}
4898             }
4899         }
4900         {
4901             \tl_put_right:Nn \l__spath_tmpe_tl {##2}
4902         }
4903     }
4904     \tl_set_eq:cN
4905     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4906     \l__spath_tmpe_tl
4907 }
4908 {
4909     \msg_warning:nxx
4910     { spath3 }
4911     { missing soft path }
4912     { \tl_use:N \l__spath_tmpe_tl }
4913 }
4914 }
4915 {

```

```

4916     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
4917   }
4918 },
4919 join~ components~ globally~ upright~ with/.code~2~args={
4920   \tl_if_exist:cTF
4921   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4922   {
4923     \tl_if_head_is_group:nTF {#2}
4924     {
4925       \tl_set:Nx \l__spath_tmpc_tl { \tl_item:nn {#2} {1} }
4926       \tl_set:Nx \l__spath_tmpd_tl { \tl_item:nn {#2} {2} }
4927     }
4928     {
4929       \tl_set:Nn \l__spath_tmpc_tl {#2}
4930       \tl_clear:N \l__spath_tmpd_tl
4931     }
4932   \tl_if_exist:cTF
4933   {
4934     \tl_use:N \l__spath_prefix_tl
4935     \tl_use:N \l__spath_tmpc_tl
4936     \tl_use:N \l__spath_suffix_tl
4937   }
4938   {
4939     \tl_if_empty:NT \l__spath_tmpd_tl
4940     {
4941       \spath_spot_weld_components:c
4942       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4943     }
4944
4945     \spath_components_to_seq:Nv
4946     \l__spath_tmpa_seq
4947     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
4948
4949     \seq_gclear:N \g__spath_tmpa_seq
4950
4951     \tl_if_empty:NTF \l__spath_tmpd_tl
4952     {
4953       \int_step_inline:nnnn {1}{1} {\seq_count:N \l__spath_tmpa_seq}
4954       {
4955         \seq_gput_right:Nn \g__spath_tmpa_seq {##1}
4956       }
4957     }
4958     {
4959       \foreach \l__spath_tmpa_tl in \l__spath_tmpd_tl
4960       {
4961         \seq_gput_right:NV \g__spath_tmpa_seq \l__spath_tmpa_tl
4962       }
4963       \seq_gsort:Nn \g__spath_tmpa_seq
4964       {
4965         \int_compare:nNnTF {##1} < {##2}
4966         { \sort_return_same: }
4967         { \sort_return_swapped: }
4968       }
4969     }

```

```

4970
4971 \seq_pop_left:NN \l__spath_tmpa_seq \l__spath_tmpa_tl
4972 \seq_gpop_left:NN \g__spath_tmpa_seq \l__spath_tmpb_tl
4973
4974 \tl_set_eq:Nc \l__spath_tmpc_tl
4975 {
4976   \tl_use:N \l__spath_prefix_tl
4977   \tl_use:N \l__spath_tmpc_tl
4978   \tl_use:N \l__spath_suffix_tl
4979 }
4980 \spath_transform:NVnnnnnn \l__spath_tmpd_tl \l__spath_tmpc_tl {1}{0}{0}{-
1}{0pt}{0pt}
4981
4982 \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
4983 {
4984   \int_compare:nTF
4985   {
4986     ##1 == \l__spath_tmpb_tl
4987   }
4988   {
4989     \seq_gpop_left:NNF \g__spath_tmpa_seq \l__spath_tmpb_tl
4990     {
4991       \tl_set:Nn \l__spath_tmpb_tl {-1}
4992     }
4993     \spath_finalpoint:NV \l__spath_tmpe_tl \l__spath_tmpa_tl
4994     \spath_initialpoint:Nn \l__spath_tmpf_tl {##2}
4995
4996     \dim_compare:nTF
4997     {
4998       \tl_item:Nn \l__spath_tmpe_tl {1}
4999       >
5000       \tl_item:Nn \l__spath_tmpf_tl {1}
5001     }
5002     {
5003       \spath_splice_between:NVn
5004       \l__spath_tmpa_tl
5005       \l__spath_tmpd_tl
5006       {##2}
5007     }
5008     {
5009       \spath_splice_between:NVn
5010       \l__spath_tmpa_tl
5011       \l__spath_tmpc_tl
5012       {##2}
5013     }
5014   }
5015   {
5016     \tl_put_right:Nn \l__spath_tmpa_tl {##2}
5017   }
5018 }
5019 \tl_gset_eq:cN
5020 {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5021 \l__spath_tmpa_tl
5022 }

```

```

5023     {
5024       \msg_warning:nxx
5025       { spath3 }
5026       { missing soft path }
5027       { \tl_use:N \l__spath_tmpc_tl }
5028     }
5029   }
5030   {
5031     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5032   }
5033 },

```

Close a path.

```

5034 close/.code={
5035   \tl_if_exist:cTF
5036   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5037   {
5038     \spath_close:c
5039     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5040   }
5041   {
5042     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5043   }
5044 },
5045 close~ globally/.code={
5046   \tl_if_exist:cTF
5047   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5048   {
5049     \spath_gcclose:c
5050     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5051   }
5052   {
5053     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5054   }
5055 },

```

Close a path with another path.

```

5056 close~ with/.code~ 2~ args={
5057   \tl_if_exist:cTF
5058   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5059   {
5060     \tl_if_exist:cTF
5061     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
5062     {
5063       \spath_close_with:cv
5064       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5065       {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
5066     }
5067     {
5068       \msg_warning:nnn { spath3 } { missing soft path } { #2 }
5069     }
5070   }
5071   {
5072     \msg_warning:nnn { spath3 } { missing soft path } { #1 }

```

```

5073     }
5074   },
5075   close~ globally~ with/.code~ 2~ args={
5076     \tl_if_exist:cTF
5077     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5078     {
5079       \tl_if_exist:cTF
5080       {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
5081       {
5082         \spath_gcclose_with:cv
5083         {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5084         {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
5085       }
5086       {
5087         \msg_warning:nnn { spath3 } { missing soft path } { #2 }
5088       }
5089     }
5090     {
5091       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5092     }
5093   },

```

These keys shorten the path.

```

5094   shorten~ at~ end/.code~ 2~ args={
5095     \tl_if_exist:cTF
5096     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5097     {
5098       \spath_shorten_at_end:cn
5099       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
5100     }
5101     {
5102       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5103     }
5104   },
5105   shorten~ at~ start/.code~ 2~ args ={
5106     \tl_if_exist:cTF
5107     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5108     {
5109       \spath_shorten_at_start:cn
5110       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
5111     }
5112     {
5113       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5114     }
5115   },
5116   shorten~ at~ both~ ends/.code~ 2~ args={
5117     \tl_if_exist:cTF
5118     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5119     {
5120       \spath_shorten_at_end:cn
5121       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
5122       \spath_shorten_at_start:cn
5123       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
5124     }

```



```

5125     {
5126       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5127     }
5128   },
5129   shorten~ globally~ at~ end/.code~ 2~ args={
5130     \tl_if_exist:cTF
5131     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5132     {
5133       \spath_gshorten_at_end:cn
5134       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
5135     }
5136     {
5137       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5138     }
5139   },
5140   shorten~ globally~ at~ start/.code~ 2~ args ={
5141     \tl_if_exist:cTF
5142     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5143     {
5144       \spath_gshorten_at_start:cn
5145       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
5146     }
5147     {
5148       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5149     }
5150   },
5151   shorten~ globally~ at~ both~ ends/.code~ 2~ args={
5152     \tl_if_exist:cTF
5153     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5154     {
5155       \spath_shorten_at_end:cn
5156       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
5157       \spath_shorten_at_start:cn
5158       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
5159     }
5160     {
5161       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5162     }
5163   },

```

This translates the named path.

```

5164   translate/.code~ n~ args={3}{
5165     \tl_if_exist:cTF
5166     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5167     {
5168       \spath_translate:cnn
5169       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}{#2}{#3}
5170     }
5171     {
5172       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5173     }
5174   },
5175   translate~ globally/.code~ n~ args={3}{
5176     \tl_if_exist:cTF

```

```

5177   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5178   {
5179     \spath_gtranslate:cnn
5180     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}{#2}{#3}
5181   }
5182   {
5183     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5184   }
5185 },

```

This normalises the named path.

```

5186   normalise/.code={
5187     \tl_if_exist:cTF
5188     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5189     {
5190       \spath_normalise:c
5191       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5192     }
5193     {
5194       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5195     }
5196   },
5197   normalise~ globally/.code={
5198     \tl_if_exist:cTF
5199     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5200     {
5201       \spath_gnormalise:c
5202       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5203     }
5204     {
5205       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5206     }
5207   },

```

Transforms the named path using TikZ transformation specifications.

```

5208   transform/.code~ 2~ args={
5209     \tl_if_exist:cTF
5210     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5211     {
5212       \group_begin:
5213       \pgftransformreset
5214       \tikzset{#2}
5215       \pgfgettransform \l__spath_tmpa_tl
5216       \tl_gset:Nn \g__spath_smuggle_tl
5217       {
5218         \spath_transform:cnnnnnn
5219         {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5220       }
5221       \tl_gput_right:NV \g__spath_smuggle_tl \l__spath_tmpa_tl
5222       \group_end:
5223       \tl_use:N \g__spath_smuggle_tl
5224     }
5225     {
5226       \msg_warning:nnn { spath3 } { missing soft path } { #1 }

```

```

5227   }
5228 },
5229 transform~globally/.code~ 2~ args={
5230   \tl_if_exist:cTF
5231   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5232   {
5233     \group_begin:
5234     \pgftransformreset
5235     \tikzset{#2}
5236     \pgfgettransform \l__spath_tmpa_tl
5237     \tl_gset:Nn \g__spath_smuggle_tl
5238     {
5239       \spath_gtransform:cnnnnn
5240       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5241     }
5242     \tl_gput_right:NV \g__spath_smuggle_tl \l__spath_tmpa_tl
5243     \group_end:
5244     \tl_use:N \g__spath_smuggle_tl
5245   }
5246   {
5247     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5248   }
5249 },

```

Splits first path where it intersects with the second.

```

5250 split~ at~ intersections~ with/.code~ n~ args={2}{
5251   \tl_if_exist:cTF
5252   {
5253     tikz@library@intersections@loaded
5254   }
5255   {
5256     \tl_if_exist:cTF
5257     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5258     {
5259       \tl_if_exist:cTF
5260       {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
5261       {
5262         \spath_split_path_at_intersections:cv
5263         {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5264         {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
5265       }
5266       {
5267         \msg_warning:nnn { spath3 } { missing soft path } { #2 }
5268       }
5269     }
5270     {
5271       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5272     }
5273   }
5274   {
5275     \msg_warning:nn { spath3 } { load intersections }
5276   }
5277 },
5278 split~ globally~ at~ intersections~ with/.code~ n~ args={2}{

```

```

5279 \tl_if_exist:cTF
5280 {
5281   tikz@library@intersections@loaded
5282 }
5283 {
5284 \tl_if_exist:cTF
5285   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5286   {
5287     \tl_if_exist:cTF
5288     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
5289     {
5290       \spath_gsplit_path_at_intersections:cv
5291       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5292       {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
5293     }
5294     {
5295       \msg_warning:nnn { spath3 } { missing soft path } { #2 }
5296     }
5297   }
5298   {
5299     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5300   }
5301 }
5302 {
5303   \msg_warning:nn { spath3 } { load intersections }
5304 }
5305 },

```

Splits two paths at their mutual intersections.

```

5306 split~ at~ intersections/.code~ n~ args={2}{
5307   \tl_if_exist:cTF
5308   {
5309     tikz@library@intersections@loaded
5310   }
5311   {
5312     \tl_if_exist:cTF
5313     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5314     {
5315       \tl_if_exist:cTF
5316       {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
5317       {
5318         \spath_split_at_intersections:cc
5319         {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5320         {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
5321       }
5322       {
5323         \msg_warning:nnn { spath3 } { missing soft path } { #2 }
5324       }
5325     }
5326     {
5327       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5328     }
5329   }
5330   {

```

```

5331     \msg_warning:nn { spath3 } { load intersections }
5332   }
5333 },
5334 split~ globally~ at~ intersections/.code~ n~ args={2}{
5335   \tl_if_exist:cTF
5336   {
5337     tikz@library@intersections@loaded
5338   }
5339   {
5340     \tl_if_exist:cTF
5341     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5342     {
5343       \tl_if_exist:cTF
5344       {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
5345       {
5346         \spath_gsplit_at_intersections:cc
5347         {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5348         {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
5349       }
5350       {
5351         \msg_warning:nnn { spath3 } { missing soft path } { #2 }
5352       }
5353     }
5354     {
5355       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5356     }
5357   }
5358   {
5359     \msg_warning:nn { spath3 } { load intersections }
5360   }
5361 },

```

Splits a path at its self-intersections.

```

5362 split~ at~ self~ intersections/.code={
5363   \tl_if_exist:cTF
5364   {
5365     tikz@library@intersections@loaded
5366   }
5367   {
5368     \tl_if_exist:cTF
5369     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5370     {
5371       \spath_split_at_self_intersections:c
5372       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5373     }
5374     {
5375       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5376     }
5377   }
5378   {
5379     \msg_warning:nn { spath3 } { load intersections }
5380   }
5381 },
5382 split~ globally~ at~ self~ intersections/.code={

```

```

5383 \tl_if_exist:cTF
5384 {
5385   tikz@library@intersections@loaded
5386 }
5387 {
5388   \tl_if_exist:cTF
5389   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5390   {
5391     \spath_gsplit_at_self_intersections:c
5392     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5393   }
5394   {
5395     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5396   }
5397 }
5398 {
5399   \msg_warning:nn { spath3 } { load intersections }
5400 }
5401 },

```

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

```

5402 get~ components~ of/.code~ 2~ args={
5403   \tl_if_exist:cTF
5404   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5405   {
5406     \clist_clear_new:N #2
5407     \spath_components_to_seq:Nv
5408     \l__spath_tmpa_seq
5409     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5410     \seq_map_inline:Nn \l__spath_tmpa_seq
5411     {
5412       \tl_new:c
5413       {
5414         \tl_use:N \l__spath_prefix_tl
5415         anonymous_\int_use:N \g__spath_anon_int
5416         \tl_use:N \l__spath_suffix_tl
5417       }
5418       \tl_set:cn
5419       {
5420         \tl_use:N \l__spath_prefix_tl
5421         anonymous_\int_use:N \g__spath_anon_int
5422         \tl_use:N \l__spath_suffix_tl
5423       } {##1}
5424       \clist_put_right:Nx #2 {anonymous_\int_use:N \g__spath_anon_int}
5425       \int_gincr:N \g__spath_anon_int
5426     }
5427   }
5428   {
5429     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5430   }
5431 },
5432 get~ components~ of~ globally/.code~ 2~ args={
5433   \tl_if_exist:cTF

```

```

5434 {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5435 {
5436   \clist_gclear_new:N #2
5437   \spath_components_to_seq:Nv
5438   \l__spath_tmpa_seq
5439   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5440   \seq_map_inline:Nn \l__spath_tmpa_seq
5441   {
5442     \tl_new:c
5443     {
5444       \tl_use:N \l__spath_prefix_tl
5445       anonymous_\int_use:N \g__spath_anon_int
5446       \tl_use:N \l__spath_suffix_tl
5447     }
5448     \tl_gset:cn
5449     {
5450       \tl_use:N \l__spath_prefix_tl
5451       anonymous_\int_use:N \g__spath_anon_int
5452       \tl_use:N \l__spath_suffix_tl
5453     } {##1}
5454     \clist_gput_right:Nx #2 {anonymous_\int_use:N \g__spath_anon_int}
5455     \int_gincr:N \g__spath_anon_int
5456   }
5457 }
5458 {
5459   \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5460 }
5461 },

```

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

```

5462 render~ components/.code={
5463   \tl_if_exist:cTF
5464   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5465   {
5466     \group_begin:
5467     \spath_components_to_seq:Nv
5468     \l__spath_tmpa_seq
5469     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5470     \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
5471     {
5472       \spath_tikz_path:nn
5473       {
5474         every~ spath~ component/.try,
5475         spath ~component~ ##1/.try,
5476         spath ~component/.try={##1},
5477         every~ #1~ component/.try,
5478         #1 ~component~ ##1/.try,
5479         #1 ~component/.try={##1},
5480       }
5481     }
5482     ##2
5483   }
5484 }

```

```

5485     \group_end:
5486   }
5487   {
5488     \msg_warning:nmn { spath3 } { missing soft path } { #1 }
5489   }
5490 },

```

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.

