

The `spath3` package: code

Andrew Stacey

loopspace@mathforge.org

v2.0 from 2021/01/19

1 Introduction

The `spath3` package is intended as a library for manipulating PGF's *soft paths*. In between defining a path and using it, PGF stores a path as a *soft path* where all the defining structure has been resolved into the basic operations but these have not yet been written to the output file. They can therefore still be manipulated by `TEX`, and as they have a very rigid form (and limited vocabulary), they are relatively easy to modify. This package provides some methods for working with these paths. It 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} {2021/01/19} {2.0} {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.

```
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 }
```

We need a slew of temporary variables.

```

30 \tl_new:N \l__spath_tmpa_tl
31 \tl_new:N \l__spath_tmpb_tl
32 \tl_new:N \l__spath_tmpc_tl
33 \tl_new:N \l__spath_tmpd_tl
34 \tl_new:N \l__spath_tmpe_tl
35
36 \seq_new:N \l__spath_tmpa_seq
37 \seq_new:N \l__spath_tmpb_seq
38
39 \tl_new:N \g__spath_output_tl
40 \int_new:N \g__spath_output_int
41 \seq_new:N \g__spath_output_seq
42
43 \dim_new:N \l__spath_tmpa_dim
44 \dim_new:N \l__spath_tmpb_dim
45 \dim_new:N \l__spath_move_x_dim
46 \dim_new:N \l__spath_move_y_dim
47 \fp_new:N \l__spath_tmpa_fp
48 \fp_new:N \l__spath_tmpb_fp
49 \int_new:N \l__spath_tmpa_int
50
51 \bool_new:N \g__spath_output_bool
52 \bool_new:N \l__spath_closed_bool

```

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.

```

53 \tl_const:Nn \c_spath_moveto_tl {\pgfsyssoftpath@movetotoken}
54 \tl_const:Nn \c_spath_lineto_tl {\pgfsyssoftpath@linetotoken}
55 \tl_const:Nn \c_spath_curveto_tl {\pgfsyssoftpath@curvetotoken}
56 \tl_const:Nn \c_spath_curvetoa_tl {\pgfsyssoftpath@curvetosupportatoken}
57 \tl_const:Nn \c_spath_curvetob_tl {\pgfsyssoftpath@curvetosupportbtoken}
58 \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.

```

59 \int_new:N \g__spath_anon_int
60 \int_gzero:N \g__spath_anon_int

```

Groups and iterations don't mix well and I haven't got a good scheme for protecting local calculations when iterating, so we do our best with iteration-specific variables.

```

61 \tl_new:N \l__spath_itera_tl
62 \tl_new:N \l__spath_iterb_tl
63 \tl_new:N \l__spath_iterc_tl
64 \tl_new:N \l__spath_iterd_tl
65 \tl_new:N \l__spath_iterp_tl
66 \dim_new:N \l__spath_itera_dim
67 \dim_new:N \l__spath_iterb_dim
68 \seq_new:N \l__spath_iter_seq

```

And some error messages

```

69 \msg_new:nnn { spath3 } { unknown path construction } { The~ path~ construction~ element~ #1

```

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
70 \cs_new_protected_nopar:Npn \__spath_segments_to_seq:n #1
71 {
72   \group_begin:
73   \tl_set:Nn \l__spath_itera_tl {#1}
74   \tl_clear:N \l__spath_iterb_tl
75   \seq_clear:N \l__spath_iter_seq
76   \dim_zero:N \l__spath_itera_dim
77   \dim_zero:N \l__spath_iterb_dim
78
79   \bool_until_do:nm {
80     \tl_if_empty_p:N \l__spath_itera_tl
81   }
82   {
83     \tl_set:Nx \l__spath_iterc_tl {\tl_head:N \l__spath_itera_tl}
84     \tl_set:Nx \l__spath_itera_tl {\tl_tail:N \l__spath_itera_tl}
85     \tl_case:NnF \l__spath_iterc_tl
86     {
87       \c_spath_moveto_tl
88       {
89         \tl_set_eq:NN \l__spath_iterb_tl \c_spath_moveto_tl
90         \tl_put_right:Nx \l__spath_iterb_tl {\tl_head:N \l__spath_itera_tl}
91         \dim_set:Nn \l__spath_itera_dim {\tl_head:N \l__spath_itera_tl}
92         \tl_set:Nx \l__spath_itera_tl {\tl_tail:N \l__spath_itera_tl}
93
94         \tl_put_right:Nx \l__spath_iterb_tl {\tl_head:N \l__spath_itera_tl}
95         \dim_set:Nn \l__spath_iterb_dim {\tl_head:N \l__spath_itera_tl}
96         \tl_set:Nx \l__spath_itera_tl {\tl_tail:N \l__spath_itera_tl}
97
98         \tl_set:Nx \l__spath_iterd_tl {\tl_head:N \l__spath_itera_tl}
99         \tl_if_eq:NNF \l__spath_iterd_tl \c_spath_moveto_tl
100        {
101          \tl_clear:N \l__spath_iterb_tl
102        }
103      }
104    }
105
106   \c_spath_lineto_tl
107   {
108     \tl_set_eq:NN \l__spath_iterb_tl \c_spath_moveto_tl
109     \tl_put_right:Nx \l__spath_iterb_tl
110     {
111       {\dim_use:N \l__spath_itera_dim}
112       {\dim_use:N \l__spath_iterb_dim}
113     }
114     \tl_put_right:Nv \l__spath_iterb_tl \c_spath_lineto_tl
115
116     \tl_put_right:Nx \l__spath_iterb_tl {\tl_head:N \l__spath_itera_tl}
117     \dim_set:Nn \l__spath_itera_dim {\tl_head:N \l__spath_itera_tl}
118     \tl_set:Nx \l__spath_itera_tl {\tl_tail:N \l__spath_itera_tl}

```

```

119
120 \tl_put_right:Nx \l__spath_iterb_tl {{\tl_head:N \l__spath_itera_tl}}
121 \dim_set:Nn \l__spath_iterb_dim {\tl_head:N \l__spath_itera_tl}
122 \tl_set:Nx \l__spath_itera_tl {\tl_tail:N \l__spath_itera_tl}
123
124 }
125
126 \c_spath_curvetoa_tl
127 {
128 \tl_set_eq:NN \l__spath_iterb_tl \c_spath_moveto_tl
129 \tl_put_right:Nx \l__spath_iterb_tl
130 {
131 {\dim_use:N \l__spath_itera_dim}
132 {\dim_use:N \l__spath_iterb_dim}
133 }
134 \tl_put_right:NV \l__spath_iterb_tl \c_spath_curvetoa_tl
135
136 \prg_replicate:nn {2} {
137 \tl_put_right:Nx \l__spath_iterb_tl {{\tl_head:N \l__spath_itera_tl}}
138 \tl_set:Nx \l__spath_itera_tl {\tl_tail:N \l__spath_itera_tl}
139 \tl_put_right:Nx \l__spath_iterb_tl {{\tl_head:N \l__spath_itera_tl}}
140 \tl_set:Nx \l__spath_itera_tl {\tl_tail:N \l__spath_itera_tl}
141 \tl_put_right:Nx \l__spath_iterb_tl {\tl_head:N \l__spath_itera_tl}
142 \tl_set:Nx \l__spath_itera_tl {\tl_tail:N \l__spath_itera_tl}
143 }
144
145 \tl_put_right:Nx \l__spath_iterb_tl {{\tl_head:N \l__spath_itera_tl}}
146 \dim_set:Nn \l__spath_itera_dim {\tl_head:N \l__spath_itera_tl}
147 \tl_set:Nx \l__spath_itera_tl {\tl_tail:N \l__spath_itera_tl}
148
149 \tl_put_right:Nx \l__spath_iterb_tl {{\tl_head:N \l__spath_itera_tl}}
150 \dim_set:Nn \l__spath_iterb_dim {\tl_head:N \l__spath_itera_tl}
151 \tl_set:Nx \l__spath_itera_tl {\tl_tail:N \l__spath_itera_tl}
152
153 }
154
155 \c_spath_closepath_tl
156 {
157 \tl_set_eq:NN \l__spath_iterb_tl \c_spath_moveto_tl
158 \tl_put_right:Nx \l__spath_iterb_tl
159 {
160 {\dim_use:N \l__spath_itera_dim}
161 {\dim_use:N \l__spath_iterb_dim}
162 }
163 \tl_put_right:NV \l__spath_iterb_tl \c_spath_lineto_tl
164
165 \tl_put_right:Nx \l__spath_iterb_tl {{\tl_head:N \l__spath_itera_tl}}
166 \dim_set:Nn \l__spath_itera_dim {\tl_head:N \l__spath_itera_tl}
167 \tl_set:Nx \l__spath_itera_tl {\tl_tail:N \l__spath_itera_tl}
168
169 \tl_put_right:Nx \l__spath_iterb_tl {{\tl_head:N \l__spath_itera_tl}}
170 \dim_set:Nn \l__spath_iterb_dim {\tl_head:N \l__spath_itera_tl}
171 \tl_set:Nx \l__spath_itera_tl {\tl_tail:N \l__spath_itera_tl}
172