```

5491 insert~ gaps~ after~ components/.code~ 2~ args={
5492   \tl_if_exist:cTF
5493   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5494   {
5495     \group_begin:
5496     \tl_if_head_is_group:nTF {#2}
5497     {
5498       \tl_set:Nx \l__spath_tmpc_tl { \tl_item:nn {#2} {1} }
5499       \tl_set:Nx \l__spath_tmpd_tl { \tl_item:nn {#2} {2} }
5500     }
5501     {
5502       \tl_set:Nn \l__spath_tmpc_tl {#2}
5503       \tl_clear:N \l__spath_tmpd_tl
5504     }
5505     \seq_gclear:N \g__spath_tmpa_seq
5506     \seq_gclear:N \g__spath_tmpb_seq
5507     \spath_numberofcomponents:Nv \l__spath_tmpa_int
5508     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5509
5510     \spath_components_to_seq:Nv
5511     \l__spath_tmpa_seq
5512     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5513
5514     \tl_if_empty:NTF \l__spath_tmpd_tl
5515     {
5516       \int_step_inline:nnnn {1}{1} { \l__spath_tmpa_int - 1 }
5517       {
5518         \seq_gput_right:Nn \g__spath_tmpa_seq {##1}
5519         \seq_gput_right:Nx
5520         \g__spath_tmpb_seq
5521         {\int_eval:n {##1 + 1}}
5522       }
5523     }
5524     {
5525       \foreach \l__spath_tmpa_tl in \l__spath_tmpd_tl
5526       {
5527         \seq_gput_right:Nv \g__spath_tmpa_seq \l__spath_tmpa_tl
5528         \seq_gput_right:Nx
5529         \g__spath_tmpb_seq
5530         {\int_eval:n
5531          {
5532            \int_mod:nn { \l__spath_tmpa_tl }{ \l__spath_tmpa_int } + 1
5533          }
5534        }
5535       }

```



```

5536     }
5537
5538     \seq_clear:N \l__spath_tmpb_seq
5539     \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
5540     {
5541         \tl_set:Nn \l__spath_tmpa_tl {##2}
5542         \seq_if_in:NnT \g__spath_tmpa_seq {##1}
5543         {
5544             \spath_shorten_at_end:Nn \l__spath_tmpa_tl {\l__spath_tmpc_tl/2}
5545         }
5546         \seq_if_in:NnT \g__spath_tmpb_seq {##1}
5547         {
5548             \spath_shorten_at_start:Nn \l__spath_tmpa_tl {\l__spath_tmpc_tl/2}
5549         }
5550         \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
5551     }
5552     \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpb_seq {} }
5553     \group_end:
5554     \tl_set_eq:cN
5555     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5556     \g__spath_output_tl
5557     \tl_gclear:N \g__spath_output_tl
5558 }
5559 {
5560     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5561 }
5562 },
5563 insert~ gaps~ globally~ after~ components/.code~ 2~ args={
5564     \tl_if_exist:cTF
5565     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5566     {
5567         \group_begin:
5568         \tl_if_head_is_group:nTF {#2}
5569         {
5570             \tl_set:Nx \l__spath_tmpc_tl { \tl_item:nn {#2} {1} }
5571             \tl_set:Nx \l__spath_tmpd_tl { \tl_item:nn {#2} {2} }
5572         }
5573         {
5574             \tl_set:Nn \l__spath_tmpc_tl {#2}
5575             \tl_clear:N \l__spath_tmpd_tl
5576         }
5577         \seq_gclear:N \g__spath_tmpa_seq
5578         \seq_gclear:N \g__spath_tmpb_seq
5579         \spath_numberofcomponents:Nv \l__spath_tmpa_int
5580         {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5581
5582         \spath_components_to_seq:Nv
5583         \l__spath_tmpa_seq
5584         {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5585
5586         \tl_if_empty:NTF \l__spath_tmpd_tl
5587         {
5588             \int_step_inline:nnnn {1}{1} { \l__spath_tmpa_int - 1 }
5589             {

```

```

5590     \seq_gput_right:Nn \g__spath_tmpa_seq {##1}
5591     \seq_gput_right:Nx
5592     \g__spath_tmpb_seq
5593     {\int_eval:n {##1 + 1}}
5594   }
5595 }
5596 {
5597   \foreach \l__spath_tmpa_tl in \l__spath_tmpd_tl
5598   {
5599     \seq_gput_right:NV \g__spath_tmpa_seq \l__spath_tmpa_tl
5600     \seq_gput_right:Nx
5601     \g__spath_tmpb_seq
5602     {\int_eval:n
5603      {
5604        \int_mod:nn { \l__spath_tmpa_tl }{ \l__spath_tmpa_int } + 1
5605      }
5606    }
5607   }
5608 }
5609
5610 \seq_clear:N \l__spath_tmpb_seq
5611 \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
5612 {
5613   \tl_set:Nn \l__spath_tmpa_tl {##2}
5614   \seq_if_in:NnT \g__spath_tmpa_seq {##1}
5615   {
5616     \spath_shorten_at_end:Nn \l__spath_tmpa_tl {\l__spath_tmpc_tl/2}
5617   }
5618   \seq_if_in:NnT \g__spath_tmpb_seq {##1}
5619   {
5620     \spath_shorten_at_start:Nn \l__spath_tmpa_tl {\l__spath_tmpc_tl/2}
5621   }
5622   \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
5623 }
5624 \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpb_seq {} }
5625 \group_end:
5626 \tl_gset_eq:cN
5627 {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5628 \g__spath_output_tl
5629 \tl_gclear:N \g__spath_output_tl
5630 }
5631 {
5632   \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5633 }
5634 },

```

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

```

5635 join~ components/.code~ 2~ args={
5636   \tl_if_exist:cTF
5637   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5638   {
5639     \seq_gclear:N \g__spath_tmpa_seq
5640     \foreach \l__spath_tmpa_tl in {#2}
5641     {

```

```

5642     \seq_gput_right:NV \g__spath_tmpa_seq \l__spath_tmpa_tl
5643   }
5644   \seq_gsort:Nn \g__spath_tmpa_seq
5645   {
5646     \int_compare:nNnTF {##1} > {##2}
5647     { \sort_return_same: }
5648     { \sort_return_swapped: }
5649   }
5650   \seq_map_inline:Nn \g__spath_tmpa_seq
5651   {
5652     \spath_join_component:cn
5653     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}{##1}
5654   }
5655   {
5656     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5657   }
5658 },
5659 join~ components~ globally/.code~ 2~ args={
5660   \tl_if_exist:cTF
5661   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5662   {
5663     \seq_gclear:N \g__spath_tmpa_seq
5664     \foreach \l__spath_tmpa_tl in {#2}
5665     {
5666       \seq_gput_right:NV \g__spath_tmpa_seq \l__spath_tmpa_tl
5667     }
5668     \seq_gsort:Nn \g__spath_tmpa_seq
5669     {
5670       \int_compare:nNnTF {##1} > {##2}
5671       { \sort_return_same: }
5672       { \sort_return_swapped: }
5673     }
5674     \seq_map_inline:Nn \g__spath_tmpa_seq
5675     {
5676       \spath_gjoin_component:cn
5677       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}{##1}
5678     }
5679   }
5680   {
5681     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5682   }
5683 },
5684 },

```

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

```

5685 remove~ empty~ components/.code={
5686   \tl_if_exist:cTF
5687   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5688   {
5689     \spath_remove_empty_components:c
5690     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5691   }
5692   {
5693     \msg_warning:nnn { spath3 } { missing soft path } { #1 }

```

```

5694     }
5695   },
5696   remove~ empty~ components~ globally/.code={
5697     \tl_if_exist:cTF
5698     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5699     {
5700       \spath_gremove_empty_components:c
5701       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5702     }
5703     {
5704       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5705     }
5706   },

```

Replace all line segments by Bézier curves.

```

5707   replace~ lines/.code={
5708     \tl_if_exist:cTF
5709     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5710     {
5711       \spath_replace_lines:c
5712       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5713     }
5714     {
5715       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5716     }
5717   },
5718   replace~ lines~ globally/.code={
5719     \tl_if_exist:cTF
5720     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5721     {
5722       \spath_greplace_lines:c
5723       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5724     }
5725     {
5726       \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5727     }
5728   },

```

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

```

5729   remove~ components/.code~ 2~ args={
5730     \tl_if_exist:cTF
5731     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5732     {
5733       \seq_gclear:N \g__spath_tmpa_seq
5734       \foreach \l__spath_tmpa_tl in {#2}
5735       {
5736         \seq_gput_right:NV \g__spath_tmpa_seq \l__spath_tmpa_tl
5737       }
5738       \seq_gsort:Nn \g__spath_tmpa_seq
5739       {
5740         \int_compare:nNnTF {##1} < {##2}
5741         { \sort_return_same: }
5742         { \sort_return_swapped: }
5743       }

```

```

5744 \spath_components_to_seq:Nv
5745 \l__spath_tmpa_seq
5746 {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5747 \seq_gpop_left:NNF \g__spath_tmpa_seq \l__spath_tmpa_tl
5748 {
5749   \tl_clear:N \l__spath_tmpa_tl
5750 }
5751 \seq_clear:N \l__spath_tmpb_seq
5752 \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
5753 {
5754   \tl_set:Nn \l__spath_tmpb_tl {##1}
5755   \tl_if_eq:NNTF \l__spath_tmpb_tl \l__spath_tmpa_tl
5756   {
5757     \seq_gpop_left:NNF \g__spath_tmpa_seq \l__spath_tmpa_tl
5758     {
5759       \tl_clear:N \l__spath_tmpa_tl
5760     }
5761   }
5762   {
5763     \seq_put_right:Nn \l__spath_tmpb_seq {##2}
5764   }
5765 }
5766 \tl_set:cx {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5767 {\seq_use:Nn \l__spath_tmpb_seq {}}
5768 }
5769 {
5770   \msg_warning:nmn { spath3 } { missing soft path } { #1 }
5771 }
5772 },
5773 remove~ components~ globally/.code~ 2~ args={
5774   \tl_if_exist:cTF
5775   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5776   {
5777     \seq_gclear:N \g__spath_tmpa_seq
5778     \foreach \l__spath_tmpa_tl in {#2}
5779     {
5780       \seq_gput_right:NV \g__spath_tmpa_seq \l__spath_tmpa_tl
5781     }
5782     \seq_gsort:Nn \g__spath_tmpa_seq
5783     {
5784       \int_compare:nNnTF {##1} < {##2}
5785       { \sort_return_same: }
5786       { \sort_return_swapped: }
5787     }
5788     \spath_components_to_seq:Nv
5789     \l__spath_tmpa_seq
5790     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5791     \seq_gpop_left:NNF \g__spath_tmpa_seq \l__spath_tmpa_tl
5792     {
5793       \tl_clear:N \l__spath_tmpa_tl
5794     }
5795     \seq_clear:N \l__spath_tmpb_seq
5796     \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
5797     {

```

```

5798     \tl_set:Nn \l__spath_tmpb_tl {##1}
5799     \tl_if_eq:NNTF \l__spath_tmpb_tl \l__spath_tmpa_tl
5800     {
5801         \seq_gpop_left:NNTF \g__spath_tmpa_seq \l__spath_tmpa_tl
5802         {
5803             \tl_clear:N \l__spath_tmpa_tl
5804         }
5805     }
5806     {
5807         \seq_put_right:Nn \l__spath_tmpb_seq {##2}
5808     }
5809 }
5810 \tl_gset:cx {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5811 {\seq_use:Nn \l__spath_tmpb_seq {}}
5812 }
5813 {
5814     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5815 }
5816 },

```

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.

```

5817 draft~ mode/.is~ choice,
5818 draft~ mode/true/.code={
5819     \bool_set_true:N \l__spath_draft_bool
5820 },
5821 draft~ mode/false/.code={
5822     \bool_set_false:N \l__spath_draft_bool
5823 },
5824 maybe~ spot~ weld/.code={
5825     \bool_if:NF \l__spath_draft_bool
5826     {
5827         \tl_if_exist:cTF
5828         {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5829         {
5830             \spath_spot_weld_components:c
5831             {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5832         }
5833         {
5834             \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5835         }
5836     }
5837 },
5838 maybe~ spot~ weld~ globally/.code={
5839     \bool_if:NF \l__spath_draft_bool
5840     {
5841         \tl_if_exist:cTF
5842         {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5843         {
5844             \spath_spot_gweld_components:c
5845             {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
5846         }
5847     }

```

```

5848         \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5849     }
5850 }
5851 },

```

Set the transformation to lie along a path.