```

```

173     }
174
175   }
176   {
177
178     \tl_set_eq:NN \l__spath_iterb_tl \l__spath_iterc_tl
179     \tl_put_right:Nx \l__spath_iterb_tl {{\tl_head:N \l__spath_itera_tl}}
180     \dim_set:Nn \l__spath_itera_dim {\tl_head:N \l__spath_itera_tl}
181     \tl_set:Nx \l__spath_itera_tl {\tl_tail:N \l__spath_itera_tl}
182
183     \tl_put_right:Nx \l__spath_iterb_tl {{\tl_head:N \l__spath_itera_tl}}
184     \dim_set:Nn \l__spath_iterb_dim {\tl_head:N \l__spath_itera_tl}
185     \tl_set:Nx \l__spath_itera_tl {\tl_tail:N \l__spath_itera_tl}
186
187   }
188
189   \tl_if_empty:NF \l__spath_iterb_tl
190   {
191     \seq_put_right:NV \l__spath_iter_seq \l__spath_iterb_tl
192   }
193   \tl_clear:N \l__spath_iterb_tl
194 }
195
196 \seq_gclear:N \g__spath_output_seq
197 \seq_gset_eq:NN \g__spath_output_seq \l__spath_iter_seq
198 \group_end:
199 }
200 \cs_new_protected_nopar:Npn \spath_segments_to_seq:Nn #1#2
201 {
202   \__spath_segments_to_seq:n {#2}
203   \seq_clear_new:N #1
204   \seq_set_eq:NN #1 \g__spath_output_seq
205   \seq_gclear:N \g__spath_output_seq
206 }
207 \cs_generate_variant:Nn \spath_segments_to_seq:Nn {NV, cn, cV, Nv, cv}
208 \cs_new_protected_nopar:Npn \spath_segments_gto_seq:Nn #1#2
209 {
210   \__spath_segments_to_seq:n {#2}
211   \seq_clear_new:N #1
212   \seq_gset_eq:NN #1 \g__spath_output_seq
213   \seq_gclear:N \g__spath_output_seq
214 }
215 \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`. These functions are documented on page ??.)

`\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 216 \cs_new_protected_nopar:Npn \__spath_components_to_seq:n #1
\spath_components_to_clist:Nn 217 {
\spath_components_gto_clist:Nn 218   \group_begin:
219   \tl_set:Nn \l__spath_itera_tl {#1}
220   \seq_clear:N \l__spath_iter_seq
221   \tl_set:Nx \l__spath_itera_tl {\tl_tail:N \l__spath_itera_tl}

```

```

222 \tl_put_right:NV \l__spath_itera_tl \c_spath_moveto_tl
223 \tl_set_eq:NN \l__spath_iterb_tl \c_spath_moveto_tl
224 \bool_do_until:nm {
225   \tl_if_empty_p:N \l__spath_itera_tl
226 }
227 {
228   \tl_set:Nx \l__spath_iterc_tl {\tl_head:N \l__spath_itera_tl}
229   \tl_if_eq:NNT \l__spath_iterc_tl \c_spath_moveto_tl
230   {
231     \seq_put_right:NV \l__spath_iter_seq \l__spath_iterb_tl
232     \tl_clear:N \l__spath_iterb_tl
233   }
234   \tl_if_single:NTF \l__spath_iterc_tl
235   {
236     \tl_put_right:NV \l__spath_iterb_tl \l__spath_iterc_tl
237   }
238   {
239     \tl_put_right:Nx \l__spath_iterb_tl {{\l__spath_iterc_tl}}
240   }
241   \tl_set:Nx \l__spath_itera_tl {\tl_tail:N \l__spath_itera_tl}
242 }
243
244 \seq_gclear:N \g__spath_output_seq
245 \seq_gset_eq:NN \g__spath_output_seq \l__spath_iter_seq
246 \group_end:
247 }
248 \cs_new_protected_nopar:Npn \spath_components_to_seq:Nn #1#2
249 {
250   \__spath_components_to_seq:n {#2}
251   \seq_clear_new:N #1
252   \seq_set_eq:NN #1 \g__spath_output_seq
253   \seq_gclear:N \g__spath_output_seq
254 }
255 \cs_generate_variant:Nn \spath_components_to_seq:Nn {NV, cn, cV, cv, Nv}
256 \cs_new_protected_nopar:Npn \spath_components_gto_seq:Nn #1#2
257 {
258   \__spath_components_to_seq:n {#2}
259   \seq_clear_new:N #1
260   \seq_gset_eq:NN #1 \g__spath_output_seq
261   \seq_gclear:N \g__spath_output_seq
262 }
263 \cs_generate_variant:Nn \spath_components_gto_seq:Nn {NV, cn, cV, cv, Nv}
264 \cs_new_protected_nopar:Npn \spath_components_to_clist:Nn #1#2
265 {
266   \__spath_components_to_seq:n {#2}
267   \clist_clear_new:N #1
268   \clist_set_from_seq:NN #1 \g__spath_output_seq
269   \seq_gclear:N \g__spath_output_seq
270 }
271 \cs_generate_variant:Nn \spath_components_to_clist:Nn {NV, cn, cV, cv, Nv}
272 \cs_new_protected_nopar:Npn \spath_components_gto_clist:Nn #1#2
273 {
274   \__spath_components_to_seq:n {#2}
275   \clist_clear_new:N #1

```

```

276 \clist_gset_from_seq:NN #1 \g__spath_output_seq
277 \seq_gclear:N \g__spath_output_seq
278 }
279 \cs_generate_variant:Nn \spath_components_gto_clist:Nn {NV, cn, cV, cv, Nv}

```

(End definition for `\spath_components_to_seq:Nn` and others. These functions are documented on page ??.)

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

```

280 \cs_new_protected_nopar:Npn \spath_length:n #1
281 {
282   \int_eval:n {\tl_count:n {#1} / 3}
283 }
284 \cs_generate_variant:Nn \spath_length:n {V}

```

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

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

`\spath_greallength:Nn`

```

285 \cs_new_protected_nopar:Npn \__spath_reallength:n #1
286 {
287   \group_begin:
288   \int_set:Nn \l__spath_tmpa_int {0}
289   \tl_map_inline:nn {#1} {
290     \tl_set:Nn \l__spath_tmpc_tl {##1}
291     \tl_case:NnT \l__spath_tmpc_tl
292     {
293       \c_spath_lineto_tl {}
294       \c_spath_curveto_tl {}
295       \c_spath_closepath_tl {}
296     }
297     {
298       \int_incr:N \l__spath_tmpa_int
299     }
300   }
301   \int_gzero:N \g__spath_output_int
302   \int_gset_eq:NN \g__spath_output_int \l__spath_tmpa_int
303   \group_end:
304 }
305 \cs_new_protected_nopar:Npn \spath_reallength:Nn #1#2
306 {
307   \__spath_reallength:n {#2}
308   \int_set_eq:NN #1 \g__spath_output_int
309   \int_gzero:N \g__spath_output_int
310 }
311 \cs_generate_variant:Nn \spath_reallength:Nn {NV, cn, cV, Nv, cv}
312 \cs_new_protected_nopar:Npn \spath_greallength:Nn #1#2
313 {
314   \__spath_reallength:n {#2}
315   \int_gset_eq:NN #1 \g__spath_output_int
316   \int_gzero:N \g__spath_output_int
317 }
318 \cs_generate_variant:Nn \spath_greallength:Nn {NV, cn, cV}

```

(End definition for `\spath_reallength:Nn` and `\spath_greallength:Nn`. These functions are documented on page ??.)

`\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`

```

319 \cs_new_protected_nopar:Npn \__spath_numberofcomponents:n #1
320 {
321   \group_begin:
322   \int_set:Nn \l__spath_tmpa_int {0}
323   \tl_map_inline:nn {#1} {
324     \tl_set:Nn \l__spath_tmpa_tl {##1}
325     \tl_case:Nn \l__spath_tmpa_tl
326     {
327       \c_spath_moveto_tl
328       {
329         \int_incr:N \l__spath_tmpa_int
330       }
331     }
332   }
333   \int_gzero:N \g__spath_output_int
334   \int_gset_eq:NN \g__spath_output_int \l__spath_tmpa_int
335   \group_end:
336 }
337 \cs_new_protected_nopar:Npn \spath_numberofcomponents:Nn #1#2
338 {
339   \__spath_numberofcomponents:n {#2}
340   \int_set_eq:NN #1 \g__spath_output_int
341   \int_gzero:N \g__spath_output_int
342 }
343 \cs_generate_variant:Nn \spath_numberofcomponents:Nn {NV, cn, cV}
344 \cs_new_protected_nopar:Npn \spath_gnumberofcomponents:Nn #1#2
345 {
346   \__spath_numberofcomponents:n {#2}
347   \int_gset_eq:NN #1 \g__spath_output_int
348   \int_gzero:N \g__spath_output_int
349 }
350 \cs_generate_variant:Nn \spath_gnumberofcomponents:Nn {NV, cn, cV}

```

(End definition for `\spath_numberofcomponents:Nn` and `\spath_gnumberofcomponents:Nn`. These functions are documented on page ??.)

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

`\spath_ginitialpoint:Nn`

```

351 \cs_new_protected_nopar:Npn \__spath_initialpoint:n #1
352 {
353   \group_begin:
354   \tl_clear:N \l__spath_tmpb_tl
355   \tl_set:Nx \l__spath_tmpb_tl
356   {
357     { \tl_item:nn {#1} {2} }
358     { \tl_item:nn {#1} {3} }
359   }
360   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
361   \group_end:
362 }
363 \cs_new_protected_nopar:Npn \spath_initialpoint:Nn #1#2
364 {
365   \__spath_initialpoint:n {#2}

```

```

366 \tl_set_eq:NN #1 \g__spath_output_tl
367 \tl_gclear:N \g__spath_output_tl
368 }
369 \cs_generate_variant:Nn \spath_initialpoint:Nn {NV, cn, cV, Nv}
370 \cs_new_protected_nopar:Npn \spath_ginitialpoint:Nn #1#2
371 {
372   \__spath_initialpoint:n {#2}
373   \tl_gset_eq:NN #1 \g__spath_output_tl
374   \tl_gclear:N \g__spath_output_tl
375 }
376 \cs_generate_variant:Nn \spath_ginitialpoint:Nn {NV, cn, cV, Nv}

```

(End definition for `\spath_initialpoint:Nn` and `\spath_ginitialpoint:Nn`. These functions are documented on page ??.)

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

```

\spath_gfinalpoint:Nn
377 \cs_new_protected_nopar:Npn \__spath_finalpoint:n #1
378 {
379   \group_begin:
380   \tl_set:Nn \l__spath_tmpa_tl {#1}
381   \tl_reverse:N \l__spath_tmpa_tl
382   \tl_clear:N \l__spath_tmpb_tl
383   \tl_set:Nx \l__spath_tmpb_tl
384   {
385     { \tl_item:Nn \l__spath_tmpa_tl {2} }
386     { \tl_item:Nn \l__spath_tmpa_tl {1} }
387   }
388   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
389   \group_end:
390 }
391 \cs_new_protected_nopar:Npn \spath_finalpoint:Nn #1#2
392 {
393   \__spath_finalpoint:n {#2}
394   \tl_set_eq:NN #1 \g__spath_output_tl
395   \tl_gclear:N \g__spath_output_tl
396 }
397 \cs_generate_variant:Nn \spath_finalpoint:Nn {NV, cn, cV, Nv}
398 \cs_new_protected_nopar:Npn \spath_gfinalpoint:Nn #1#2
399 {
400   \__spath_finalpoint:n {#2}
401   \tl_gset_eq:NN #1 \g__spath_output_tl
402   \tl_gclear:N \g__spath_output_tl
403 }
404 \cs_generate_variant:Nn \spath_gfinalpoint:Nn {NV, cn, cV, Nv}

```

(End definition for `\spath_finalpoint:Nn` and `\spath_gfinalpoint:Nn`. These functions are documented on page ??.)

`\spath_reverse:Nn` This computes the reverse of the path.

```

\spath_greverse:Nn
405 \cs_new_protected_nopar:Npn \__spath_reverse:n #1
406 {
407   \group_begin:
408   \tl_set:Nn \l__spath_tmpa_tl {#1}
409
410   \tl_clear:N \l__spath_tmpb_tl

```

```

411 \tl_clear:N \l__spath_tmpd_tl
412 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
413 \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
414 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
415 \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
416 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
417
418 \tl_put_left:Nx \l__spath_tmpd_tl
419 {
420   {\dim_use:N \l__spath_tmpa_dim}
421   {\dim_use:N \l__spath_tmpb_dim}
422 }
423
424 \bool_set_false:N \l__spath_closed_bool
425
426 \bool_until_do:nn {
427   \tl_if_empty_p:N \l__spath_tmpa_tl
428 }
429 {
430   \tl_set:Nx \l__spath_tmpe_tl {\tl_head:N \l__spath_tmpa_tl}
431
432   \tl_case:NnTF \l__spath_tmpe_tl
433   {
434     \c_spath_moveto_tl {
435
436       \bool_if:NT \l__spath_closed_bool
437       {
438         \tl_put_right:NV \l__spath_tmpd_tl \c_spath_closepath_tl
439         \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl}
440         \tl_put_right:Nx \l__spath_tmpd_tl
441         {
442           {\tl_head:N \l__spath_tmpe_tl}
443           {\tl_head:N \l__spath_tmpe_tl}
444         }
445       }
446       \bool_set_false:N \l__spath_closed_bool
447       \tl_put_left:NV \l__spath_tmpd_tl \c_spath_moveto_tl
448       \tl_put_left:NV \l__spath_tmpe_tl \l__spath_tmpd_tl
449       \tl_clear:N \l__spath_tmpe_tl
450     }
451     \c_spath_lineto_tl {
452       \tl_put_left:NV \l__spath_tmpd_tl \c_spath_lineto_tl
453     }
454     \c_spath_curveto_tl {
455       \tl_put_left:NV \l__spath_tmpd_tl \c_spath_curvetoa_tl
456     }
457     \c_spath_curvetoa_tl {
458       \tl_put_left:NV \l__spath_tmpd_tl \c_spath_curveto_tl
459     }
460     \c_spath_curvetob_tl {
461       \tl_put_left:NV \l__spath_tmpd_tl \c_spath_curvetob_tl
462     }
463   }
464 {

```

```

465 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
466
467 \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
468 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
469 \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
470 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
471
472 \tl_put_left:Nx \l__spath_tmpd_tl
473 {
474   {\dim_use:N \l__spath_tmpa_dim}
475   {\dim_use:N \l__spath_tmpb_dim}
476 }
477
478 }
479 {
480 \tl_if_eq:NNTF \l__spath_tmpe_tl \c_spath_closepath_tl
481 {
482   \bool_set_true:N \l__spath_closed_bool
483 }
484 {
485   \msg_warning:nx { spath3 } { unknown path construction } {\l__spath_tmpe_tl }
486 }
487
488 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
489 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
490 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
491
492 }
493 }
494
495 \bool_if:NT \l__spath_closed_bool
496 {
497   \tl_put_right:NV \l__spath_tmpd_tl \c_spath_closepath_tl
498   \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpd_tl}
499   \tl_put_right:Nx \l__spath_tmpd_tl
500   {
501     { \tl_head:N \l__spath_tmpd_tl }
502     { \tl_head:N \l__spath_tmpe_tl }
503   }
504 }
505
506 \bool_set_false:N \l__spath_closed_bool
507 \tl_put_left:NV \l__spath_tmpd_tl \c_spath_moveto_tl
508 \tl_put_left:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
509
510 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
511 \group_end:
512 }
513 \cs_new_protected_nopar:Npn \spath_reverse:Nn #1#2
514 {
515   \__spath_reverse:n {#2}
516   \tl_set_eq:NN #1 \g__spath_output_tl
517   \tl_gclear:N \g__spath_output_tl
518 }

```

```

519 \cs_generate_variant:Nn \spath_reverse:Nn {NV, cn, cV, Nv}
520 \cs_new_protected_nopar:Npn \spath_reverse:N #1
521 {
522   \spath_reverse:NV #1#1
523 }
524 \cs_generate_variant:Nn \spath_reverse:N {c}
525 \cs_new_protected_nopar:Npn \spath_greverse:Nn #1#2
526 {
527   \__spath_reverse:n {#2}
528   \tl_gset_eq:NN #1 \g__spath_output_tl
529   \tl_gclear:N \g__spath_output_tl
530 }
531 \cs_generate_variant:Nn \spath_greverse:Nn {NV, cn, cV, Nv}
532 \cs_new_protected_nopar:Npn \spath_greverse:N #1
533 {
534   \spath_greverse:NV #1#1
535 }
536 \cs_generate_variant:Nn \spath_greverse:N {c}

```

(End definition for \spath_reverse:Nn and \spath_greverse:Nn. These functions are documented on page ??.)

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

```

537 \cs_new_protected_nopar:Npn \__spath_initialaction:n #1
538 {
539   \group_begin:
540   \tl_clear:N \l__spath_tmpa_tl
541   \int_compare:nT
542   {
543     \tl_count:n {#1} > 3
544   }
545   {
546     \tl_set:Nx \l__spath_tmpa_tl
547     {
548       \tl_item:Nn {#1} {4}
549     }
550   }
551   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
552   \group_end:
553 }
554 \cs_new_protected_nopar:Npn \spath_initialaction:Nn #1#2
555 {
556   \__spath_initialaction:n {#2}
557   \tl_set_eq:NN #1 \g__spath_output_tl
558   \tl_gclear:N \g__spath_output_tl
559 }
560 \cs_generate_variant:Nn \spath_initialaction:Nn {NV}
561 \cs_new_protected_nopar:Npn \spath_ginitialaction:Nn #1#2
562 {
563   \__spath_initialaction:n {#2}
564   \tl_gset_eq:NN #1 \g__spath_output_tl
565   \tl_gclear:N \g__spath_output_tl
566 }
567 \cs_generate_variant:Nn \spath_ginitialaction:Nn {NV}

```

(End definition for `\spath_initialaction:Nn` and `\spath_ginitialaction:Nn`. These functions are documented on page ??.)

`\spath_finalaction:Nn` This is the last thing that the path does.

```

\spath_gfinalaction:Nn 568 \cs_new_protected_nopar:Npn \__spath_finalaction:n #1
569 {
570   \group_begin:
571   \tl_clear:N \l__spath_tmpb_tl
572   \int_compare:nT
573   {
574     \tl_count:n {#1} > 3
575   }
576   {
577     \tl_set:Nn \l__spath_tmpa_tl {#1}
578     \tl_reverse:N \l__spath_tmpa_tl
579     \tl_set:Nx \l__spath_tmpb_tl
580     {
581       \tl_item:Nn \l__spath_tmpa_tl {3}
582     }
583     \tl_if_eq:NNT \l__spath_tmpb_tl \c_spath_curvetoa_tl
584     {
585       \tl_set_eq:NN \l__spath_tmpb_tl \c_spath_curveto_tl
586     }
587     }
588     \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
589     \group_end:
590   }
591   \cs_new_protected_nopar:Npn \spath_finalaction:Nn #1#2
592   {
593     \__spath_finalaction:n {#2}
594     \tl_set_eq:NN #1 \g__spath_output_tl
595     \tl_gclear:N \g__spath_output_tl
596   }
597   \cs_generate_variant:Nn \spath_finalaction:Nn {NV}
598   \cs_new_protected_nopar:Npn \spath_gfinalaction:Nn #1#2
599   {
600     \__spath_finalaction:n {#2}
601     \tl_gset_eq:NN #1 \g__spath_output_tl
602     \tl_gclear:N \g__spath_output_tl
603   }
604   \cs_generate_variant:Nn \spath_gfinalaction:Nn {NV}

```

(End definition for `\spath_finalaction:Nn` and `\spath_gfinalaction:Nn`. These functions are documented on page ??.)