```

5852 transform~ to/.code~ 2~ args={
5853   \group_begin:
5854   \tl_if_exist:cTF
5855   {
5856     \tl_use:N \l__spath_prefix_tl
5857     #1
5858     \tl_use:N \l__spath_suffix_tl
5859   }
5860   {
5861     \spath_reallength:Nv
5862     \l__spath_tmpa_int
5863     {
5864       \tl_use:N \l__spath_prefix_tl
5865       #1
5866       \tl_use:N \l__spath_suffix_tl
5867     }
5868
5869     \tl_set:Nx \l__spath_tmpb_tl
5870     {\fp_to_decimal:n {(#2) * (\l__spath_tmpa_int)}}
5871     \spath_transformation_at:NvV \l__spath_tmpc_tl
5872     {
5873       \tl_use:N \l__spath_prefix_tl
5874       #1
5875       \tl_use:N \l__spath_suffix_tl
5876     }
5877     \l__spath_tmpb_tl
5878     \tl_gset_eq:NN \g__spath_smuggle_tl \l__spath_tmpc_tl
5879   }
5880   {
5881     \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5882     \tl_gset_eq:NN \g__spath_smuggle_tl { {1}{0}{1}{Opt}{Opt} }
5883   }
5884   \group_end:
5885   \exp_last_unbraced:Nv \pgfsettransformentries \g__spath_smuggle_tl
5886   \tl_gclear:N \g__spath_smuggle_tl
5887 },

```

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

```

5888 upright~ transform~ to/.code~ 2~ args={
5889   \group_begin:
5890   \tl_if_exist:cTF
5891   {
5892     \tl_use:N \l__spath_prefix_tl
5893     #1
5894     \tl_use:N \l__spath_suffix_tl
5895   }
5896   {

```

```

5897 \spath_reallength:Nv
5898 \l__spath_tmpa_int
5899 {
5900 \tl_use:N \l__spath_prefix_tl
5901 #1
5902 \tl_use:N \l__spath_suffix_tl
5903 }
5904
5905 \tl_set:Nx \l__spath_tmpb_tl
5906 {\fp_to_decimal:n {(#2) * (\l__spath_tmpa_int)}}
5907 \spath_transformation_at:NvV \l__spath_tmpc_tl
5908 {
5909 \tl_use:N \l__spath_prefix_tl
5910 #1
5911 \tl_use:N \l__spath_suffix_tl
5912 }
5913 \l__spath_tmpb_tl
5914 \tl_gset_eq:NN \g__spath_smuggle_tl \l__spath_tmpc_tl
5915 }
5916 {
5917 \msg_warning:nnn { spath3 } { missing soft path } { #1 }
5918 \tl_gset_eq:NN \g__spath_smuggle_tl { {1}{0}{0}{1}{0pt}{0pt} }
5919 }
5920 \fp_compare:nT { \tl_item:Nn \g__spath_smuggle_tl {4} < 0 }
5921 {
5922 \tl_gset:Nx \g__spath_smuggle_tl
5923 {
5924 { \fp_eval:n { - (\tl_item:Nn \g__spath_smuggle_tl {1}) } }
5925 { \fp_eval:n { - (\tl_item:Nn \g__spath_smuggle_tl {2}) } }
5926 { \fp_eval:n { - (\tl_item:Nn \g__spath_smuggle_tl {3}) } }
5927 { \fp_eval:n { - (\tl_item:Nn \g__spath_smuggle_tl {4}) } }
5928 { \tl_item:Nn \g__spath_smuggle_tl {5} }
5929 { \tl_item:Nn \g__spath_smuggle_tl {6} }
5930 }
5931 }
5932 \group_end:
5933 \exp_last_unbraced:NV \pgfsettransformentries \g__spath_smuggle_tl
5934 \tl_gclear:N \g__spath_smuggle_tl
5935 },

```

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

```

5936 knot/.style~ n~ args={3}{
5937   spath/split~ at~ self~ intersections=#1,
5938   spath/remove~ empty~ components=#1,
5939   spath/insert~ gaps~ after~ components={#1}{#2}{#3},
5940   spath/maybe~ spot~ weld=#1,
5941   spath/render~ components=#1
5942 },
5943 global~ knot/.style~ n~ args={3}{
5944   spath/split~ globally~ at~ self~ intersections=#1,
5945   spath/remove~ empty~ components~ globally=#1,
5946   spath/insert~ gaps~ globally ~after~ components={#1}{#2}{#3},
5947   spath/maybe~ spot~ weld~ globally=#1,
5948   spath/render~ components=#1

```



```

5949 },
5950 }

```

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

```

5951 \tikzdeclarecoordinatesystem{spath}{%
5952   \group_begin:
5953   \tl_set:Nn \l__spath_tmpa_tl {#1}
5954   \tl_trim_spaces:N \l__spath_tmpa_tl
5955
5956   \seq_set_split:NnV \l__spath_tmpa_seq {~} \l__spath_tmpa_tl
5957   \seq_pop_right:NN \l__spath_tmpa_seq \l__spath_tmpb_tl
5958
5959   \tl_set:Nx \l__spath_tmpa_tl { \seq_use:Nn \l__spath_tmpa_seq {-} }
5960   \tl_if_exist:cTF
5961   {
5962     \tl_use:N \l__spath_prefix_tl
5963     \tl_use:N \l__spath_tmpa_tl
5964     \tl_use:N \l__spath_suffix_tl
5965   }
5966   {
5967
5968     \tl_set_eq:Nc
5969     \l__spath_tmpa_tl
5970     {
5971       \tl_use:N \l__spath_prefix_tl
5972       \tl_use:N \l__spath_tmpa_tl
5973       \tl_use:N \l__spath_suffix_tl
5974     }
5975
5976     \tl_if_empty:NTF \l__spath_tmpa_tl
5977     {
5978       \tl_gset_eq:NN \g__spath_smuggle_tl \pgfpointorigin
5979     }
5980     {
5981       \spath_reallength:NV \l__spath_tmpa_int \l__spath_tmpa_tl
5982       \tl_set:Nx \l__spath_tmpb_tl
5983       {\fp_to_decimal:n {(\l__spath_tmpb_tl) * (\l__spath_tmpa_int)}}
5984       \spath_point_at:NVV \l__spath_tmpc_tl \l__spath_tmpa_tl \l__spath_tmpb_tl
5985
5986       \tl_clear:N \l__spath_tmpd_tl
5987       \tl_put_right:Nn \l__spath_tmpd_tl {\pgf@x=}
5988       \tl_put_right:Nx \l__spath_tmpd_tl {\tl_item:Nn \l__spath_tmpc_tl {1}}
5989       \tl_put_right:Nn \l__spath_tmpd_tl {\relax}
5990       \tl_put_right:Nn \l__spath_tmpd_tl {\pgf@y=}
5991       \tl_put_right:Nx \l__spath_tmpd_tl {\tl_item:Nn \l__spath_tmpc_tl {2}}
5992       \tl_put_right:Nn \l__spath_tmpd_tl {\relax}
5993       \tl_gset_eq:NN \g__spath_smuggle_tl \l__spath_tmpd_tl
5994     }
5995   }
5996   {
5997     \msg_warning:nxx
5998     { spath3 }
5999     { missing soft path }
6000     { \tl_use:N \l__spath_tmpa_tl }

```

```

6001     \tl_gset_eq:NN \g__spath_smuggle_tl \pgfpointorigin
6002   }
6003   \group_end:
6004   \use:c {pgf@process}{%
6005     \tl_use:N \g__spath_smuggle_tl
6006     \pgftransforminvert
6007     \use:c {pgf@pos@transform@glob}
6008   }
6009 }
6010
6011 \ExplSyntaxOff

```

5 The Calligraphy Package

```

6012 <@@=cal>

```

5.1 Initialisation

```

6013 \RequirePackage{spath3}
6014 \ExplSyntaxOn
6015
6016 \tl_new:N \l__cal_tmpa_tl
6017 \tl_new:N \l__cal_tmpb_tl
6018 \tl_new:N \l__cal_tmp_path_tl
6019 \tl_new:N \l__cal_tmp_rpath_tl
6020 \tl_new:N \l__cal_tmp_rpathb_tl
6021 \tl_new:N \l__cal_tmp_patha_tl
6022
6023 \seq_new:N \l__cal_tmpa_seq
6024
6025 \int_new:N \l__cal_tmpa_int
6026 \int_new:N \l__cal_tmpb_int
6027 \int_new:N \g__cal_path_component_int
6028 \int_new:N \g__cal_label_int
6029
6030 \fp_new:N \l__cal_tmpa_fp
6031 \fp_new:N \l__cal_tmpb_fp
6032 \fp_new:N \l__cal_tmpe_fp
6033 \fp_new:N \l__cal_tmpe_fp
6034 \fp_new:N \l__cal_tmpe_fp
6035
6036 \dim_new:N \l__cal_tmpe_dim
6037 \dim_new:N \l__cal_tmpe_dim
6038 \dim_new:N \l__cal_tmpe_dim
6039 \dim_new:N \l__cal_tmpe_dim
6040 \dim_new:N \l__cal_tmpe_dim
6041 \dim_new:N \l__cal_tmpe_dim
6042 \dim_new:N \l__cal_tmpe_dim
6043 \dim_new:N \l__cal_tmpe_dim
6044
6045 \bool_new:N \l__cal_annotate_bool
6046 \bool_new:N \l__cal_taper_start_bool
6047 \bool_new:N \l__cal_taper_end_bool
6048 \bool_new:N \l__cal_taperable_bool

```

```

6049
6050 \dim_new:N \l__cal_taper_width_dim
6051 \dim_new:N \l__cal_line_width_dim
6052
6053 \bool_set_true:N \l__cal_taper_start_bool
6054 \bool_set_true:N \l__cal_taper_end_bool
6055
6056 \cs_generate_variant:Nn \tl_put_right:Nn {Nv}
6057
6058 \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.

```

6059 \tikzset{
6060   define~pen/.code={
6061     \tikzset{pen~name=#1}
6062     \pgf@relevantforpicturesizefalse
6063     \tikz@addmode{
6064       \pgfsyssoftpath@getcurrentpath\l__cal_tmpa_tl
6065       \spath_components_to_seq:NV \l__cal_tmpa_seq \l__cal_tmpa_tl
6066       \seq_gclear_new:c {g__cal_pen_\pgfkeysvalueof{/tikz/pen~name}_seq}
6067       \seq_gset_eq:cN
6068       {g__cal_pen_\pgfkeysvalueof{/tikz/pen~name}_seq} \l__cal_tmpa_seq
6069       \pgfusepath{discard}%
6070     }
6071   },
6072   define~pen/.default={default},
6073   use~pen/.code={
6074     \tikzset{pen~name=#1}
6075     \int_gzero:N \g__cal_path_component_int
6076     \cs_set_eq:NN \pgfpathmoveto \cal_moveto:n
6077     \tikz@addmode{
6078       \pgfsyssoftpath@getcurrentpath\l__cal_tmpa_tl
6079       \spath_components_to_seq:NV \l__cal_tmpa_seq \l__cal_tmpa_tl
6080       \tl_if_exist:cTF {g__cal_pen_\pgfkeysvalueof{/tikz/pen~name}_seq}
6081       {
6082         \cal_path_create:Nc \l__cal_tmpa_seq
6083         {g__cal_pen_\pgfkeysvalueof{/tikz/pen~name}_seq}
6084       }
6085       {
6086         \msg_warning:nx { calligraphy } { undefined pen }
6087         { \pgfkeysvalueof{/tikz/pen~name} }
6088       }
6089     }
6090   },
6091   use~pen/.default={default},
6092   pen~name/.initial={default},
6093   copperplate/.style={pen~name=copperplate},
6094   pen~colour/.initial={black},
6095   weight/.is~choice,
6096   weight/heavy/.style={
6097     line~width=\pgfkeysvalueof{/tikz/heavy~line~width},
6098     taper~width=\pgfkeysvalueof{/tikz/light~line~width},

```

```

6099 },
6100 weight/light/.style={
6101   line-width=\pgfkeysvalueof{/tikz/light-line-width},
6102   taper-width=0pt,
6103 },
6104 heavy/.style={
6105   weight=heavy
6106 },
6107 light/.style={
6108   weight=light
6109 },
6110 heavy~line-width/.initial=2pt,
6111 light~line-width/.initial=1pt,
6112 taper/.is~choice,
6113 taper/.default=both,
6114 taper/none/.style={
6115   taper~start=false,
6116   taper~end=false,
6117 },
6118 taper/both/.style={
6119   taper~start=true,
6120   taper~end=true,
6121 },
6122 taper/start/.style={
6123   taper~start=true,
6124   taper~end=false,
6125 },
6126 taper/end/.style={
6127   taper~start=false,
6128   taper~end=true,
6129 },
6130 taper~start/.code={
6131   \tl_if_eq:nnTF {#1} {true}
6132   {
6133     \bool_set_true:N \l__cal_taper_start_bool
6134   }
6135   {
6136     \bool_set_false:N \l__cal_taper_start_bool
6137   }
6138 },
6139 taper~start/.default={true},
6140 taper~end/.code={
6141   \tl_if_eq:nnTF {#1} {true}
6142   {
6143     \bool_set_true:N \l__cal_taper_end_bool
6144   }
6145   {
6146     \bool_set_false:N \l__cal_taper_end_bool
6147   }
6148 },
6149 taper~end/.default={true},
6150 taper~width/.code={\dim_set:Nn \l__cal_taper_width_dim {#1}},
6151 nib~style/.code~2~args={
6152   \tl_clear_new:c {l__cal_nib_style_#1}

```

```

6153     \tl_set:cn {l__cal_nib_style_#1} {#2}
6154   },
6155   stroke~style/.code~2~args={
6156     \tl_clear_new:c {l__cal_stroke_style_#1}
6157     \tl_set:cn {l__cal_stroke_style_#1} {#2}
6158   },
6159   this~stroke~style/.code={
6160     \tl_clear_new:c
6161     {l__cal_stroke_inline_style_ \int_use:N \g__cal_path_component_int}
6162     \tl_set:cn
6163     {l__cal_stroke_inline_style_ \int_use:N \g__cal_path_component_int} {#1}
6164   },
6165   annotate/.style={
6166     annotate~if,
6167     annotate~reset,
6168     annotation~style/.update~value={#1},
6169   },
6170   annotate~if/.default={true},
6171   annotate~if/.code={
6172     \tl_if_eq:nnTF {#1} {true}
6173     {
6174       \bool_set_true:N \l__cal_annotate_bool
6175     }
6176     {
6177       \bool_set_false:N \l__cal_annotate_bool
6178     }
6179   },
6180   annotate~reset/.code={
6181     \int_gzero:N \g__cal_label_int
6182   },
6183   annotation~style/.initial={draw,->},
6184   annotation~shift/.initial={(0,1ex)},
6185   every~annotation~node/.initial={anchor=south~west},
6186   annotation~node~style/.code~2~args={
6187     \tl_clear_new:c {l__cal_annotation_style_ #1 _tl}
6188     \tl_set:cn {l__cal_annotation_style_ #1 _tl}{#2}
6189   },
6190   tl~use:N/.code={
6191     \exp_args:NV \pgfkeysalso #1
6192   },
6193   tl~use:c/.code={
6194     \tl_if_exist:cT {#1}
6195     {
6196       \exp_args:Nv \pgfkeysalso {#1}
6197     }
6198   },
6199   /handlers/.update~style/.code={
6200     \tl_if_eq:nnF {#1} {\pgfkeysnovalue}
6201     {
6202       \pgfkeys{\pgfkeyscurrentpath/.code=\pgfkeysalso{#1}}
6203     }
6204   },
6205   /handlers/.update~value/.code={
6206     \tl_if_eq:nnF {#1} {\pgfkeysnovalue}

```

```

6207     {
6208     \pgfkeyssetvalue{\pgfkeyscurrentpath}{#1}
6209     }
6210   },
6211 }

```

Some wrappers around the TikZ keys.

```

6212 \NewDocumentCommand \pen { 0{} }
6213 {
6214   \path[define~ pen,every~ calligraphy~ pen/.try,#1]
6215 }
6216
6217 \NewDocumentCommand \definepen { 0{} }
6218 {
6219   \tikz \path[define~ pen,every~ calligraphy~ pen/.try,#1]
6220 }
6221
6222 \NewDocumentCommand \calligraphy { 0{} }
6223 {
6224   \path[use~ pen,every~ calligraphy/.try,#1]
6225 }

```

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

```

6226 \cs_new_protected_nopar:Npn \cal_path_create:NN #1#2
6227 {
6228   \int_zero:N \l__cal_tmpa_int
6229   \seq_map_inline:Nn #1
6230   {
6231     \int_compare:nT {\tl_count:n {##1} > 3}
6232     {
6233
6234       \int_incr:N \l__cal_tmpa_int
6235       \int_zero:N \l__cal_tmpb_int
6236
6237       \tl_set:Nn \l__cal_tmp_path_tl {##1}
6238       \spath_open:N \l__cal_tmp_path_tl
6239       \spath_reverse:NV \l__cal_tmp_rpath_tl \l__cal_tmp_path_tl
6240
6241       \seq_map_inline:Nn #2
6242       {
6243         \int_incr:N \l__cal_tmpb_int
6244         \group_begin:
6245         \pgfsys@beginscope
6246         \cal_apply_style:c {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_int}
6247         \cal_apply_style:c {l__cal_stroke_inline_style_ \int_use:N \l__cal_tmpa_int}
6248         \cal_apply_style:c {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}
6249
6250         \spath_initialpoint:Nn \l__cal_tmpa_tl {###1}
6251         \tl_set_eq:NN \l__cal_tmp_patha_tl \l__cal_tmp_path_tl
6252         \spath_translate:NV \l__cal_tmp_patha_tl \l__cal_tmpa_tl
6253

```

```

6254 \int_compare:nTF {\tl_count:n {####1} == 3}
6255 {
6256 \cal_at_least_three:N \l__cal_tmp_patha_tl
6257 \spath_protocol_path:V \l__cal_tmp_patha_tl
6258
6259 \tikz@options
6260 \dim_set:Nn \l__cal_line_width_dim {\pgflinewidth}
6261 \cal_maybe_taper:N \l__cal_tmp_patha_tl
6262 }
6263 {
6264 \spath_weld:Nn \l__cal_tmp_patha_tl {####1}
6265 \spath_weld:NV \l__cal_tmp_patha_tl \l__cal_tmp_rpath_tl
6266 \spath_reverse:Nn \l__cal_tmp_rpathb_tl {####1}
6267 \spath_weld:NV \l__cal_tmp_patha_tl \l__cal_tmp_rpathb_tl
6268
6269 \tl_clear:N \l__cal_tmpa_tl
6270 \tl_set:Nn \l__cal_tmpa_tl
6271 {
6272 fill=\pgfkeysvalueof{/tikz/pen-colour},
6273 draw=none
6274 }
6275 \tl_if_exist:cT {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_int}
6276 {
6277 \tl_put_right:Nv \l__cal_tmpa_tl
6278 {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_int}
6279 }
6280 \tl_if_exist:cT {l__cal_stroke_inline_style_ \int_use:N \l__cal_tmpa_int}
6281 {
6282 \tl_put_right:Nn \l__cal_tmpa_tl {,}
6283 \tl_put_right:Nv \l__cal_tmpa_tl
6284 {l__cal_stroke_inline_style_ \int_use:N \l__cal_tmpa_int}
6285 }
6286 \tl_if_exist:cT {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}
6287 {
6288 \tl_put_right:Nn \l__cal_tmpa_tl {,}
6289 \tl_put_right:Nv \l__cal_tmpa_tl
6290 {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}
6291 }
6292 \spath_tikz_path:VV \l__cal_tmpa_tl \l__cal_tmp_patha_tl
6293 }
6294 \pgfsys@endscope
6295 \group_end:
6296 }
6297
6298 \bool_if:NT \l__cal_annotate_bool
6299 {
6300 \seq_get_right:NN #2 \l__cal_tmpa_tl
6301 \spath_finalpoint:NV \l__cal_tmpa_tl \l__cal_tmpa_tl
6302 \spath_translate:NV \l__cal_tmp_path_tl \l__cal_tmpa_tl
6303 \tikz@scan@one@point
6304 \pgfutil@firstofone
6305 \pgfkeysvalueof{/tikz/annotation~shift}
6306
6307 \spath_translate:Nnn \l__cal_tmp_path_tl {\pgf@x} {\pgf@y}

```

```

6308
6309     \pgfkeysgetvalue{/tikz/annotation~style}{\l__cal_tmpa_tl}
6310     \spath_tikz_path:VV \l__cal_tmpa_tl \l__cal_tmp_path_tl
6311
6312     \spath_finalpoint:NV \l__cal_tmpa_tl \l__cal_tmp_path_tl
6313
6314     \exp_last_unbraced:NV \pgfqpoint \l__cal_tmpa_tl
6315     \begin{scope}[reset~ cm]
6316     \node[
6317         every~annotation~node/.try,
6318         tl~use:c = {\l__cal_annotation_style_ \int_use:N \l__cal_tmpa_int _tl}
6319     ] at (\pgf@x,\pgf@y) {\int_use:N \l__cal_tmpa_int};
6320     \end{scope}
6321     }
6322 }
6323 }
6324 }
6325 \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.

```

6326 \cs_new_eq:NN \cal_orig_moveto:n \pgfpathmoveto
6327 \cs_new_nopar:Npn \cal_moveto:n #1
6328 {
6329     \int_gincr:N \g__cal_path_component_int
6330     \cal_orig_moveto:n {#1}
6331 }

```

(End definition for \cal_moveto:n.)

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

```

6332 \cs_new_nopar:Npn \cal_apply_style:N #1
6333 {
6334     \tl_if_exist:NT #1 {
6335         \exp_args:NV \tikzset #1
6336     }
6337 }
6338 \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.

```

6339 \cs_new_protected_nopar:Npn \cal_at_least_three:Nn #1#2
6340 {
6341     \spath_reallength:Nn \l__cal_tmpa_int {#2}
6342     \tl_clear:N \l__cal_tmpb_tl
6343     \tl_set:Nn \l__cal_tmpb_tl {#2}
6344     \int_compare:nTF {\l__cal_tmpa_int = 1}
6345     {
6346         \spath_split_at:Nn \l__cal_tmpb_tl {2/3}
6347         \spath_split_at:Nn \l__cal_tmpb_tl {1/2}
6348     }

```



```

6349 {
6350   \int_compare:nT {\l__cal_tmpa_int = 2}
6351   {
6352     \spath_split_at:Nn \l__cal_tmpb_tl {1.5}
6353     \spath_split_at:Nn \l__cal_tmpb_tl {.5}
6354   }
6355 }
6356 \tl_set_eq:NN #1 \l__cal_tmpb_tl
6357 }
6358 \cs_generate_variant:Nn \cal_at_least_three:Nn {NV}
6359 \cs_new_protected_nopar:Npn \cal_at_least_three:N #1
6360 {
6361   \cal_at_least_three:NV #1#1
6362 }
6363 \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.

```

6364 \cs_new_protected_nopar:Npn \cal_maybe_taper:N #1
6365 {
6366   \tl_set_eq:NN \l__cal_tmpa_tl #1
6367
6368   \bool_if:NT \l__cal_taper_start_bool
6369   {
6370
6371     \dim_set:Nn \l__cal_tmpa_dim {\tl_item:Nn \l__cal_tmpa_tl {2}}
6372     \dim_set:Nn \l__cal_tmpb_dim {\tl_item:Nn \l__cal_tmpa_tl {3}}
6373     \tl_set:Nx \l__cal_tmpb_tl {\tl_item:Nn \l__cal_tmpa_tl {4}}
6374
6375     \tl_case:NnF \l__cal_tmpb_tl
6376     {
6377       \c_spath_lineto_tl
6378       {
6379
6380         \bool_set_true:N \l__cal_taperable_bool
6381         \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {5}}
6382         \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {6}}
6383         \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpa_dim + \l__cal_tmph_dim)/3}
6384         \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmph_dim + \l__cal_tmpe_dim)/3}
6385         \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmpa_dim + 2\l__cal_tmpe_dim)/3}
6386         \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmph_dim + 2\l__cal_tmpe_dim)/3}
6387         \prg_replicate:nn {4}
6388         {
6389           \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
6390         }
6391         \tl_put_left:NV \l__cal_tmpa_tl \c_spath_moveto_tl
6392       }
6393     }
6394     \c_spath_curvetoa_tl
6395     {
6396       \bool_set_true:N \l__cal_taperable_bool
6397       \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {5}}
6398       \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {6}}
6399       \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {8}}

```

```

6399     \dim_set:Nn \l__cal_tmpf_dim {\tl_item:Nn \l__cal_tmpa_tl {9}}
6400     \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {11}}
6401     \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {12}}
6402     \prg_replicate:nn {10}
6403     {
6404         \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
6405     }
6406     \tl_put_left:NV \l__cal_tmpa_tl \c_spath_moveto_tl
6407 }
6408 }
6409 {
6410     \bool_set_false:N \l__cal_taperable_bool
6411 }
6412
6413 \bool_if:NT \l__cal_taperable_bool
6414 {
6415     \__cal_taper_aux:
6416 }
6417
6418 }
6419
6420 \bool_if:NT \l__cal_taper_end_bool
6421 {
6422
6423     \dim_set:Nn \l__cal_tmpa_dim {\tl_item:Nn \l__cal_tmpa_tl {-2}}
6424     \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {-1}}
6425     \tl_set:Nx \l__cal_tmph_tl {\tl_item:Nn \l__cal_tmpa_tl {-3}}
6426
6427     \tl_case:NnF \l__cal_tmph_tl
6428     {
6429         \c_spath_lineto_tl
6430     {
6431
6432         \bool_set_true:N \l__cal_taperable_bool
6433         \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {-5}}
6434         \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {-4}}
6435         \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpa_dim + \l__cal_tmph_dim)/3}
6436         \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmph_dim + \l__cal_tmpe_dim)/3}
6437         \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmpa_dim + 2\l__cal_tmpe_dim)/3}
6438         \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmpe_dim + 2\l__cal_tmpe_dim)/3}
6439         \tl_reverse:N \l__cal_tmpa_tl
6440         \prg_replicate:nn {3}
6441         {
6442             \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
6443         }
6444         \tl_reverse:N \l__cal_tmpa_tl
6445     }
6446     \c_spath_curveto_tl
6447     {
6448         \bool_set_true:N \l__cal_taperable_bool
6449         \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-5}}
6450         \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-4}}
6451         \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-8}}
6452         \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-7}}

```

```

6453     \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {-11}}
6454     \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {-10}}
6455     \tl_reverse:N \l__cal_tmpa_tl
6456     \prg_replicate:nn {9}
6457     {
6458       \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
6459     }
6460     \tl_reverse:N \l__cal_tmpa_tl
6461   }
6462 }
6463 {
6464   \bool_set_false:N \l__cal_taperable_bool
6465 }
6466
6467 \bool_if:NT \l__cal_taperable_bool
6468 {
6469   \__cal_taper_aux:
6470 }
6471
6472 }
6473
6474 \pgfsyssoftpath@setcurrentpath\l__cal_tmpa_tl
6475 \pgfsetstrokecolor{\pgfkeysvalueof{/tikz/pen~colour}}
6476 \pgfusepath{stroke}
6477
6478 }

```

(End definition for \cal_maybe_taper:N.)

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

```

6479 \cs_new_protected_nopar:Npn \__cal_taper_aux:
6480 {
6481   \tl_clear:N \l__cal_tmph_tl
6482   \tl_put_right:NV \l__cal_tmph_tl \c_spath_moveto_tl
6483
6484   \fp_set:Nn \l__cal_tmpha_fp
6485   {
6486     \l__cal_tmph_dim - \l__cal_tmph_dim
6487   }
6488   \fp_set:Nn \l__cal_tmphb_fp
6489   {
6490     \l__cal_tmpha_dim - \l__cal_tmphc_dim
6491   }
6492   \fp_set:Nn \l__cal_tmpe_fp
6493   {
6494      $(\l__cal_tmpha\_fp^2 + \l__cal_tmphb\_fp^2)^{.5}$ 
6495   }
6496
6497   \fp_set:Nn \l__cal_tmpha_fp
6498   {
6499     .5*\l__cal_taper_width_dim
6500     *
6501     \l__cal_tmpha_fp / \l__cal_tmpe_fp
6502   }

```

```

6503 \fp_set:Nn \l__cal_tmpb_fp
6504 {
6505   .5*\l__cal_taper_width_dim
6506   *
6507   \l__cal_tmpb_fp / \l__cal_tmpe_fp
6508 }
6509
6510 \fp_set:Nn \l__cal_tmpe_fp
6511 {
6512   \l__cal_tmph_dim - \l__cal_tmpe_dim
6513 }
6514 \fp_set:Nn \l__cal_tmpe_dim
6515 {
6516   \l__cal_tmpe_dim - \l__cal_tmpe_dim
6517 }
6518 \fp_set:Nn \l__cal_tmpe_dim
6519 {
6520   (\l__cal_tmpe_dim^2 + \l__cal_tmpe_dim)^.5
6521 }
6522
6523 \fp_set:Nn \l__cal_tmpe_dim
6524 {
6525   .5*\l__cal_line_width_dim
6526   *
6527   \l__cal_tmpe_dim / \l__cal_tmpe_dim
6528 }
6529 \fp_set:Nn \l__cal_tmpe_dim
6530 {
6531   .5*\l__cal_line_width_dim
6532   *
6533   \l__cal_tmpe_dim / \l__cal_tmpe_dim
6534 }
6535
6536 \tl_put_right:Nx \l__cal_tmpe_dim
6537 {
6538   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
6539   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
6540 }
6541
6542 \tl_put_right:NV \l__cal_tmpe_dim \c_spath_curvetoa_tl
6543
6544 \tl_put_right:Nx \l__cal_tmpe_dim
6545 {
6546   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
6547   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
6548 }
6549
6550 \tl_put_right:NV \l__cal_tmpe_dim \c_spath_curvetob_tl
6551
6552 \tl_put_right:Nx \l__cal_tmpe_dim
6553 {
6554   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
6555   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim}}
6556 }

```

```

6557
6558 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curveto_tl
6559
6560 \tl_put_right:Nx \l__cal_tmpb_tl
6561 {
6562   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim}}
6563   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim}}
6564 }
6565
6566 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetoa_tl
6567
6568 \tl_put_right:Nx \l__cal_tmpb_tl
6569 {
6570   {
6571     \dim_eval:n
6572     {
6573       \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim
6574       - \fp_to_dim:n{ 1.32 * \l__cal_tmpe_fp
6575     }
6576   }
6577 }
6578 {
6579   \dim_eval:n
6580   {
6581     \fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim
6582     + \fp_to_dim:n {1.32* \l__cal_tmpe_fp
6583   }
6584 }
6585 }
6586 }
6587
6588 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetob_tl
6589
6590 \tl_put_right:Nx \l__cal_tmpb_tl
6591 {
6592   {
6593     \dim_eval:n
6594     {
6595       -\fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim
6596       - \fp_to_dim:n {1.32 * \l__cal_tmpe_fp
6597     }
6598   }
6599 }
6600 {
6601   \dim_eval:n
6602   {
6603     -\fp_to_dim:N \l__cal_tmpe_fp + \l__cal_tmpe_dim
6604     + \fp_to_dim:n {1.32 * \l__cal_tmpe_fp
6605   }
6606 }
6607 }
6608 }
6609
6610 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curveto_tl

```

```

6611 \tl_put_right:Nx \l__cal_tmpb_tl
6612 {
6613   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim}}
6614   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim}}
6615 }
6616
6617 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetoa_tl
6618
6619 \tl_put_right:Nx \l__cal_tmpb_tl
6620 {
6621   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim}}
6622   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim}}
6623 }
6624
6625 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetob_tl
6626
6627 \tl_put_right:Nx \l__cal_tmpb_tl
6628 {
6629   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim}}
6630   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim}}
6631 }
6632
6633 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetoa_tl
6634
6635 \tl_put_right:Nx \l__cal_tmpb_tl
6636 {
6637   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim}}
6638   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim}}
6639 }
6640
6641 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetoa_tl
6642
6643 \tl_put_right:Nx \l__cal_tmpb_tl
6644 {
6645   {
6646     \dim_eval:n
6647     {
6648       -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim
6649       + \fp_to_dim:n{ 1.32 * \l__cal_tmpe_dim}
6650     }
6651   }
6652   {
6653     \dim_eval:n
6654     {
6655       -\fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim
6656       - \fp_to_dim:n {1.32* \l__cal_tmpe_dim}
6657     }
6658   }
6659 }
6660
6661 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetob_tl
6662
6663 \tl_put_right:Nx \l__cal_tmpb_tl
6664

```

```

6665 {
6666   {
6667     \dim_eval:n
6668     {
6669       \fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim
6670       + \fp_to_dim:n {1.32 * \l__cal_tmpb_fp}
6671     }
6672   }
6673   {
6674     \dim_eval:n
6675     {
6676       \fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim
6677       - \fp_to_dim:n {1.32 * \l__cal_tmpa_fp}
6678     }
6679   }
6680 }
6681
6682 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curveto_tl
6683
6684 \tl_put_right:Nx \l__cal_tmpb_tl
6685 {
6686   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim}}
6687   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim}}
6688 }
6689
6690 \pgfsyssoftpath@setcurrentpath\l__cal_tmpb_tl
6691 \pgfsetfillcolor{\pgfkeysvalueof{/tikz/pen~colour}}
6692 \pgfusepath{fill}
6693 }

```

(End definition for _cal_taper_aux:.)

Defines a copperplate pen.