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

```

\spath_gminbb:Nn 605 \cs_new_protected_nopar:Npn \__spath_minbb:n #1
606 {
607   \group_begin:
608   \tl_set:Nn \l__spath_tmpa_tl {#1}
609   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
610   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
611   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
612   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}

```

```

613 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
614 \bool_until_do:nn {
615   \tl_if_empty_p:N \l__spath_tmpa_tl
616 }
617 {
618   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
619   \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tm
620   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
621   \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tm
622   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
623 }
624 \tl_clear:N \l__spath_tmpb_tl
625 \tl_put_right:Nx \l__spath_tmpb_tl
626 {
627   {\dim_use:N \l__spath_tmpa_dim}
628   {\dim_use:N \l__spath_tmpb_dim}
629 }
630 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
631 \group_end:
632 }
633 \cs_new_protected_nopar:Npn \spath_minbb:Nn #1#2
634 {
635   \__spath_minbb:n {#2}
636   \tl_set_eq:NN #1 \g__spath_output_tl
637   \tl_gclear:N \g__spath_output_tl
638 }
639 \cs_generate_variant:Nn \spath_minbb:Nn {NV, cn, cV}
640 \cs_new_protected_nopar:Npn \spath_gminbb:Nn #1#2
641 {
642   \__spath_minbb:n {#2}
643   \tl_gset_eq:NN #1 \g__spath_output_tl
644   \tl_gclear:N \g__spath_output_tl
645 }
646 \cs_generate_variant:Nn \spath_gminbb:Nn {NV, cn, cV}

```

(End definition for `\spath_minbb:Nn` and `\spath_gminbb:Nn`. These functions are documented on page ??.)

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

```

647 \cs_new_protected_nopar:Npn \__spath_maxbb:n #1
648 {
649   \group_begin:
650   \tl_set:Nn \l__spath_tmpa_tl {#1}
651   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
652   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
653   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
654   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
655   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
656   \bool_until_do:nn {
657     \tl_if_empty_p:N \l__spath_tmpa_tl
658   }
659   {
660     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
661     \dim_set:Nn \l__spath_tmpa_dim {\dim_max:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tm

```

```

662 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
663 \dim_set:Nn \l__spath_tmpb_dim {\dim_max:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tm
664 \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
665 }
666 \tl_clear:N \l__spath_tmpb_tl
667 \tl_put_right:Nx \l__spath_tmpb_tl
668 {
669   {\dim_use:N \l__spath_tmpa_dim}
670   {\dim_use:N \l__spath_tmpb_dim}
671 }
672 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
673 \group_end:
674 }
675 \cs_new_protected_nopar:Npn \spath_maxbb:Nn #1#2
676 {
677   \__spath_maxbb:n {#2}
678   \tl_set_eq:NN #1 \g__spath_output_tl
679   \tl_gclear:N \g__spath_output_tl
680 }
681 \cs_generate_variant:Nn \spath_maxbb:Nn {NV, cn, cV}
682 \cs_new_protected_nopar:Npn \spath_gmaxbb:Nn #1#2
683 {
684   \__spath_maxbb:n {#2}
685   \tl_gset_eq:NN #1 \g__spath_output_tl
686   \tl_gclear:N \g__spath_output_tl
687 }
688 \cs_generate_variant:Nn \spath_gmaxbb:Nn {NV, cn, cV}

```

(End definition for `\spath_maxbb:Nn` and `\spath_gmaxbb:Nn`. These functions are documented on page ??.)

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

```

689 \int_set:Nn \l__spath_tmpa_int {\char_value_catcode:n {'@}}
690 \char_set_catcode_letter:N @
691 \cs_new_protected_nopar:Npn \spath_save_to_aux:Nn #1#2 {
692   \tl_if_empty:nF {#2}
693   {
694     \tl_clear:N \l__spath_tmpa_tl
695     \tl_put_right:Nn \l__spath_tmpa_tl {
696       \ExplSyntaxOn
697       \tl_clear:N #1
698       \tl_set:Nn #1 {#2}
699       \ExplSyntaxOff
700     }
701     \protected@write\@auxout{}\{
702       \tl_to_str:N \l__spath_tmpa_tl
703     }
704   }
705 }
706 \char_set_catcode:nn {'@} {\l__spath_tmpa_int}
707 \cs_generate_variant:Nn \spath_save_to_aux:Nn {cn, cV, NV}
708 \cs_new_protected_nopar:Npn \spath_save_to_aux:N #1
709 {

```



```

710 \tl_if_exist:NT #1
711 {
712   \spath_save_to_aux:NV #1#1
713 }
714 }

```

(End definition for `\spath_save_to_aux:Nn` and `\spath_save_to_aux:N`. These functions are documented on page ??.)

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.

```

\spath_translate:Nnnn Translates a path.
\spath_translate:Nnn 715 \cs_new_protected_nopar:Npn \__spath_translate:nnn #1#2#3
\spath_gtranslate:Nnnn 716 {
\spath_gtranslate:Nnn 717   \group_begin:
718   \tl_set:Nn \l__spath_tmpa_tl {#1}
719   \tl_clear:N \l__spath_tmpb_tl
720   \bool_until_do:mn {
721     \tl_if_empty_p:N \l__spath_tmpa_tl
722   }
723   {
724     \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
725     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
726
727     \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl + #2}
728     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
729
730     \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl + #3}
731     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
732
733     \tl_put_right:Nx \l__spath_tmpb_tl
734     {
735       {\dim_use:N \l__spath_tmpa_dim}
736       {\dim_use:N \l__spath_tmpb_dim}
737     }
738   }
739   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
740   \group_end:
741 }
742 \cs_new_protected_nopar:Npn \spath_translate:Nnnn #1#2#3#4
743 {
744   \__spath_translate:nnn {#2}{#3}{#4}
745   \tl_set_eq:NN #1 \g__spath_output_tl
746   \tl_gclear:N \g__spath_output_tl
747 }
748 \cs_generate_variant:Nn \spath_translate:Nnnn {NVxx, NVVV, NVnn}
749 \cs_new_protected_nopar:Npn \spath_translate:Nnn #1#2#3
750 {

```

```

751 \spath_translate:NVnn #1#1{#2}{#3}
752 }
753 \cs_generate_variant:Nn \spath_translate:Nnn {NVV, cnn, cVV}
754 \cs_new_protected_nopar:Npn \spath_gtranslate:Nnnn #1#2#3#4
755 {
756   \__spath_translate:nnn {#2}{#3}{#4}
757   \tl_gset_eq:NN #1 \g__spath_output_tl
758   \tl_gclear:N \g__spath_output_tl
759 }
760 \cs_generate_variant:Nn \spath_gtranslate:Nnnn {NVxx, NVVV, NVnn}
761 \cs_new_protected_nopar:Npn \spath_gtranslate:Nnn #1#2#3
762 {
763   \spath_gtranslate:NVnn #1#1{#2}{#3}
764 }
765 \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.

```

766 \cs_new_protected_nopar:Npn \spath_translate:Nn #1#2
767 {
768   \spath_translate:Nnn #1 #2
769 }
770 \cs_generate_variant:Nn \spath_translate:Nn {NV}
771 \cs_new_protected_nopar:Npn \spath_gtranslate:Nn #1#2
772 {
773   \spath_gtranslate:Nnn #1 #2
774 }
775 \cs_generate_variant:Nn \spath_gtranslate:Nn {NV}

```

(End definition for \spath_translate:Nnnn and others. These functions are documented on page ??.)

```

\spath_scale:Nnnn Scale a path.
\spath_scale:Nnn
\spath_gscale:Nnnn
\spath_gscale:Nnn
776 \cs_new_protected_nopar:Npn \__spath_scale:nnn #1#2#3
777 {
778   \group_begin:
779   \tl_set:Nn \l__spath_tmpa_tl {#1}
780   \tl_clear:N \l__spath_tmpb_tl
781   \bool_until_do:nn {
782     \tl_if_empty_p:N \l__spath_tmpa_tl
783   }
784   {
785     \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
786     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
787
788     \fp_set:Nn \l__spath_tmpa_fp {\tl_head:N \l__spath_tmpa_tl * #2}
789     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
790
791     \fp_set:Nn \l__spath_tmpb_fp {\tl_head:N \l__spath_tmpa_tl * #3}
792     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
793
794     \tl_put_right:Nx \l__spath_tmpb_tl
795     {
796       {\fp_to_dim:N \l__spath_tmpa_fp}
797       {\fp_to_dim:N \l__spath_tmpb_fp}
798     }

```

```

799   }
800   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
801   \group_end:
802 }
803 \cs_new_protected_nopar:Npn \spath_scale:Nnnn #1#2#3#4
804 {
805   \__spath_scale:nnn {#2}{#3}{#4}
806   \tl_set_eq:NN #1 \g__spath_output_tl
807   \tl_gclear:N \g__spath_output_tl
808 }
809 \cs_generate_variant:Nn \spath_scale:Nnnn {NVnn, Nnxx}
810 \cs_new_protected_nopar:Npn \spath_scale:Nnn #1#2#3
811 {
812   \spath_scale:NVnn #1#1{#2}{#3}
813 }
814 \cs_generate_variant:Nn \spath_scale:Nnn {cnn, cVV, NVV}
815 \cs_new_protected_nopar:Npn \spath_gscale:Nnnn #1#2#3#4
816 {
817   \__spath_scale:nnn {#2}{#3}{#4}
818   \tl_gset_eq:NN #1 \g__spath_output_tl
819   \tl_gclear:N \g__spath_output_tl
820 }
821 \cs_generate_variant:Nn \spath_gscale:Nnnn {NVnn, Nnxx}
822 \cs_new_protected_nopar:Npn \spath_gscale:Nnn #1#2#3
823 {
824   \spath_gscale:NVnn #1#1{#2}{#3}
825 }
826 \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.

```

827 \cs_new_protected_nopar:Npn \spath_scale:Nn #1#2
828 {
829   \spath_scale:Nnn #1 #2
830 }
831
832 \cs_generate_variant:Nn \spath_scale:Nn {NV}
833 \cs_new_protected_nopar:Npn \spath_gscale:Nn #1#2
834 {
835   \spath_gscale:Nnn #1 #2
836 }
837
838 \cs_generate_variant:Nn \spath_gscale:Nn {NV}

```

(End definition for \spath_scale:Nnnn and others. These functions are documented on page ??.)