```

6694 \tl_set:Nn \l__cal_tmpa_tl {\pgfsyssoftpath@movetotoken{Opt}{Opt}}
6695 \spath_components_to_seq:NV \l__cal_tmpa_seq \l__cal_tmpa_tl
6696 \seq_gclear_new:N \g__cal_pen_copperplate_seq
6697 \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.

```

6698 \DeclareDocumentCommand \CopperplatePath { m }
6699 {
6700   \spath_components_to_seq:NV \l__cal_tmpa_seq #1
6701   \cal_path_create:NN \l__cal_tmpa_seq \g__cal_pen_copperplate_seq
6702 }

```

(End definition for \CopperplatePath.)

```

6703 \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.

```

6704 \expandafter\ifx\csname pgfdeclaredecoration\endcsname\relax

```

```

6705 \else
6706 \pgfdeclaredecoration{calligraphic brace}{brace}%
6707 {%
6708   \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]%
6709   {%
6710     \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}%
6711     \pgfpathmoveto{\pgfpointorigin}%
6712     \pgfpathcurveto%
6713     {%
6714       \pgfqpoint%
6715       {.15\pgfdecorationsegmentamplitude}%
6716       {.3\pgfdecorationsegmentamplitude}%
6717     }%
6718     {%
6719       \pgfqpoint%
6720       {.5\pgfdecorationsegmentamplitude}%
6721       {.5\pgfdecorationsegmentamplitude}%
6722     }%
6723     {%
6724       \pgfqpoint%
6725       {\pgfdecorationsegmentamplitude}%
6726       {.5\pgfdecorationsegmentamplitude}%
6727     }%
6728     {%
6729       \pgftransformxshift%
6730       {+\pgfdecorationsegmentaspect\pgfdecoratedremainingdistance}%
6731       \pgfpathlineto%
6732       {%
6733         \pgfqpoint%
6734         {-\pgfdecorationsegmentamplitude}%
6735         {.5\pgfdecorationsegmentamplitude}%
6736       }%
6737       \pgfpathcurveto
6738       {%
6739         \pgfqpoint%
6740         {- .5\pgfdecorationsegmentamplitude}%
6741         {.5\pgfdecorationsegmentamplitude}%
6742       }%
6743       {%
6744         \pgfqpoint%
6745         {- .15\pgfdecorationsegmentamplitude}%
6746         {.7\pgfdecorationsegmentamplitude}%
6747       }%
6748       {%
6749         \pgfqpoint%
6750         {0\pgfdecorationsegmentamplitude}%
6751         {1\pgfdecorationsegmentamplitude}%
6752       }%
6753       \pgfpathmoveto%
6754       {%
6755         \pgfqpoint%
6756         {0\pgfdecorationsegmentamplitude}%
6757         {1\pgfdecorationsegmentamplitude}%
6758       }%

```



```

6759 \pgfpathcurveto%
6760 {%
6761 \pgfqpoint%
6762 {.15\pgfdecorationsegmentamplitude}%
6763 {.7\pgfdecorationsegmentamplitude}%
6764 }%
6765 {%
6766 \pgfqpoint%
6767 {.5\pgfdecorationsegmentamplitude}%
6768 {.5\pgfdecorationsegmentamplitude}%
6769 }%
6770 {%
6771 \pgfqpoint%
6772 {\pgfdecorationsegmentamplitude}%
6773 {.5\pgfdecorationsegmentamplitude}%
6774 }%
6775 }%
6776 {%
6777 \pgftransformxshift{+\pgfdecoratedremainingdistance}%
6778 \pgfpathlineto%
6779 {%
6780 \pgfqpoint%
6781 {-\pgfdecorationsegmentamplitude}%
6782 {.5\pgfdecorationsegmentamplitude}%
6783 }%
6784 \pgfpathcurveto%
6785 {%
6786 \pgfqpoint%
6787 {-.5\pgfdecorationsegmentamplitude}%
6788 {.5\pgfdecorationsegmentamplitude}%
6789 }%
6790 {%
6791 \pgfqpoint%
6792 {-.15\pgfdecorationsegmentamplitude}%
6793 {.3\pgfdecorationsegmentamplitude}%
6794 }%
6795 {\pgfqpoint{Opt}{Opt}}%
6796 }%
6797 \tikzset{%
6798 taper width=.5\pgflinewidth,%
6799 taper%
6800 }%
6801 \pgfsyssoftpath@getcurrentpath\cal@tmp@path%
6802 \CopperplatePath{\cal@tmp@path}%
6803 }%
6804 \state{final}{}%
6805 }%

```

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

```

6806 \pgfdeclaredecoration{calligraphic straight parenthesis}{brace}
6807 {
6808 \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]%
6809 {%

```

```

6810 \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}%
6811 \pgfpathmoveto{\pgfpointorigin}%
6812 \pgfpathcurveto%
6813 {%
6814   \pgfqpoint%
6815   {.76604\pgfdecorationsegmentamplitude}%
6816   {.64279\pgfdecorationsegmentamplitude}%
6817   }%
6818 {%
6819   \pgfqpoint%
6820   {2.3333\pgfdecorationsegmentamplitude}%
6821   {\pgfdecorationsegmentamplitude}%
6822   }%
6823 {%
6824   \pgfqpoint%
6825   {3.3333\pgfdecorationsegmentamplitude}%
6826   {\pgfdecorationsegmentamplitude}%
6827   }%
6828 {%
6829   \pgftransformxshift{+\pgfdecoratedremainingdistance}%
6830   \pgfpathlineto%
6831   {%
6832     \pgfqpoint%
6833     {-3.3333\pgfdecorationsegmentamplitude}%
6834     {\pgfdecorationsegmentamplitude}%
6835     }%
6836   \pgfpathcurveto%
6837   {%
6838     \pgfqpoint%
6839     {-2.3333\pgfdecorationsegmentamplitude}%
6840     {\pgfdecorationsegmentamplitude}%
6841     }%
6842   {%
6843     \pgfqpoint%
6844     {-.76604\pgfdecorationsegmentamplitude}%
6845     {.64279\pgfdecorationsegmentamplitude}%
6846     }%
6847     {\pgfqpoint{Opt}{Opt}}%
6848   }%
6849   \tikzset{%
6850     taper width=.5\pgflinewidth,%
6851     taper%
6852   }%
6853   \pgfsyssoftpath@getcurrentpath\cal@tmp@path%
6854   \CopperplatePath{\cal@tmp@path}%
6855   }%
6856   \state{final}{}%
6857 }

```

The third is a curved parenthesis.

```

6858 \pgfdeclaredecoration{calligraphic curved parenthesis}{brace}
6859 {
6860   \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]%
6861   {%
6862     \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}%

```

```

6863 \pgfpathmoveto{\pgfpointorigin}%
6864 \pgf@xa=\pgfdecoratedremainingdistance\relax%
6865 \advance\pgf@xa by -1.5890\pgfdecorationsegmentamplitude\relax%
6866 \edef\cgrphy@xa{\the\pgf@xa}%
6867 \pgfpathcurveto%
6868 {%
6869 \pgfqpoint%
6870 {1.5890\pgfdecorationsegmentamplitude}%
6871 {1.3333\pgfdecorationsegmentamplitude}%
6872 }%
6873 {\pgfqpoint{\cgrphy@xa}{1.3333\pgfdecorationsegmentamplitude}}%
6874 {\pgfqpoint{\pgfdecoratedremainingdistance}{0pt}}%
6875 \tikzset{%
6876 taper width=.5\pgflinewidth,%
6877 taper%
6878 }%
6879 \pgfsyssoftpath@getcurrentpath\cal@tmp@path%
6880 \CopperplatePath{\cal@tmp@path}%
6881 }%
6882 \state{final}{}%
6883 }

```

End the conditional for if pgfdecoration module is loaded

```

6884 \fi

```

6 Drawing Knots

```

6885 <@=knot>

```

6.1 Initialisation

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

```

6886 \RequirePackage{spath3}
6887 \usetikzlibrary{intersections,spath3}
6888
6889 \ExplSyntaxOn
6890
6891 \tl_new:N \l__knot_tmpa_tl
6892 \tl_new:N \l__knot_tmpb_tl
6893 \tl_new:N \l__knot_tmpc_tl
6894 \tl_new:N \l__knot_tmpd_tl
6895 \tl_new:N \l__knot_tmpg_tl
6896 \tl_new:N \l__knot_redraws_tl
6897 \tl_new:N \l__knot_clip_width_tl
6898 \tl_new:N \l__knot_name_tl
6899 \tl_new:N \l__knot_node_tl
6900 \tl_new:N \l__knot_aux_tl
6901 \tl_new:N \l__knot_auxa_tl
6902 \tl_new:N \l__knot_prefix_tl
6903
6904 \seq_new:N \l__knot_segments_seq
6905
6906 \int_new:N \l__knot_tmpa_int
6907 \int_new:N \l__knot_strands_int

```

```

6908 \int_new:N \g__knot_intersections_int
6909 \int_new:N \g__knot_filaments_int
6910 \int_new:N \l__knot_component_start_int
6911
6912 \fp_new:N \l__knot_tmpa_fp
6913 \fp_new:N \l__knot_tmpb_fp
6914
6915 \dim_new:N \l__knot_tmpa_dim
6916 \dim_new:N \l__knot_tmpb_dim
6917 \dim_new:N \l__knot_tolerance_dim
6918 \dim_new:N \l__knot_redraw_tolerance_dim
6919 \dim_new:N \l__knot_clip_bg_radius_dim
6920 \dim_new:N \l__knot_clip_draw_radius_dim
6921
6922 \bool_new:N \l__knot_draft_bool
6923 \bool_new:N \l__knot_ignore_ends_bool
6924 \bool_new:N \l__knot_self_intersections_bool
6925 \bool_new:N \l__knot_splits_bool
6926 \bool_new:N \l__knot_super_draft_bool
6927
6928 \bool_new:N \l__knot_prepend_prev_bool
6929 \bool_new:N \l__knot_append_next_bool
6930 \bool_new:N \l__knot_skip_bool
6931 \bool_new:N \l__knot_save_bool
6932
6933 \seq_new:N \g__knot_nodes_seq
6934
6935 \bool_set_true:N \l__knot_ignore_ends_bool
Configuration is via TikZ keys and styles.
6936 \tikzset{
6937   spath/prefix/knot/.style={
6938     spath/set~ prefix=knot strand,
6939   },
6940   spath/suffix/knot/.style={
6941     spath/set~ suffix={},
6942   },
6943   knot/.code={
6944     \tl_if_eq:nnTF {#1} {none}
6945     {
6946       \tikz@addmode{\tikz@mode@doublefalse}
6947     }
6948     {
6949       \tikz@addmode{\tikz@mode@doubletrue}
6950       \tl_if_eq:nnTF {\pgfkeysnovalue} {#1}
6951       {
6952         \tikz@adoption{\pgfsetinnerstrokecolor{.}}
6953       }
6954       {
6955         \pgfsetinnerstrokecolor{#1}
6956       }
6957       \tikz@adoption{
6958         \pgfsetstrokecolor{knotbg}
6959       }
6960       \tl_set:Nn \tikz@double@setup{

```

```

6961         \pgfsetinnerlinewidth{\pgflinewidth}
6962         \pgfsetlinewidth{\dim_eval:n {\tl_use:N \l__knot_gap_tl \pgflinewidth}}
6963     }
6964 }
6965 },
6966 knot~ gap/.store~ in=\l__knot_gap_tl,
6967 knot~ gap=3,
6968 knot~ diagram/.is~family,
6969 knot~ diagram/.unknown/.code={
6970     \tl_set_eq:NN \l__knot_tmpa_tl \pgfkeyscurrentname
6971     \pgfkeysalso{
6972         /tikz/\l__knot_tmpa_tl=#1
6973     }
6974 },
6975 background~ colour/.code={%
6976     \colorlet{knotbg}{#1}%
6977 },
6978 background~ color/.code={%
6979     \colorlet{knotbg}{#1}%
6980 },
6981 background~ colour=white,
6982 knot~ diagram,
6983 name/.store~ in=\l__knot_name_tl,
6984 name={knot},
6985 save~ intersections/.is~ choice,
6986 save~ intersections/.default=true,
6987 save~ intersections/true/.code={
6988     \bool_set_true:N \l__knot_save_bool
6989 },
6990 save~ intersections/false/.code={
6991     \bool_set_false:N \l__knot_save_bool
6992 },
6993 every~ strand/.style={draw},
6994 ignore~ endpoint~ intersections/.code={
6995     \tl_if_eq:nnTF {#1} {true}
6996     {
6997         \bool_set_true:N \l__knot_ignore_ends_bool
6998     }
6999     {
7000         \bool_set_false:N \l__knot_ignore_ends_bool
7001     }
7002 },
7003 ignore~ endpoint~ intersections/.default=true,
7004 consider~ self~ intersections/.is~choice,
7005 consider~ self~ intersections/true/.code={
7006     \bool_set_true:N \l__knot_self_intersections_bool
7007     \bool_set_true:N \l__knot_splits_bool
7008 },
7009 consider~ self~ intersections/false/.code={
7010     \bool_set_false:N \l__knot_self_intersections_bool
7011     \bool_set_false:N \l__knot_splits_bool
7012 },
7013 consider~ self~ intersections/no~ splits/.code={
7014     \bool_set_true:N \l__knot_self_intersections_bool

```