`\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
839 \cs_new_protected_nopar:Npn \__spath_transform:nnnnnnn #1#2#3#4#5#6#7
840 {
841   \group_begin:
842   \tl_set:Nn \l__spath_tmpa_tl {#1}
843   \tl_clear:N \l__spath_tmpb_tl
844   \bool_until_do:nn {
845     \tl_if_empty_p:N \l__spath_tmpa_tl

```

```

846 }
847 {
848   \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
849   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
850   \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
851   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
852   \tl_set:Nx \l__spath_tmpd_tl {\tl_head:N \l__spath_tmpa_tl}
853   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
854
855   \fp_set:Nn \l__spath_tmpa_fp {\l__spath_tmpc_tl * #2 + \l__spath_tmpd_tl * #4 + #6}
856   \fp_set:Nn \l__spath_tmpb_fp {\l__spath_tmpc_tl * #3 + \l__spath_tmpd_tl * #5 + #7}
857   \tl_put_right:Nx \l__spath_tmpb_tl
858   {
859     {\fp_to_dim:N \l__spath_tmpa_fp}
860     {\fp_to_dim:N \l__spath_tmpb_fp}
861   }
862 }
863
864 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
865 \group_end:
866 }
867 \cs_new_protected_nopar:Npn \spath_transform:Nnnnnnnn #1#2#3#4#5#6#7#8
868 {
869   \__spath_transform:nnnnnnn {#2}{#3}{#4}{#5}{#6}{#7}{#8}
870   \tl_set_eq:NN #1 \g__spath_output_tl
871   \tl_gclear:N \g__spath_output_tl
872 }
873 \cs_generate_variant:Nn \spath_transform:Nnnnnnnn {NVnnnnnn, Nnxxxxxx, cnnnnnnn}
874 \cs_new_protected_nopar:Npn \spath_transform:Nnnnnnnn #1#2#3#4#5#6#7
875 {
876   \spath_transform:NVnnnnnn #1#1{#2}{#3}{#4}{#5}{#6}{#7}
877 }
878 \cs_generate_variant:Nn \spath_transform:Nnnnnnnn {cnnnnnnn}
879 \cs_new_protected_nopar:Npn \spath_transform:Nnn #1#2#3
880 {
881   \spath_transform:Nnnnnnnn #1{#2}#3
882 }
883 \cs_generate_variant:Nn \spath_transform:Nnn {cnn, cVn, NVn, NnV}
884 \cs_new_protected_nopar:Npn \spath_transform:Nn #1#2
885 {
886   \spath_transform:NVnnnnnn #1#1#2
887 }
888 \cs_generate_variant:Nn \spath_transform:Nn {cn, cV, NV}
889
890 \cs_new_protected_nopar:Npn \spath_gtransform:Nnnnnnnn #1#2#3#4#5#6#7#8
891 {
892   \__spath_transform:nnnnnnn {#2}{#3}{#4}{#5}{#6}{#7}{#8}
893   \tl_gset_eq:NN #1 \g__spath_output_tl
894   \tl_gclear:N \g__spath_output_tl
895 }
896 \cs_generate_variant:Nn \spath_gtransform:Nnnnnnnn {NVnnnnnn, Nnxxxxxx, cnnnnnnn}
897 \cs_new_protected_nopar:Npn \spath_gtransform:Nnnnnnnn #1#2#3#4#5#6#7
898 {
899   \spath_gtransform:NVnnnnnn #1#1{#2}{#3}{#4}{#5}{#6}{#7}

```

```

900 }
901 \cs_generate_variant:Nn \spath_gtransform:Nnnnnnnn {cnnnnnn}
902 \cs_new_protected_nopar:Npn \spath_gtransform:Nnn #1#2#3
903 {
904   \spath_gtransform:Nnnnnnnn #1{#2}#3
905 }
906 \cs_generate_variant:Nn \spath_gtransform:Nnn {cnn, cVn, NVn, NnV}
907 \cs_new_protected_nopar:Npn \spath_gtransform:Nn #1#2
908 {
909   \spath_gtransform:NVnnnnnn #1#1#2
910 }
911 \cs_generate_variant:Nn \spath_gtransform:Nn {cn, cV, NV}

```

(End definition for `\spath_transform:Nnnnnnnn` and others. These functions are documented on page ??.)

`\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
912 \cs_new_protected_nopar:Npn \__spath_weld:nn #1#2
913 {
914   \group_begin:
915   \tl_set:Nn \l__spath_tmpa_tl {#1}
916   \tl_set:Nn \l__spath_tmpb_tl {#2}
917
918   \spath_finalpoint:NV \l__spath_tmpe_tl \l__spath_tmpa_tl
919   \spath_initialpoint:NV \l__spath_tmpd_tl \l__spath_tmpb_tl
920
921   \dim_set:Nn \l__spath_tmpa_dim
922   {
923     \tl_item:Nn \l__spath_tmpe_tl {1}
924     -
925     \tl_item:Nn \l__spath_tmpd_tl {1}
926   }
927   \dim_set:Nn \l__spath_tmpb_dim
928   {
929     \tl_item:Nn \l__spath_tmpe_tl {2}
930     -
931     \tl_item:Nn \l__spath_tmpd_tl {2}
932   }
933
934   \spath_translate:NVV \l__spath_tmpb_tl \l__spath_tmpa_dim \l__spath_tmpb_dim
935
936   \prg_replicate:nn {3}
937   {
938     \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
939   }
940
941   \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
942   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
943   \group_end:
944 }
945 \cs_new_protected_nopar:Npn \spath_weld:Nnn #1#2#3
946 {

```

```

947 \__spath_weld:nn {#2}{#3}
948 \tl_set_eq:NN #1 \g__spath_output_tl
949 \tl_gclear:N \g__spath_output_tl
950 }
951 \cs_generate_variant:Nn \spath_weld:Nnn {NVV,NVn}
952 \cs_new_protected_nopar:Npn \spath_weld:Nn #1#2
953 {
954 \spath_weld:NVn #1#1{#2}
955 }
956 \cs_generate_variant:Nn \spath_weld:Nn {NV, Nv, cV, cv}
957 \cs_new_protected_nopar:Npn \spath_gweld:Nnn #1#2#3
958 {
959 \__spath_weld:nn {#2}{#3}
960 \tl_gset_eq:NN #1 \g__spath_output_tl
961 \tl_gclear:N \g__spath_output_tl
962 }
963 \cs_generate_variant:Nn \spath_gweld:Nnn {NVV, NVn}
964 \cs_new_protected_nopar:Npn \spath_gweld:Nn #1#2
965 {
966 \spath_gweld:NVn #1#1{#2}
967 }
968 \cs_generate_variant:Nn \spath_gweld:Nn {NV, Nv, cV, cv}

```

(End definition for `\spath_weld:Nnn` and others. These functions are documented on page ??.)

```

\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
969 \cs_new_protected_nopar:Npn \__spath_append_no_move:nn #1#2
970 {
971 \group_begin:
972 \tl_set:Nn \l__spath_tmpa_tl {#1}
973 \tl_set:Nn \l__spath_tmpb_tl {#2}
974 \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
975 \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
976 \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
977
978 \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
979 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
980 \group_end:
981 }
982 \cs_new_protected_nopar:Npn \spath_append_no_move:Nnn #1#2#3
983 {
984 \__spath_append_no_move:nn {#2}{#3}
985 \tl_set_eq:NN #1 \g__spath_output_tl
986 \tl_gclear:N \g__spath_output_tl
987 }
988 \cs_generate_variant:Nn \spath_append_no_move:Nnn {NVV, NVn}
989 \cs_new_protected_nopar:Npn \spath_append_no_move:Nn #1#2
990 {
991 \spath_append_no_move:NVn #1#1{#2}
992 }
993 \cs_generate_variant:Nn \spath_append_no_move:Nn {NV, cv}
994 \cs_new_protected_nopar:Npn \spath_gappend_no_move:Nnn #1#2#3
995 {
996 \__spath_append_no_move:nn {#2}{#3}

```

```

997 \tl_gset_eq:NN #1 \g__spath_output_tl
998 \tl_gclear:N \g__spath_output_tl
999 }
1000 \cs_generate_variant:Nn \spath_gappend_no_move:Nnn {NVV, NVn}
1001 \cs_new_protected_nopar:Npn \spath_gappend_no_move:Nn #1#2
1002 {
1003   \spath_gappend_no_move:NVn #1#1{#2}
1004 }
1005 \cs_generate_variant:Nn \spath_gappend_no_move:Nn {NV, cv}

```

(End definition for `\spath_append_no_move:Nnn` and others. These functions are documented on page ??.)

```

\spath_append:Nnn Prepend the path from the second spath to the first.
\spath_append:Nn 1006 \cs_new_protected_nopar:Npn \spath_append:Nnn #1#2#3
\spath_gappend:Nnn 1007 {
\spath_gappend:Nn 1008   \tl_set:Nn #1 {#2}
1009   \tl_put_right:Nn #1 {#3}
1010 }
1011 \cs_generate_variant:Nn \spath_append:Nnn {NVV, NVn}
1012 \cs_new_protected_nopar:Npn \spath_append:Nn #1#2
1013 {
1014   \spath_append:NVn #1#1{#2}
1015 }
1016 \cs_generate_variant:Nn \spath_append:Nn {NV, Nv}
1017 \cs_new_protected_nopar:Npn \spath_gappend:Nnn #1#2#3
1018 {
1019   \tl_gset:Nn #1 {#2}
1020   \tl_gput_right:Nn #1 {#3}
1021 }
1022 \cs_generate_variant:Nn \spath_gappend:Nnn {NVV, NVn}
1023 \cs_new_protected_nopar:Npn \spath_gappend:Nn #1#2
1024 {
1025   \spath_gappend:NVn #1#1{#2}
1026 }
1027 \cs_generate_variant:Nn \spath_gappend:Nn {NV, Nv}

```

(End definition for `\spath_append:Nnn` and others. These functions are documented on page ??.)

```

\spath_prepend_no_move:Nnn Prepend the path from the second spath to the first, removing the adjoining move.
\spath_prepend_no_move:Nn 1028 \cs_new_protected_nopar:Npn \spath_prepend_no_move:Nnn #1#2#3
\spath_gprepend_no_move:Nnn 1029 {
\spath_gprepend_no_move:Nn 1030   \spath_append_no_move:Nnn #1{#3}{#2}
1031 }
1032 \cs_generate_variant:Nn \spath_prepend_no_move:Nnn {NVV, NVn}
1033 \cs_new_protected_nopar:Npn \spath_prepend_no_move:Nn #1#2
1034 {
1035   \spath_prepend_no_move:NVn #1#1{#2}
1036 }
1037 \cs_generate_variant:Nn \spath_prepend_no_move:Nn {NV, cv}
1038 \cs_new_protected_nopar:Npn \spath_gprepend_no_move:Nnn #1#2#3
1039 {
1040   \spath_gappend_no_move:Nnn #1{#3}{#2}
1041 }
1042 \cs_generate_variant:Nn \spath_gprepend_no_move:Nnn {NVV, NVn}

```

```

1043 \cs_new_protected_nopar:Npn \spath_gprepend_no_move:Nn #1#2
1044 {
1045   \spath_gprepend_no_move:NVn #1#1{#2}
1046 }
1047 \cs_generate_variant:Nn \spath_gprepend_no_move:Nn {NV, cv}

```

(End definition for `\spath_prepend_no_move:Nnn` and others. These functions are documented on page ??.)

```

\spath_prepend:Nnn Prepend the path from the second spath to the first.
\spath_prepend:Nn
\spath_gprepend:Nnn
\spath_gprepend:Nn
1048 \cs_new_protected_nopar:Npn \spath_prepend:Nnn #1#2#3
1049 {
1050   \spath_append:Nnn #1{#3}{#2}
1051 }
1052 \cs_generate_variant:Nn \spath_prepend:Nnn {NVV, NVn}
1053 \cs_new_protected_nopar:Npn \spath_prepend:Nn #1#2
1054 {
1055   \spath_prepend:NVn #1#1{#2}
1056 }
1057 \cs_generate_variant:Nn \spath_prepend:Nn {NV}
1058 \cs_new_protected_nopar:Npn \spath_gprepend:Nnn #1#2#3
1059 {
1060   \spath_gappend:Nnn #1{#3}{#2}
1061 }
1062 \cs_generate_variant:Nn \spath_gprepend:Nnn {NVV, NVn}
1063 \cs_new_protected_nopar:Npn \spath_gprepend:Nn #1#2
1064 {
1065   \spath_gprepend:NVn #1#1{#2}
1066 }
1067 \cs_generate_variant:Nn \spath_gprepend:Nn {NV}

```

(End definition for `\spath_prepend:Nnn` and others. These functions are documented on page ??.)

```

\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
\spath_gbake_round:N
1068 \cs_new_protected_nopar:Npn \__spath_bake_round:n #1
1069 {
1070   \group_begin:
1071   \tl_set:Nn \l__spath_tmpa_tl {#1}
1072   \pgf@@processround \l__spath_tmpa_tl\l__spath_tmpb_tl
1073   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
1074   \group_end:
1075 }
1076 \cs_new_protected_nopar:Npn \spath_bake_round:Nn #1#2
1077 {
1078   \__spath_bake_round:n {#2}
1079   \tl_set_eq:NN #1 \g__spath_output_tl
1080   \tl_gclear:N \g__spath_output_tl
1081 }
1082 \cs_generate_variant:Nn \spath_bake_round:Nn {NV}
1083 \cs_new_protected_nopar:Npn \spath_bake_round:N #1
1084 {
1085   \spath_bake_round:NV #1#1
1086 }

```



```

1087 \cs_generate_variant:Nn \spath_bake_round:N {c}
1088 \cs_new_protected_nopar:Npn \spath_gbake_round:Nn #1#2
1089 {
1090   \__spath_bake_round:n {#2}
1091   \tl_gset_eq:NN #1 \g__spath_output_tl
1092   \tl_gclear:N \g__spath_output_tl
1093 }
1094 \cs_generate_variant:Nn \spath_gbake_round:Nn {NV}
1095 \cs_new_protected_nopar:Npn \spath_gbake_round:N #1
1096 {
1097   \spath_gbake_round:NV #1#1
1098 }
1099 \cs_generate_variant:Nn \spath_gbake_round:N {c}

```

(End definition for `\spath_bake_round:Nn` and others. These functions are documented on page ??.)

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

```

\spath_close:Nn
\spath_close:N
\spath_gclose:Nn
\spath_gclose:N
1100 \cs_new_protected_nopar:Npn \__spath_close:n #1
1101 {
1102   \group_begin:
1103   \tl_set:Nn \l__spath_tmpa_tl {#1}
1104   \spath_initialpoint:NV \l__spath_tmpb_tl \l__spath_tmpa_tl
1105   \tl_put_right:NV \l__spath_tmpa_tl \c_spath_closepath_tl
1106   \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
1107   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
1108   \group_end:
1109 }
1110 \cs_new_protected_nopar:Npn \spath_close:Nn #1#2
1111 {
1112   \__spath_close:n {#2}
1113   \tl_set_eq:NN #1 \g__spath_output_tl
1114   \tl_gclear:N \g__spath_output_tl
1115 }
1116 \cs_generate_variant:Nn \spath_close:Nn {NV}
1117 \cs_new_protected_nopar:Npn \spath_close:N #1
1118 {
1119   \spath_close:NV #1#1
1120 }
1121 \cs_generate_variant:Nn \spath_close:N {c}
1122 \cs_new_protected_nopar:Npn \spath_gclose:Nn #1#2
1123 {
1124   \__spath_close:n {#2}
1125   \tl_gset_eq:NN #1 \g__spath_output_tl
1126   \tl_gclear:N \g__spath_output_tl
1127 }
1128 \cs_generate_variant:Nn \spath_gclose:Nn {NV}
1129 \cs_new_protected_nopar:Npn \spath_gclose:N #1
1130 {
1131   \spath_gclose:NV #1#1
1132 }
1133 \cs_generate_variant:Nn \spath_gclose:N {c}

```

(End definition for `\spath_close:Nn` and others. These functions are documented on page ??.)

```

\spath_open:Nn Removes all close paths from the path, replacing them by lineto if they move any
\spath_open:N distance.
\spath_gopen:Nn 1134 \cs_new_protected_nopar:Npn \__spath_open:n #1
\spath_gopen:N 1135 {
1136   \group_begin:
1137   \tl_set:Nn \l__spath_tmpa_tl {#1}
1138   \tl_clear:N \l__spath_tmpb_tl
1139   \bool_until_do:nn {
1140     \tl_if_empty_p:N \l__spath_tmpa_tl
1141   }
1142   {
1143     \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
1144     \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1145
1146     \tl_case:NnF \l__spath_tmpc_tl
1147     {
1148       \c_spath_closepath_tl {
1149
1150         \bool_if:nF
1151         {
1152           \dim_compare_p:n
1153           {
1154             \l__spath_move_x_dim == \l__spath_tmpa_dim
1155           }
1156           &&
1157           \dim_compare_p:n
1158           {
1159             \l__spath_move_y_dim == \l__spath_tmpb_dim
1160           }
1161         }
1162         {
1163           \tl_put_right:NV \l__spath_tmpb_tl \c_spath_lineto_tl
1164
1165           \tl_put_right:Nx \l__spath_tmpb_tl {
1166             { \dim_use:N \l__spath_move_x_dim }
1167             { \dim_use:N \l__spath_move_y_dim }
1168           }
1169         }
1170
1171         \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
1172         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1173         \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
1174         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1175       }
1176
1177       \c_spath_moveto_tl {
1178         \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpc_tl
1179
1180         \dim_set:Nn \l__spath_move_x_dim {\tl_head:N \l__spath_tmpa_tl}
1181         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1182         \dim_set:Nn \l__spath_move_y_dim {\tl_head:N \l__spath_tmpa_tl}
1183         \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1184
1185         \tl_put_right:Nx \l__spath_tmpb_tl {