```

7015     \bool_set_false:N \l__knot_splits_bool
7016   },
7017   consider~ self~ intersections/.default={true},
7018   clip~ radius/.code={
7019     \dim_set:Nn \l__knot_clip_bg_radius_dim {#1}
7020     \dim_set:Nn \l__knot_clip_draw_radius_dim {#1+2pt}
7021   },
7022   clip~ draw~ radius/.code={
7023     \dim_set:Nn \l__knot_clip_draw_radius_dim {#1}
7024   },
7025   clip~ background~ radius/.code={
7026     \dim_set:Nn \l__knot_clip_bg_radius_dim {#1}
7027   },
7028   clip~ radius=10pt,
7029   end~ tolerance/.code={
7030     \dim_set:Nn \l__knot_tolerance_dim {#1}
7031   },
7032   end~ tolerance=14pt,
7033   clip/.style={
7034     clip
7035   },
7036   background~ clip/.style={
7037     clip
7038   },
7039   clip~ width/.code={
7040     \tl_set:Nn \l__knot_clip_width_tl {#1}
7041   },
7042   clip~ width=3,
7043   flip~ crossing/.code={%
7044     \tl_clear_new:c {l__knot_crossing_#1}
7045     \tl_set:cn {l__knot_crossing_#1} {x}
7046   },
7047   ignore~ crossing/.code={%
7048     \tl_clear_new:c {l__knot_ignore_crossing_#1}
7049     \tl_set:cn {l__knot_ignore_crossing_#1} {x}
7050   },
7051   draft~ mode/.is~ choice,
7052   draft~ mode/off/.code={%
7053     \bool_set_false:N \l__knot_draft_bool
7054     \bool_set_false:N \l__knot_super_draft_bool
7055   },
7056   draft~ mode/crossings/.code={%
7057     \bool_set_true:N \l__knot_draft_bool
7058     \bool_set_false:N \l__knot_super_draft_bool
7059   },
7060   draft~ mode/strands/.code={%
7061     \bool_set_true:N \l__knot_draft_bool
7062     \bool_set_true:N \l__knot_super_draft_bool
7063   },
7064   draft/.is~ family,
7065   draft,
7066   crossing~ label/.style={
7067     overlay,
7068     fill=white,

```

```

7069     fill~ opacity=.5,
7070     text~ opacity=1,
7071     text=blue,
7072     pin~ edge={blue,<-}
7073   },
7074   strand~ label/.style={
7075     overlay,
7076     circle,
7077     draw=purple,
7078     fill=white,
7079     fill~ opacity=.5,
7080     text~ opacity=1,
7081     text=purple,
7082     inner~ sep=0pt
7083   },
7084 }

```

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

```

7085 \cs_new_nopar:Npn \knot_apply_style:N #1
7086 {
7087   \tl_if_exist:NT #1 {
7088     \exp_args:NV \tikzset #1
7089   }
7090 }
7091 \cs_generate_variant:Nn \knot_apply_style:N {c}

```

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

```

7092 \NewDocumentCommand \flipcrossings {m}
7093 {
7094   \tikzset{knot~ diagram/flip~ crossing/.list={#1}}%
7095 }

```

(End definition for `\flipcrossings`.)

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

```

7096 \NewDocumentCommand \strand { 0{} }
7097 {
7098   \int_incr:N \l__knot_strands_int
7099   \tl_clear_new:c {l__knot_options_strand \int_use:N \l__knot_strands_int}
7100   \tl_set:cn {l__knot_options_strand \int_use:N \l__knot_strands_int} {#1}
7101   \path[#1,spath/set~ name=knot,spath/save=\int_use:N \l__knot_strands_int]
7102 }

```

(End definition for `\strand`.)

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

```

7103 \NewDocumentEnvironment{knot} { 0{} }
7104 {
7105   \knot_initialise:n {#1}
7106 }
7107 {
7108   \knot_render:
7109 }

```

(End definition for knot.)

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

```
7110 \cs_new_protected_nopar:Npn \knot_initialise:n #1
7111 {
7112   \tikzset{knot~ diagram/.cd,every~ knot~ diagram/.try,#1}
7113   \int_zero:N \l__knot_strands_int
7114   \tl_clear:N \l__knot_redraws_tl
7115   \seq_gclear:N \g__knot_nodes_seq
7116 }
```

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

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

```
7117 \cs_new_protected_nopar:Npn \knot_render:
7118 {
```

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

```
7119   \pgfscope
7120   \pgftransformreset
```

Set the dimension for deciding when to include neighbouring strands

```
7121   \dim_set:Nn \l__knot_redraw_tolerance_dim {\fp_to_dim:n
7122     {
7123       sqrt(2) * max(\l__knot_clip_bg_radius_dim, \l__knot_clip_draw_radius_dim)
7124     }
7125 }
```

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

```
7126   \int_step_function:nnnN {1} {1} {\l__knot_strands_int} \knot_draw_strand:n
```

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

```
7127   \bool_if:NF \l__knot_super_draft_bool
7128   {
```

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.

```
7129     \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.

```
7130     \bool_if:NTF \l__knot_self_intersections_bool
7131     {
7132       \knot_split_strands:
7133       \int_set_eq:NN \l__knot_tmpa_int \g__knot_filaments_int
7134       \tl_set:Nn \l__knot_prefix_tl {filament}
7135     }
7136     {
7137       \int_set_eq:NN \l__knot_tmpa_int \l__knot_strands_int
7138       \tl_set:Nn \l__knot_prefix_tl {strand}
7139     }
```

Initialise the intersection count.

```
7140     \int_gzero:N \g__knot_intersections_int
```


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

```

7141 \tl_clear:N \l__knot_node_tl
7142 \bool_if:NT \l__knot_draft_bool
7143 {
7144   \tl_set:Nn \l__knot_node_tl {
7145     \exp_not:N \node[coordinate,
7146       pin={[
7147         node~ contents={\int_use:N \g__knot_intersections_int},
7148         knot~ diagram/draft/crossing~ label,
7149         knot~ diagram/draft/crossing~
7150         \int_use:N \g__knot_intersections_int \c_space_tl label/.try
7151       ]
7152     }]
7153   }
7154 }

```

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

```

7155 \int_step_variable:nnnNn {1} {1} {\l__knot_tmpa_int - 1} \l__knot_tmpa_tl
7156 {
7157   \int_step_variable:nnnNn
7158   {\tl_use:N \l__knot_tmpa_tl + 1}
7159   {1}
7160   {\l__knot_tmpa_int} \l__knot_tmpb_tl
7161   {
7162     \knot_intersections:VV \l__knot_tmpa_tl \l__knot_tmpb_tl
7163   }
7164 }

```

If any redraws were requested, do them here.

```

7165 \tl_use:N \l__knot_redraws_tl

```

Draw the crossing nodes

```

7166 \seq_use:Nn \g__knot_nodes_seq {}
7167 }

```

Close the scope

```

7168 \endpgfscope
7169 }

```

(End definition for \knot_render:.)

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

```

7170 \cs_new_protected_nopar:Npn \knot_draw_strand:n #1
7171 {
7172   \pgfscope
7173   \group_begin:
7174   \spath_bake_round:c {knot strand #1}
7175   \tl_set:Nn \l__knot_tmpa_tl {knot~ diagram/every~ strand/.try,}
7176   \tl_put_right:Nv \l__knot_tmpa_tl {\l__knot_options_strand #1}
7177   \tl_put_right:Nn \l__knot_tmpa_tl
7178   {
7179     ,
7180     knot~ diagram/only~ when~ rendering/.try,

```

```

7181     only~ when~ rendering/.try
7182   }
7183   \spath_tikz_path:Vv \l__knot_tmpa_tl {knot strand #1}
7184   \group_end:
7185   \endpgfscope
7186 }
7187 \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.

```

7188 \cs_new_protected_nopar:Npn \knot_draw_labels:n #1
7189 {
7190   \bool_if:NT \l__knot_draft_bool
7191   {
7192     \spath_finalpoint:Nv \l__knot_tmpb_tl {knot strand #1}
7193     \dim_set:Nn \l__knot_tmpa_dim {\tl_item:Nn \l__knot_tmpb_tl {1}}
7194     \dim_set:Nn \l__knot_tmpb_dim {\tl_item:Nn \l__knot_tmpb_tl {2}}
7195     \node[
7196       knot~ diagram/draft/strand~label
7197     ] at (\l__knot_tmpa_dim,\l__knot_tmpb_dim) {#1};
7198     \spath_initialpoint:Nv \l__knot_tmpb_tl {knot strand #1}
7199     \dim_set:Nn \l__knot_tmpa_dim {\tl_item:Nn \l__knot_tmpb_tl {1}}
7200     \dim_set:Nn \l__knot_tmpb_dim {\tl_item:Nn \l__knot_tmpb_tl {2}}
7201     \node[
7202       knot~ diagram/draft/strand~label
7203     ] at (\l__knot_tmpa_dim,\l__knot_tmpb_dim) {#1};
7204   }
7205   \bool_if:nT {
7206     \l__knot_self_intersections_bool
7207     &&
7208     \l__knot_splits_bool
7209   }
7210   {
7211     \tl_clear:N \l__knot_tmpa_tl
7212     \spath_initialpoint:Nv \l__knot_tmpa_tl {knot strand #1}
7213     \tl_put_left:Nv \l__knot_tmpa_tl \c_spath_moveto_tl
7214     \spath_segments_to_seq:Nv \l__knot_segments_seq {knot strand #1}
7215     \seq_map_function:NN \l__knot_segments_seq \knot_split_self_intersects:N
7216     \tl_set_eq:cN {knot strand #1} \l__knot_tmpa_tl
7217   }
7218 }

```

(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!

```

7219 \cs_new_protected_nopar:Npn \knot_split_self_intersects:N #1
7220 {
7221   \tl_set:Nx \l__knot_tmpc_tl {\tl_item:nn {#1} {4}}
7222   \tl_case:NnF \l__knot_tmpc_tl

```

```

7223 {
7224   \c_spath_curvetoa_tl
7225   {
7226     \fp_set:Nn \l__knot_tmpa_fp
7227     {
7228       (\tl_item:nn {#1} {3} - 3 * \tl_item:nn {#1} {6}
7229       + 3 * \tl_item:nn {#1} {9} - \tl_item:nn {#1} {12})
7230       *
7231       (3 * \tl_item:nn {#1} {8} - 3 * \tl_item:nn {#1} {11})
7232       -
7233       (\tl_item:nn {#1} {2} - 3 * \tl_item:nn {#1} {5}
7234       + 3 * \tl_item:nn {#1} {8} - \tl_item:nn {#1} {11})
7235       *
7236       (3 * \tl_item:nn {#1} {9} - 3 * \tl_item:nn {#1} {12})
7237     }
7238     \fp_set:Nn \l__knot_tmpb_fp
7239     {
7240       (\tl_item:nn {#1} {2} - 3 * \tl_item:nn {#1} {5}
7241       + 3 * \tl_item:nn {#1} {8} - \tl_item:nn {#1} {11})
7242       *
7243       (3 * \tl_item:nn {#1} {6} - 6 * \tl_item:nn {#1} {9}
7244       + 3 * \tl_item:nn {#1} {12})
7245       -
7246       (\tl_item:nn {#1} {3} - 3 * \tl_item:nn {#1} {6}
7247       + 3 * \tl_item:nn {#1} {9} - \tl_item:nn {#1} {12})
7248       *
7249       (3 * \tl_item:nn {#1} {5} - 6 * \tl_item:nn {#1} {8}
7250       + 3 * \tl_item:nn {#1} {11})
7251     }
7252     \fp_compare:nTF
7253     {
7254       \l__knot_tmpb_fp != 0
7255     }
7256     {
7257       \fp_set:Nn \l__knot_tmpa_fp { .5 * \l__knot_tmpa_fp / \l__knot_tmpb_fp }
7258       \fp_compare:nTF
7259       {
7260         0 < \l__knot_tmpa_fp && \l__knot_tmpa_fp < 1
7261       }
7262       {
7263         \spath_split_curve:NNnV
7264         \l__knot_tmpc_tl
7265         \l__knot_tmpd_tl
7266         {#1}
7267         \l__knot_tmpa_fp
7268         \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
7269         \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
7270         \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
7271         \tl_set:Nx \l__knot_tmpd_tl {\tl_tail:N \l__knot_tmpd_tl}
7272         \tl_set:Nx \l__knot_tmpd_tl {\tl_tail:N \l__knot_tmpd_tl}
7273         \tl_set:Nx \l__knot_tmpd_tl {\tl_tail:N \l__knot_tmpd_tl}
7274         \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpc_tl
7275         \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpd_tl
7276       }

```

```

7277     {
7278         \tl_set:Nn \l__knot_tmpc_tl {#1}
7279         \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
7280         \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
7281         \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
7282         \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpc_tl
7283     }
7284 }
7285 {
7286     \tl_set:Nn \l__knot_tmpc_tl {#1}
7287     \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
7288     \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
7289     \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
7290     \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpc_tl
7291 }
7292 }
7293 \c_spath_lineto_tl
7294 {
7295     \tl_set:Nn \l__knot_tmpc_tl {#1}
7296     \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
7297     \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
7298     \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
7299     \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpc_tl
7300 }
7301 }
7302 {
7303     \tl_put_right:Nn \l__knot_tmpa_tl {#1}
7304 }
7305 }

```

(End definition for `\knot_split_self_intersects:N`.)

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

```

7306 \cs_new_protected_nopar:Npn \knot_intersections:nm #1#2
7307 {
7308     \group_begin:
7309     \tl_set_eq:NN \l__knot_tmpa_tl \l__knot_prefix_tl
7310     \tl_put_right:Nn \l__knot_tmpa_tl {#1}
7311     \tl_set_eq:NN \l__knot_tmpb_tl \l__knot_prefix_tl
7312     \tl_put_right:Nn \l__knot_tmpb_tl {#2}
7313     \tl_set_eq:Nc \l__knot_tmpc_tl {knot \tl_use:N \l__knot_tmpa_tl}
7314     \tl_set_eq:Nc \l__knot_tmpd_tl {knot \tl_use:N \l__knot_tmpb_tl}
7315
7316     \bool_if:nTF {
7317         \l__knot_save_bool
7318         &&
7319         \tl_if_exist_p:c {
7320             knot~ intersections~
7321             \tl_use:N \l__knot_name_tl -
7322             \tl_use:N \l__knot_tmpa_tl -
7323             \tl_use:N \l__knot_tmpb_tl
7324         }
7325     }
7326     {

```

```

7327 \tl_use:c
7328 {
7329   knot~ intersections~ \tl_use:N \l__knot_name_tl -
7330   \tl_use:N \l__knot_tmpa_tl -
7331   \tl_use:N \l__knot_tmpb_tl
7332 }
7333 }
7334 {
7335   \pgfintersectionofpaths{\pgfsetpath\l__knot_tmpc_tl}{\pgfsetpath\l__knot_tmpd_tl}
7336 }
7337 }
7338
7339 \int_compare:nT {\pgfintersectionsolutions > 0}
7340 {
7341   \int_step_function:nnnN
7342   {1}
7343   {1}
7344   {\pgfintersectionsolutions}
7345   \knot_do_intersection:n
7346 }
7347
7348 \knot_save_intersections:VV \l__knot_tmpa_tl \l__knot_tmpb_tl
7349 \group_end:
7350 }

```

(End definition for \knot_intersections:nn.)

\knot_save_intersections:nn