```

```

1186     { \dim_use:N \l__spath_move_x_dim }
1187     { \dim_use:N \l__spath_move_y_dim }
1188   }
1189
1190   \dim_set_eq:NN \l__spath_tmpa_dim \l__spath_move_x_dim
1191   \dim_set_eq:NN \l__spath_tmpb_dim \l__spath_move_y_dim
1192 }
1193 }
1194 {
1195   \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpc_tl
1196
1197   \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
1198   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1199   \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
1200   \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
1201
1202   \tl_put_right:Nx \l__spath_tmpb_tl {
1203     { \dim_use:N \l__spath_tmpa_dim }
1204     { \dim_use:N \l__spath_tmpb_dim }
1205   }
1206 }
1207 }
1208 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
1209 \group_end:
1210 }
1211 \cs_new_protected_nopar:Npn \spath_open:Nn #1#2
1212 {
1213   \__spath_open:n {#2}
1214   \tl_set_eq:NN #1 \g__spath_output_tl
1215   \tl_gclear:N \g__spath_output_tl
1216 }
1217 \cs_generate_variant:Nn \spath_open:Nn {NV}
1218 \cs_new_protected_nopar:Npn \spath_open:N #1
1219 {
1220   \spath_open:NV #1#1
1221 }
1222 \cs_new_protected_nopar:Npn \spath_gopen:Nn #1#2
1223 {
1224   \__spath_open:n {#2}
1225   \tl_gset_eq:NN #1 \g__spath_output_tl
1226   \tl_gclear:N \g__spath_output_tl
1227 }
1228 \cs_generate_variant:Nn \spath_gopen:Nn {NV}
1229 \cs_new_protected_nopar:Npn \spath_gopen:N #1
1230 {
1231   \spath_gopen:NV #1#1
1232 }

```

(End definition for \spath_open:Nn and others. These functions are documented on page ??.)

```

\spath_remove_empty_components:Nn Remove any component that is simply a moveto.
\spath_remove_empty_components:N 1233 \cs_new_protected_nopar:Npn \__spath_remove_empty_components:n #1
\spath_gremove_empty_components:Nn 1234 {
\spath_gremove_empty_components:N 1235 \group_begin:

```

```

1236 \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
1237 \tl_clear:N \l__spath_tmpa_tl
1238 \seq_map_inline:Nn \l__spath_tmpa_seq
1239 {
1240   \int_compare:nF
1241   {
1242     \tl_count:n {##1} == 3
1243   }
1244   {
1245     \tl_put_right:Nn \l__spath_tmpa_tl {##1}
1246   }
1247 }
1248 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
1249 \group_end:
1250 }
1251 \cs_new_protected_nopar:Npn \spath_remove_empty_components:Nn #1#2
1252 {
1253   \__spath_remove_empty_components:n {#2}
1254   \tl_set_eq:NN #1 \g__spath_output_tl
1255   \tl_gclear:N \g__spath_output_tl
1256 }
1257 \cs_generate_variant:Nn \spath_remove_empty_components:Nn {NV}
1258 \cs_new_protected_nopar:Npn \spath_remove_empty_components:N #1
1259 {
1260   \spath_remove_empty_components:NV #1#1
1261 }
1262 \cs_generate_variant:Nn \spath_remove_empty_components:N {c}
1263 \cs_new_protected_nopar:Npn \spath_gremove_empty_components:Nn #1#2
1264 {
1265   \__spath_remove_empty_components:n {#2}
1266   \tl_gset_eq:NN #1 \g__spath_output_tl
1267   \tl_gclear:N \g__spath_output_tl
1268 }
1269 \cs_generate_variant:Nn \spath_gremove_empty_components:Nn {NV}
1270 \cs_new_protected_nopar:Npn \spath_gremove_empty_components:N #1
1271 {
1272   \spath_gremove_empty_components:NV #1#1
1273 }
1274 \cs_generate_variant:Nn \spath_gremove_empty_components:N {c}

```

(End definition for \spath_remove_empty_components:Nn and others. These functions are documented on page ??.)

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

```

1275 \prg_new_protected_conditional:Npnn \spath_if_eq:nn #1#2 { T, F, TF }
1276 {
1277   \group_begin:
1278   \tl_set:Nn \l__spath_tmpa_tl {#1}
1279   \tl_set:Nn \l__spath_tmpb_tl {#2}
1280   \bool_gset_true:N \g__spath_tmpa_bool
1281   \int_compare:nNnTF {\tl_count:N \l__spath_tmpa_tl} = {\tl_count:N \l__spath_tmpb_tl}
1282   {
1283     \int_step_inline:nnnn {1} {3} {\tl_count:N \l__spath_tmpa_tl}
1284     {

```

```

1285 \tl_set:Nx \l__spath_tmpc_tl {\tl_item:Nn \l__spath_tmpa_tl {##1}}
1286 \tl_set:Nx \l__spath_tmpd_tl {\tl_item:Nn \l__spath_tmpb_tl {##1}}
1287 \tl_if_eq:NNF \l__spath_tmpc_tl \l__spath_tmpd_tl
1288 {
1289   \bool_gset_false:N \g__spath_tmpa_bool
1290 }
1291 \dim_set:Nn \l__spath_tmpa_dim {\tl_item:Nn \l__spath_tmpa_tl {##1+1}}
1292 \dim_set:Nn \l__spath_tmpb_dim {\tl_item:Nn \l__spath_tmpb_tl {##1+1}}
1293 \dim_compare:nF { \dim_abs:n { \l__spath_tmpa_dim - \l__spath_tmpb_dim } < 0.001pt }
1294 {
1295   \bool_gset_false:N \g__spath_tmpa_bool
1296 }
1297 \dim_set:Nn \l__spath_tmpa_dim {\tl_item:Nn \l__spath_tmpa_tl {##1+2}}
1298 \dim_set:Nn \l__spath_tmpb_dim {\tl_item:Nn \l__spath_tmpb_tl {##1+2}}
1299 \dim_compare:nF { \dim_abs:n { \l__spath_tmpa_dim - \l__spath_tmpb_dim } < 0.001pt }
1300 {
1301   \bool_gset_false:N \g__spath_tmpa_bool
1302 }
1303 }
1304 }
1305 {
1306   \bool_gset_false:N \g__spath_tmpa_bool
1307 }
1308 \group_end:
1309 \bool_if:NTF \g__spath_tmpa_bool
1310 {
1311   \prg_return_true:
1312 }
1313 {
1314   \prg_return_false:
1315 }
1316 }
1317 \prg_generate_conditional_variant:Nnn \spath_if_eq:nn {VV, Vn, nV, vv} {TF, T, F}

```

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

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

```

1318 \cs_new_protected_nopar:Npn \__spath_split_curve:nn #1#2
1319 {
1320   \group_begin:
1321   \tl_set_eq:NN \l__spath_tmpa_tl \c_spath_moveto_tl
1322   \tl_put_right:Nx \l__spath_tmpa_tl {
1323     {\tl_item:nn {#1} {2}}
1324     {\tl_item:nn {#1} {3}}
1325   }
1326   \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curvetoa_tl
1327   \tl_put_right:Nx \l__spath_tmpa_tl
1328   {
1329     {\fp_to_dim:n
1330     {
1331       (1 - #2) * \tl_item:nn {#1} {2} + (#2) * \tl_item:nn {#1} {5}
1332     }}

```

```

1333     {\fp_to_dim:n
1334     {
1335       (1 - #2) * \tl_item:nn {#1} {3} + (#2) * \tl_item:nn {#1} {6}
1336     }}
1337   }
1338
1339   \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curvetob_tl
1340   \tl_put_right:Nx \l__spath_tmpa_tl
1341   {
1342     {\fp_to_dim:n
1343     {
1344       (1 - #2)^2 * \tl_item:nn {#1} {2} + 2 * (1 - #2) * (#2) * \tl_item:nn {#1} {5} + (#2)^2 * \tl_item:nn {#1} {6}
1345     }}
1346     {\fp_to_dim:n
1347     {
1348       (1 - #2)^2 * \tl_item:nn {#1} {3} + 2 * (1 - #2) * (#2) * \tl_item:nn {#1} {6} + (#2)^2 * \tl_item:nn {#1} {6}
1349     }}
1350   }
1351
1352   \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curveto_tl
1353   \tl_put_right:Nx \l__spath_tmpa_tl
1354   {
1355     {\fp_to_dim:n
1356     {
1357       (1 - #2)^3 * \tl_item:nn {#1} {2} + 3 * (1 - #2)^2 * (#2) * \tl_item:nn {#1} {5} + 3 * (#2)^2 * \tl_item:nn {#1} {6}
1358     }}
1359     {\fp_to_dim:n
1360     {
1361       (1 - #2)^3 * \tl_item:nn {#1} {3} + 3 * (1 - #2)^2 * (#2) * \tl_item:nn {#1} {6} + 3 * (#2)^2 * \tl_item:nn {#1} {6}
1362     }}
1363   }
1364
1365   \tl_gclear:N \g__spath_output_tl
1366   \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpa_tl
1367
1368   \tl_clear:N \l__spath_tmpa_tl
1369   \tl_set_eq:NN \l__spath_tmpa_tl \c_spath_moveto_tl
1370   \tl_put_right:Nx \l__spath_tmpa_tl
1371   {
1372     {\fp_to_dim:n
1373     {
1374       (1 - #2)^3 * \tl_item:nn {#1} {2} + 3 * (1 - #2)^2 * (#2) * \tl_item:nn {#1} {5} + 3 * (#2)^2 * \tl_item:nn {#1} {6}
1375     }}
1376     {\fp_to_dim:n
1377     {
1378       (1 - #2)^3 * \tl_item:nn {#1} {3} + 3 * (1 - #2)^2 * (#2) * \tl_item:nn {#1} {6} + 3 * (#2)^2 * \tl_item:nn {#1} {6}
1379     }}
1380   }
1381
1382   \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curvetoa_tl
1383   \tl_put_right:Nx \l__spath_tmpa_tl
1384   {
1385     {\fp_to_dim:n
1386     {