```

7351 \cs_new_protected_nopar:Npn \knot_save_intersections:nn #1#2
7352 {
7353   \bool_if:NT \l__knot_save_bool
7354   {
7355     \tl_clear:N \l__knot_aux_tl
7356     \tl_put_right:Nn \l__knot_aux_tl
7357     {
7358       \def\pgfintersectionsolutions
7359     }
7360     \tl_put_right:Nx \l__knot_aux_tl
7361     {
7362       {\int_eval:n {\pgfintersectionsolutions}}
7363     }
7364     \int_compare:nT {\pgfintersectionsolutions > 0}
7365     {
7366       \int_step_inline:nnnn {1} {1} {\pgfintersectionsolutions}
7367       {
7368         \pgfpointintersectionsolution{##1}
7369         \dim_set:Nn \l__knot_tmpa_dim {\pgf@x}
7370         \dim_set:Nn \l__knot_tmpb_dim {\pgf@y}
7371         \tl_put_right:Nn \l__knot_aux_tl
7372         {
7373           \expandafter\def\csname pgfpoint@intersect@solution@##1\endcsname
7374         }
7375         \tl_put_right:Nx \l__knot_aux_tl
7376         {

```

```

7377     {
7378         \exp_not:N \pgf@x
7379         =
7380         \dim_use:N \l__knot_tmpa_dim
7381         \exp_not:N \relax
7382         \exp_not:N \pgf@y
7383         =
7384         \dim_use:N \l__knot_tmpb_dim
7385         \exp_not:N \relax
7386     }
7387 }
7388 }
7389 \tl_set:Nn \l__knot_auxa_tl {\expandafter \gdef \csname knot~ intersections~}
7390 \tl_put_right:Nx \l__knot_auxa_tl {\tl_use:N \l__knot_name_tl - #1 - #2}
7391 \tl_put_right:Nn \l__knot_auxa_tl {\endcsname}
7392 \tl_put_right:Nx \l__knot_auxa_tl {\tl_to_str:N \l__knot_aux_tl}
7393 \protected@write\@auxout{}\tl_to_str:N \l__knot_auxa_tl}
7394 }
7395 }
7396 }
7397 \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.

```

7398 \cs_new_protected_nopar:Npn \knot_do_intersection:n #1
7399 {

```

Get the intersection coordinates.

```

7400 \pgfpointintersectionsolution{#1}
7401 \dim_set:Nn \l__knot_tmpa_dim {\pgf@x}
7402 \dim_set:Nn \l__knot_tmpb_dim {\pgf@y}

```

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

```

7403 \bool_set_false:N \l__knot_skip_bool
7404 \bool_if:NT \l__knot_self_intersections_bool
7405 {

```

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

```

7406 \tl_set:Nn \l__knot_tmpc_tl {knot previous}
7407 \tl_put_right:NV \l__knot_tmpc_tl \l__knot_tmpa_tl
7408 \tl_set:Nv \l__knot_tmpc_tl \l__knot_tmpc_tl
7409 \tl_if_eq:NNT \l__knot_tmpc_tl \l__knot_tmpb_tl
7410 {
7411     \knot_test_endpoint:NVT \l__knot_tolerance_dim \l__knot_tmpb_tl {final point}
7412     {
7413         \bool_set_true:N \l__knot_skip_bool
7414     }
7415 }
7416 }
7417 \tl_set:Nn \l__knot_tmpc_tl {knot previous}
7418 \tl_put_right:NV \l__knot_tmpc_tl \l__knot_tmpb_tl
7419 \tl_set:Nv \l__knot_tmpc_tl \l__knot_tmpc_tl
7420 \tl_if_eq:NNT \l__knot_tmpc_tl \l__knot_tmpa_tl

```

```

7421     {
7422         \knot_test_endpoint:NVnT \l__knot_tolerance_dim \l__knot_tmpa_tl {final point}
7423         {
7424             \bool_set_true:N \l__knot_skip_bool
7425         }
7426     }
7427 }

```

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

```

7428 \bool_if:NT \l__knot_ignore_ends_bool
7429 {
7430     \knot_test_endpoint:NVnT \l__knot_tolerance_dim \l__knot_tmpa_tl {initial point}
7431     {
7432         \bool_set_true:N \l__knot_skip_bool
7433     }
7434     \knot_test_endpoint:NVnT \l__knot_tolerance_dim \l__knot_tmpa_tl {final point}
7435     {
7436         \bool_set_true:N \l__knot_skip_bool
7437     }
7438     \knot_test_endpoint:NVnT \l__knot_tolerance_dim \l__knot_tmpb_tl {initial point}
7439     {
7440         \bool_set_true:N \l__knot_skip_bool
7441     }
7442     \knot_test_endpoint:NVnT \l__knot_tolerance_dim \l__knot_tmpb_tl {final point}
7443     {
7444         \bool_set_true:N \l__knot_skip_bool
7445     }
7446 }

```

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

```

7447 \bool_if:NF \l__knot_skip_bool
7448 {
7449
7450     \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.

```

7451 \bool_if:nF
7452 {
7453     \tl_if_exist_p:c {l__knot_ignore_crossing_ \int_use:N
7454         \g__knot_intersections_int}
7455     &&
7456     ! \tl_if_empty_p:c {l__knot_ignore_crossing_ \int_use:N
7457         \g__knot_intersections_int}
7458 }
7459 {

```

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

```

7460 \bool_if:nTF
7461 {
7462     \tl_if_exist_p:c {l__knot_crossing_ \int_use:N
7463         \g__knot_intersections_int}
7464     &&

```

```

7465     ! \tl_if_empty_p:c {l__knot_crossing_ \int_use:N
7466     \g__knot_intersections_int}
7467   }
7468   {
7469     \tl_set_eq:NN \l__knot_tmpg_tl \l__knot_tmpb_tl
7470   }
7471   {
7472     \tl_set_eq:NN \l__knot_tmpg_tl \l__knot_tmpa_tl
7473   }

```

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.

```

7474   \bool_if:NT \l__knot_self_intersections_bool
7475   {
7476     \knot_test_endpoint:NVnT
7477     \l__knot_redraw_tolerance_dim \l__knot_tmpg_tl {initial point}
7478     {
7479       \bool_set_true:N \l__knot_prepend_prev_bool
7480     }
7481     {
7482       \bool_set_false:N \l__knot_prepend_prev_bool
7483     }
7484     \knot_test_endpoint:NVnT
7485     \l__knot_redraw_tolerance_dim \l__knot_tmpg_tl {final point}
7486     {
7487       \bool_set_true:N \l__knot_append_next_bool
7488     }
7489     {
7490       \bool_set_false:N \l__knot_append_next_bool
7491     }

```

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

```

7492   \bool_if:nT
7493   {
7494     \l__knot_prepend_prev_bool || \l__knot_append_next_bool
7495   }
7496   {
7497     \tl_clear_new:c {knot \tl_use:N \l__knot_prefix_tl -1}
7498     \tl_set_eq:cc
7499     {knot \tl_use:N \l__knot_prefix_tl -1}
7500     {knot \tl_use:N \l__knot_tmpg_tl}
7501
7502     \tl_clear_new:c {l__knot_options_ \tl_use:N \l__knot_prefix_tl -1}
7503     \tl_set_eq:cc
7504     {l__knot_options_ \tl_use:N \l__knot_prefix_tl -1}
7505     {l__knot_options_ \tl_use:N \l__knot_tmpg_tl}
7506
7507     \bool_if:nT
7508     {
7509       \l__knot_prepend_prev_bool
7510       &&
7511       \tl_if_exist_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
7512       &&
7513       !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}

```



```

7514     }
7515     {
7516     \spath_prepend_no_move:cv
7517     {knot \tl_use:N \l__knot_prefix_tl -1}
7518     {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.

```

7519     \bool_if:nT
7520     {
7521     \l__knot_splits_bool
7522     &&
7523     \tl_if_exist_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
7524     &&
7525     !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
7526     }
7527     {
7528     \knot_test_endpoint:NvNT
7529     \l__knot_redraw_tolerance_dim
7530     {knot previous \tl_use:N \l__knot_tmpg_tl}
7531     {initial point}
7532     {
7533     \spath_prepend_no_move:cv
7534     {knot \tl_use:N \l__knot_prefix_tl -1}
7535     {knot \tl_use:c
7536     {knot previous \tl_use:c
7537     {knot previous \tl_use:N \l__knot_tmpg_tl}
7538     }
7539     }
7540     \tl_set_eq:Nc \l__knot_tmpa_tl {knot \tl_use:N \l__knot_prefix_tl -
1}
7541     }
7542     }
7543     }

```

Now the same for appending.

```

7544     \bool_if:nT
7545     {
7546     \l__knot_append_next_bool
7547     &&
7548     \tl_if_exist_p:c {knot next \tl_use:N \l__knot_tmpg_tl}
7549     &&
7550     !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
7551     }
7552     {
7553     \spath_append_no_move:cv
7554     {knot \tl_use:N \l__knot_prefix_tl -1}
7555     {knot \tl_use:c {knot next \tl_use:N \l__knot_tmpg_tl}}
7556     \bool_if:nT
7557     {
7558     \l__knot_splits_bool
7559     &&
7560     \tl_if_exist_p:c {knot previous \tl_use:N
7561     \l__knot_tmpg_tl}
7562     &&

```

```

7563         !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
7564     }
7565     {
7566         \knot_test_endpoint:NvnT
7567         \l__knot_redraw_tolerance_dim
7568         {knot previous \tl_use:N \l__knot_tmpg_tl}
7569         {final point}
7570         {
7571             \spath_append_no_move:cv
7572             {knot \tl_use:N \l__knot_prefix_tl -1}
7573             {knot \tl_use:c
7574                 {knot next \tl_use:c
7575                     {knot next \tl_use:N \l__knot_tmpg_tl}
7576                 }
7577             }
7578         }
7579     }
7580 }
7581 \tl_set:Nn \l__knot_tmpg_tl {\tl_use:N \l__knot_prefix_tl -1}
7582 }
7583 }

```

Now we render the crossing.

```

7584     \pgfscope
7585     \group_begin:
7586     \tikzset{
7587         knot~ diagram/every~ intersection/.try,
7588         every~ intersection/.try,
7589         knot~ diagram/intersection~ \int_use:N \g__knot_intersections_int/.try
7590     }
7591     \knot_draw_crossing:VVV \l__knot_tmpg_tl \l__knot_tmpa_dim \l__knot_tmpb_dim
7592     \coordinate
7593     (\l__knot_name_tl \c_space_tl \int_use:N \g__knot_intersections_int)
7594     at (\dim_use:N \l__knot_tmpa_dim, \dim_use:N \l__knot_tmpb_dim);
7595     \group_end:
7596     \endpgfscope

```

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

```

7597     }

```

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

```

7598     \tl_if_empty:NF \l__knot_node_tl
7599     {
7600         \seq_gpush:Nx
7601         \g__knot_nodes_seq
7602         {
7603             \l__knot_node_tl
7604             at
7605             (\dim_use:N \l__knot_tmpa_dim, \dim_use:N \l__knot_tmpb_dim) {};
7606         }
7607     }
7608 }
7609 }
7610
7611 \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.

```
7612 \prg_new_conditional:Npnn \knot_test_endpoint:NN #1#2 {p,T,F,TF}
7613 {
7614   \dim_compare:nTF
7615   {
7616     \dim_abs:n { \l__knot_tmpa_dim - \tl_item:Nn #2 {1}}
7617     +
7618     \dim_abs:n { \l__knot_tmpb_dim - \tl_item:Nn #2 {2}}
7619     <
7620     #1
7621   }
7622   {
7623     \prg_return_true:
7624   }
7625   {
7626     \prg_return_false:
7627   }
7628 }
```

(End definition for `\knot_test_endpoint:N`.)

`\knot_test_endpoint:nn` Wrapper around the above.

```
7629 \prg_new_protected_conditional:Npnn \knot_test_endpoint:Nnn #1#2#3 {T,F,TF}
7630 {
7631   \use:c {spath_#3:Nv} \l__knot_tmpd_tl {knot #2}
7632   \knot_test_endpoint:NNTF #1 \l__knot_tmpd_tl
7633   {
7634     \prg_return_true:
7635   }
7636   {
7637     \prg_return_false:
7638   }
7639 }
7640
7641 \cs_generate_variant:Nn \knot_test_endpoint:NnnT {NVnT,NvnT}
7642 \cs_generate_variant:Nn \knot_test_endpoint:NnnF {NVnF,NvnF}
7643 \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.

```
7644 \cs_new_protected_nopar:Npn \knot_draw_crossing:nnn #1#2#3
7645 {
7646   \group_begin:
7647   \pgfscope
7648   \path[knot~ diagram/background~ clip] (#2, #3)
7649   circle[radius=\l__knot_clip_bg_radius_dim];
7650
7651   \tl_set:Nn \l__knot_tmpa_tl {knot~ diagram/every~ strand/.try,}
7652   \tl_if_exist:cT {l__knot_options_ #1}
7653   {
7654     \tl_put_right:Nv \l__knot_tmpa_tl {l__knot_options_ #1}
```

```

7655 }
7656 \tl_put_right:Nn \l__knot_tmpa_tl
7657 {
7658   ,knotbg
7659   ,line~ width= \tl_use:N \l__knot_clip_width_tl * \pgflinewidth
7660 }
7661 \spath_tikz_path:Vv \l__knot_tmpa_tl {knot #1}
7662
7663 \endpgfscope
7664
7665 \pgfscope
7666 \path[knot~ diagram/clip] (#2, #3)
7667 circle[radius=\l__knot_clip_draw_radius_dim];
7668
7669 \tl_set:Nn \l__knot_tmpa_tl {knot~ diagram/every~ strand/.try,}
7670 \tl_if_exist:cT {l__knot_options_ #1}
7671 {
7672   \tl_put_right:Nv \l__knot_tmpa_tl {l__knot_options_ #1}
7673 }
7674 \tl_put_right:Nn \l__knot_tmpa_tl
7675 {
7676   ,knot~ diagram/only~ when~ rendering/.try
7677   ,only~ when~ rendering/.try
7678 }
7679 \spath_tikz_path:Vv \l__knot_tmpa_tl {knot #1}
7680
7681 \endpgfscope
7682 \group_end:
7683 }
7684
7685 \cs_generate_variant:Nn \knot_draw_crossing:nnn {nVV, VVV}
7686
7687 \cs_new_protected_nopar:Npn \knot_draw_crossing:nn #1#2
7688 {
7689   \tikz@scan@one@point\pgfutil@firstofone #2 \relax
7690   \knot_draw_crossing:nVV {#1} \pgf@x \pgf@y
7691 }

```

(End definition for \knot_draw_crossing:nnn.)

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

```

7692 \cs_new_protected_nopar:Npn \knot_split_strands:
7693 {
7694   \int_gzero:N \g__knot_filaments_int
7695   \int_step_function:nnnN {1} {1} {\l__knot_strands_int} \knot_split_strand:n
7696   \int_step_function:nnnN {1} {1} {\g__knot_filaments_int} \knot_compute_nexts:n
7697 }

```

(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.

```

7698 \cs_new_protected_nopar:Npn \knot_compute_nexts:n #1
7699 {

```

```

7700 \tl_clear_new:c {knot next \tl_use:c {knot previous filament #1}}
7701 \tl_set:cn {knot next \tl_use:c {knot previous filament #1}} {filament #1}
7702 }

```

(End definition for \knot_compute_nexts:n.)

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

```

7703 \cs_new_protected_nopar:Npn \knot_split_strand:n #1
7704 {
7705   \int_set_eq:NN \l__knot_component_start_int \g__knot_filaments_int
7706   \int_incr:N \l__knot_component_start_int
7707   \tl_set_eq:Nc \l__knot_tmpa_tl {l__knot_options_strand #1}
7708   \spath_segments_to_seq:Nv \l__knot_segments_seq {knot strand #1}
7709   \seq_map_function:NN \l__knot_segments_seq \knot_save_filament:N
7710 }

```

(End definition for \knot_split_strand:n.)

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

```

7711 \cs_new_protected_nopar:Npn \knot_save_filament:N #1
7712 {
7713   \tl_set:Nx \l__knot_tmpb_tl {\tl_item:nn {#1} {4}}
7714   \tl_case:NnF \l__knot_tmpb_tl
7715   {
7716     \c_spath_moveto_tl
7717     {
7718       \int_compare:nT {\l__knot_component_start_int < \g__knot_filaments_int}
7719       {
7720         \int_set_eq:NN \l__knot_component_start_int \g__knot_filaments_int
7721       }
7722     }
7723     \c_spath_lineto_tl
7724     {
7725       \int_gincr:N \g__knot_filaments_int
7726       \tl_clear_new:c {knot filament \int_use:N \g__knot_filaments_int}
7727       \tl_set:cn {knot filament \int_use:N \g__knot_filaments_int} {#1}
7728
7729       \tl_clear_new:c {l__knot_options_filament \int_use:N \g__knot_filaments_int}
7730       \tl_set_eq:cN {l__knot_options_filament \int_use:N \g__knot_filaments_int}
7731       \l__knot_tmpa_tl
7732
7733       \tl_clear_new:c {knot previous filament \int_use:N \g__knot_filaments_int}
7734       \int_compare:nF {\l__knot_component_start_int == \g__knot_filaments_int}
7735       {
7736         \tl_set:cx {knot previous filament \int_use:N \g__knot_filaments_int}
7737         {filament \int_eval:n {\g__knot_filaments_int - 1}}
7738       }
7739     }
7740     \c_spath_curvetoa_tl
7741     {
7742       \int_gincr:N \g__knot_filaments_int
7743       \tl_clear_new:c {knot filament \int_use:N \g__knot_filaments_int}
7744       \tl_set:cn {knot filament \int_use:N \g__knot_filaments_int} {#1}
7745       \tl_clear_new:c {l__knot_options_filament \int_use:N \g__knot_filaments_int}
7746       \tl_set_eq:cN {l__knot_options_filament \int_use:N \g__knot_filaments_int}

```

```

7747 \l__knot_tmpa_tl
7748
7749 \tl_clear_new:c {knot previous filament \int_use:N \g__knot_filaments_int}
7750 \int_compare:nF {\l__knot_component_start_int == \g__knot_filaments_int}
7751 {
7752   \tl_set:cx
7753   {knot previous filament \int_use:N \g__knot_filaments_int}
7754   {filament \int_eval:n {\g__knot_filaments_int - 1}}
7755 }
7756 }
7757 \c_spath_closepath_tl
7758 {
7759   \int_gincr:N \g__knot_filaments_int
7760   \tl_clear_new:c {knot filament \int_use:N \g__knot_filaments_int}
7761   \tl_clear:N \l__knot_tmpa_tl
7762   \tl_put_right:Nx
7763   {
7764     \tl_item:nn {#1} {1}\tl_item:nn {#1} {2}\tl_item:nn {#1} {3}
7765   }
7766   \tl_put_right:NV \l__knot_tmpa_tl \c_spath_lineto_tl
7767   \tl_put_right:Nx {\tl_item:nn {#1} {5}\tl_item:nn {#1} {6}}
7768
7769   \tl_set:cV {knot filament \int_use:N \g__knot_filaments_int} \l__knot_tmpa_tl
7770   \tl_set_eq:cN {\l__knot_options_filament \int_use:N \g__knot_filaments_int}
7771   \l__knot_tmpa_tl
7772   \tl_clear_new:c {knot previous filament \int_use:N \g__knot_filaments_int}
7773   \int_compare:nF {\l__knot_component_start_int == \g__knot_filaments_int}
7774   {
7775     \tl_set:cx
7776     {knot previous filament \int_use:N \g__knot_filaments_int}
7777     {filament \int_eval:n {\g__knot_filaments_int - 1}}
7778   }
7779   \tl_set:cx
7780   {knot previous filament \int_use:N \l__knot_component_start_int}
7781   {filament \int_use:N \g__knot_filaments_int}
7782 }
7783 }
7784 {
7785 }
7786 }

```

(End definition for \knot_save_filament:N.)

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

```

7787 \NewDocumentCommand \redraw { m m }
7788 {
7789   % \tikz@scan@one@point\pgfutil@firstofone #2 \relax
7790   \tl_put_right:Nn \l__knot_redraws_tl {\knot_draw_crossing:nn}
7791   \tl_put_right:Nx \l__knot_redraws_tl {
7792     {strand #1} {#2}% {\dim_use:N \pgf@x} {\dim_use:N \pgf@y}
7793   }
7794 }

```

(End definition for \redraw.)

```

7795 \ExplSyntaxOff

```

<@@=>

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

```
7796 \def\pgf@sh__knotknotanchor#1#2{%
7797   \anchor{#2 north west}{%
7798     \csname pgf@anchor@knot #1@north west\endcsname%
7799     \pgf@x=#2\pgf@x%
7800     \pgf@y=#2\pgf@y%
7801   }%
7802   \anchor{#2 north east}{%
7803     \csname pgf@anchor@knot #1@north east\endcsname%
7804     \pgf@x=#2\pgf@x%
7805     \pgf@y=#2\pgf@y%
7806   }%
7807   \anchor{#2 south west}{%
7808     \csname pgf@anchor@knot #1@south west\endcsname%
7809     \pgf@x=#2\pgf@x%
7810     \pgf@y=#2\pgf@y%
7811   }%
7812   \anchor{#2 south east}{%
7813     \csname pgf@anchor@knot #1@south east\endcsname%
7814     \pgf@x=#2\pgf@x%
7815     \pgf@y=#2\pgf@y%
7816   }%
7817   \anchor{#2 north}{%
7818     \csname pgf@anchor@knot #1@north\endcsname%
7819     \pgf@x=#2\pgf@x%
7820     \pgf@y=#2\pgf@y%
7821   }%
7822   \anchor{#2 east}{%
7823     \csname pgf@anchor@knot #1@east\endcsname%
7824     \pgf@x=#2\pgf@x%
7825     \pgf@y=#2\pgf@y%
7826   }%
7827   \anchor{#2 west}{%
7828     \csname pgf@anchor@knot #1@west\endcsname%
7829     \pgf@x=#2\pgf@x%
7830     \pgf@y=#2\pgf@y%
7831   }%
7832   \anchor{#2 south}{%
7833     \csname pgf@anchor@knot #1@south\endcsname%
7834     \pgf@x=#2\pgf@x%
7835     \pgf@y=#2\pgf@y%
7836   }%
7837 }
```

(End definition for \pgf@sh__knotknotanchor.)

knot_crossing

```
7838 \pgfdeclareshape{knot crossing}
7839 {
7840   \inheritsavedanchors[from=circle] % this is nearly a circle
7841   \inheritanchorborder[from=circle]
7842   \inheritanchor[from=circle]{north}
```

```

7843 \inheritanchor[from=circle]{north west}
7844 \inheritanchor[from=circle]{north east}
7845 \inheritanchor[from=circle]{center}
7846 \inheritanchor[from=circle]{west}
7847 \inheritanchor[from=circle]{east}
7848 \inheritanchor[from=circle]{mid}
7849 \inheritanchor[from=circle]{mid west}
7850 \inheritanchor[from=circle]{mid east}
7851 \inheritanchor[from=circle]{base}
7852 \inheritanchor[from=circle]{base west}
7853 \inheritanchor[from=circle]{base east}
7854 \inheritanchor[from=circle]{south}
7855 \inheritanchor[from=circle]{south west}
7856 \inheritanchor[from=circle]{south east}
7857 \inheritanchorborder[from=circle]
7858 \pgf@sh__knotknotanchor{crossing}{2}
7859 \pgf@sh__knotknotanchor{crossing}{3}
7860 \pgf@sh__knotknotanchor{crossing}{4}
7861 \pgf@sh__knotknotanchor{crossing}{8}
7862 \pgf@sh__knotknotanchor{crossing}{16}
7863 \pgf@sh__knotknotanchor{crossing}{32}
7864 \backgroundpath{
7865   \pgfutil@tempdima=\radius%
7866   \pgfmathsetlength{\pgf@xb}{\pgfkeysvalueof{/pgf/outer xsep}}%
7867   \pgfmathsetlength{\pgf@yb}{\pgfkeysvalueof{/pgf/outer ysep}}%
7868   \ifdim\pgf@xb<\pgf@yb%
7869     \advance\pgfutil@tempdima by-\pgf@yb%
7870   \else%
7871     \advance\pgfutil@tempdima by-\pgf@xb%
7872   \fi%
7873 }
7874 }

```

(End definition for knot crossing.)

knot_□over_□cross

```

7875 \pgfdeclareshape{knot over cross}
7876 {
7877   \inheritsavedanchors[from=rectangle] % this is nearly a circle
7878   \inheritanchorborder[from=rectangle]
7879   \inheritanchor[from=rectangle]{north}
7880   \inheritanchor[from=rectangle]{north west}
7881   \inheritanchor[from=rectangle]{north east}
7882   \inheritanchor[from=rectangle]{center}
7883   \inheritanchor[from=rectangle]{west}
7884   \inheritanchor[from=rectangle]{east}
7885   \inheritanchor[from=rectangle]{mid}
7886   \inheritanchor[from=rectangle]{mid west}
7887   \inheritanchor[from=rectangle]{mid east}
7888   \inheritanchor[from=rectangle]{base}
7889   \inheritanchor[from=rectangle]{base west}
7890   \inheritanchor[from=rectangle]{base east}
7891   \inheritanchor[from=rectangle]{south}
7892   \inheritanchor[from=rectangle]{south west}

```



```

7893 \inheritanchor[from=rectangle]{south east}
7894 \inheritanchorborder[from=rectangle]
7895 \backgroundpath{
7896   \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
7897   \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
7898   \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
7899   \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
7900 }
7901 \foregroundpath{
7902 % store lower right in xa/ya and upper right in xb/yb
7903   \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
7904   \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
7905   \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
7906   \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
7907 }
7908 }

```

(End definition for knot over cross.)

knot_{under}cross

```

7909 \pgfdeclareshape{knot under cross}
7910 {
7911   \inheritssavedanchors[from=rectangle] % this is nearly a circle
7912   \inheritanchorborder[from=rectangle]
7913   \inheritanchor[from=rectangle]{north}
7914   \inheritanchor[from=rectangle]{north west}
7915   \inheritanchor[from=rectangle]{north east}
7916   \inheritanchor[from=rectangle]{center}
7917   \inheritanchor[from=rectangle]{west}
7918   \inheritanchor[from=rectangle]{east}
7919   \inheritanchor[from=rectangle]{mid}
7920   \inheritanchor[from=rectangle]{mid west}
7921   \inheritanchor[from=rectangle]{mid east}
7922   \inheritanchor[from=rectangle]{base}
7923   \inheritanchor[from=rectangle]{base west}
7924   \inheritanchor[from=rectangle]{base east}
7925   \inheritanchor[from=rectangle]{south}
7926   \inheritanchor[from=rectangle]{south west}
7927   \inheritanchor[from=rectangle]{south east}
7928   \inheritanchorborder[from=rectangle]
7929   \backgroundpath{
7930     \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
7931     \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
7932     \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
7933     \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
7934   }
7935   \foregroundpath{
7936 % store lower right in xa/ya and upper right in xb/yb
7937     \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
7938     \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
7939     \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
7940     \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
7941   }
7942 }

```

(End definition for knot under cross.)

knot_␣vert

```
7943 \pgfdeclareshape{knot vert}
7944 {
7945   \inheritsavedanchors[from=rectangle] % this is nearly a circle
7946   \inheritanchorborder[from=rectangle]
7947   \inheritanchor[from=rectangle]{north}
7948   \inheritanchor[from=rectangle]{north west}
7949   \inheritanchor[from=rectangle]{north east}
7950   \inheritanchor[from=rectangle]{center}
7951   \inheritanchor[from=rectangle]{west}
7952   \inheritanchor[from=rectangle]{east}
7953   \inheritanchor[from=rectangle]{mid}
7954   \inheritanchor[from=rectangle]{mid west}
7955   \inheritanchor[from=rectangle]{mid east}
7956   \inheritanchor[from=rectangle]{base}
7957   \inheritanchor[from=rectangle]{base west}
7958   \inheritanchor[from=rectangle]{base east}
7959   \inheritanchor[from=rectangle]{south}
7960   \inheritanchor[from=rectangle]{south west}
7961   \inheritanchor[from=rectangle]{south east}
7962   \inheritanchorborder[from=rectangle]
7963   \backgroundpath{
7964     % store lower right in xa/ya and upper right in xb/yb
7965     \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
7966     \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
7967     \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
7968     \pgfpathlineto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
7969     \pgfpathmoveto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
7970     \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
7971   }
7972 }
```

(End definition for knot vert.)

knot_␣horiz

```
7973 \pgfdeclareshape{knot horiz}
7974 {
7975   \inheritsavedanchors[from=rectangle] % this is nearly a circle
7976   \inheritanchorborder[from=rectangle]
7977   \inheritanchor[from=rectangle]{north}
7978   \inheritanchor[from=rectangle]{north west}
7979   \inheritanchor[from=rectangle]{north east}
7980   \inheritanchor[from=rectangle]{center}
7981   \inheritanchor[from=rectangle]{west}
7982   \inheritanchor[from=rectangle]{east}
7983   \inheritanchor[from=rectangle]{mid}
7984   \inheritanchor[from=rectangle]{mid west}
7985   \inheritanchor[from=rectangle]{mid east}
7986   \inheritanchor[from=rectangle]{base}
7987   \inheritanchor[from=rectangle]{base west}
7988   \inheritanchor[from=rectangle]{base east}
7989   \inheritanchor[from=rectangle]{south}
```

```

7990 \inheritanchor[from=rectangle]{south west}
7991 \inheritanchor[from=rectangle]{south east}
7992 \inheritanchorborder[from=rectangle]
7993 \foregroundpath{
7994 % store lower right in xa/ya and upper right in xb/yb
7995   \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
7996   \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
7997   \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
7998   \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
7999   \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
8000   \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
8001 }
8002 }

```

(End definition for knot horiz.)