```

```

1387     (1 - #2)^2 * \tl_item:nn {#1} {5} + 2 * (1 - #2) * (#2) * \tl_item:nn {#1} {8} + (#2)^
1388   }}
1389   {\fp_to_dim:n
1390   {
1391     (1 - #2)^2 * \tl_item:nn {#1} {6} + 2 * (1 - #2) * (#2) * \tl_item:nn {#1} {9} + (#2)^
1392   }}
1393 }
1394 \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curvetob_tl
1395 \tl_put_right:Nx \l__spath_tmpa_tl
1396 {
1397   {\fp_to_dim:n
1398   {
1399     (1 - #2) * \tl_item:nn {#1} {8} + (#2) * \tl_item:nn {#1} {11}
1400   }}
1401   {\fp_to_dim:n
1402   {
1403     (1 - #2) * \tl_item:nn {#1} {9} + (#2) * \tl_item:nn {#1} {12}
1404   }}
1405 }
1406 \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curveto_tl
1407 \tl_put_right:Nx \l__spath_tmpa_tl {
1408   {\tl_item:nn {#1} {11}}
1409   {\tl_item:nn {#1} {12}}
1410 }
1411
1412 \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpa_tl
1413 \group_end:
1414 }
1415 \cs_new_protected_nopar:Npn \spath_split_curve:NNnn #1#2#3#4
1416 {
1417   \__spath_split_curve:nn {#3}{#4}
1418   \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
1419   \tl_set:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
1420   \tl_gclear:N \g__spath_output_tl
1421 }
1422 \cs_generate_variant:Nn \spath_split_curve:NNnn {NNnV, NNVn, NNVV}
1423 \cs_new_protected_nopar:Npn \spath_gsplit_curve:NNnn #1#2#3#4
1424 {
1425   \__spath_split_curve:nn {#3}{#4}
1426   \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
1427   \tl_gset:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
1428   \tl_gclear:N \g__spath_output_tl
1429 }
1430 \cs_generate_variant:Nn \spath_gsplit_curve:NNnn {NNnV, NNVn, NNVV}

```

(End definition for `\spath_split_curve:NNnn` and `\spath_gsplit_curve:NNnn`. These functions are documented on page ??.)

`\spath_split_line:NNnn` Splits a line segment.

```

\spath_gsplit_line:NNnn
1431 \cs_new_protected_nopar:Npn \__spath_split_line:nn #1#2
1432 {
1433   \group_begin:
1434   \tl_set_eq:NN \l__spath_tmpa_tl \c_spath_moveto_tl
1435   \tl_put_right:Nx \l__spath_tmpa_tl {

```

```

1436     {\tl_item:nn {#1} {2}}
1437     {\tl_item:nn {#1} {3}}
1438 }
1439 \tl_put_right:NV \l__spath_tmpa_tl \c_spath_lineto_tl
1440 \tl_put_right:Nx \l__spath_tmpa_tl
1441 {
1442     {\fp_to_dim:n
1443     {
1444         (1 - #2) * \tl_item:nn {#1} {2} + (#2) * \tl_item:nn {#1} {5}
1445     }}
1446     {\fp_to_dim:n
1447     {
1448         (1 - #2) * \tl_item:nn {#1} {3} + (#2) * \tl_item:nn {#1} {6}
1449     }}
1450 }
1451 \tl_gclear:N \g__spath_output_tl
1452 \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpa_tl
1453
1454 \tl_clear:N \l__spath_tmpa_tl
1455 \tl_set_eq:NN \l__spath_tmpa_tl \c_spath_moveto_tl
1456 \tl_put_right:Nx \l__spath_tmpa_tl
1457 {
1458     {\fp_to_dim:n
1459     {
1460         (1 - #2) * \tl_item:nn {#1} {2} + (#2) * \tl_item:nn {#1} {5}
1461     }}
1462     {\fp_to_dim:n
1463     {
1464         (1 - #2) * \tl_item:nn {#1} {3} + (#2) * \tl_item:nn {#1} {6}
1465     }}
1466 }
1467 \tl_put_right:NV \l__spath_tmpa_tl \c_spath_lineto_tl
1468 \tl_put_right:Nx \l__spath_tmpa_tl {
1469     {\tl_item:nn {#1} {5}}
1470     {\tl_item:nn {#1} {6}}
1471 }
1472
1473 \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpa_tl
1474 \group_end:
1475 }
1476 \cs_new_protected_nopar:Npn \spath_split_line:NNnn #1#2#3#4
1477 {
1478     \__spath_split_line:nn {#3}{#4}
1479     \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
1480     \tl_set:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
1481     \tl_gclear:N \g__spath_output_tl
1482 }
1483 \cs_generate_variant:Nn \spath_split_line:NNnn {NNnV, NNVn, NNVV}
1484 \cs_new_protected_nopar:Npn \spath_gsplit_line:NNnn #1#2#3#4
1485 {
1486     \__spath_split_line:nn {#3}{#4}
1487     \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
1488     \tl_gset:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
1489     \tl_gclear:N \g__spath_output_tl

```



```

1490 }
1491 \cs_generate_variant:Nn \spath_gsplit_line:NNnn {NNnV, NNVn, NNVV}

```

(End definition for `\spath_split_line:NNnn` and `\spath_gsplit_line:NNnn`. These functions are documented on page ??.)

```

1492 \int_new:N \l__spath_split_int
1493 \int_new:N \l__spath_splitat_int
1494 \fp_new:N \l__spath_split_fp
1495 \bool_new:N \l__spath_split_bool
1496 \tl_new:N \l__spath_split_path_tl
1497 \tl_new:N \l__spath_split_patha_tl
1498 \tl_new:N \l__spath_split_pathb_tl
1499 \tl_new:N \l__spath_split_intoa_tl
1500 \tl_new:N \l__spath_split_intob_tl
1501 \dim_new:N \l__spath_splitx_dim
1502 \dim_new:N \l__spath_splity_dim

```

`\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_split_at:NNn
\spath_gsplit_at:Nn
1503 \cs_new_protected_nopar:Npn \__spath_split_at:nn #1#2
1504 {
1505   \group_begin:
1506   \int_set:Nn \l__spath_splitat_int {\fp_to_int:n {floor(#2) + 1}}
1507   \fp_set:Nn \l__spath_split_fp {#2 - floor(#2)}
1508
1509   % Is split point near one end or other of a component?
1510   \fp_compare:nT
1511   {
1512     \l__spath_split_fp < 0.01
1513   }
1514   {
1515     % Near the start, so we'll place it at the start
1516     \fp_set:Nn \l__spath_split_fp {0}
1517   }
1518   \fp_compare:nT
1519   {
1520     \l__spath_split_fp > 0.99
1521   }
1522   {
1523     % Near the end, so we'll place it at the end
1524     \fp_set:Nn \l__spath_split_fp {0}
1525     \int_incr:N \l__spath_splitat_int
1526   }
1527
1528   \int_zero:N \l__spath_split_int
1529   \bool_set_true:N \l__spath_split_bool
1530
1531   \tl_set:Nn \l__spath_split_path_tl {#1}
1532   \tl_clear:N \l__spath_split_patha_tl
1533
1534   \dim_zero:N \l__spath_splitx_dim
1535   \dim_zero:N \l__spath_splity_dim
1536

```

```

1537 \bool_until_do:nn {
1538   \tl_if_empty_p:N \l__spath_split_path_tl
1539   ||
1540   \int_compare_p:n { \l__spath_splitat_int == \l__spath_split_int }
1541 }
1542 {
1543   \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_split_path_tl}
1544   \tl_set:Nx \l__spath_split_path_tl {\tl_tail:N \l__spath_split_path_tl }
1545   \tl_case:Nn \l__spath_tmpc_tl
1546   {
1547     \c_spath_lineto_tl
1548     {
1549       \int_incr:N \l__spath_split_int
1550     }
1551     \c_spath_curvetoa_tl
1552     {
1553       \int_incr:N \l__spath_split_int
1554     }
1555   }
1556   \int_compare:nT { \l__spath_split_int < \l__spath_splitat_int }
1557   {
1558     \tl_put_right:NV \l__spath_split_patha_tl \l__spath_tmpc_tl
1559
1560     \tl_put_right:Nx \l__spath_split_patha_tl
1561     {{ \tl_head:N \l__spath_split_path_tl }}
1562     \dim_set:Nn \l__spath_splitx_dim {\tl_head:N \l__spath_split_path_tl}
1563     \tl_set:Nx \l__spath_split_path_tl {\tl_tail:N \l__spath_split_path_tl }
1564
1565     \tl_put_right:Nx \l__spath_split_patha_tl
1566     {{ \tl_head:N \l__spath_split_path_tl }}
1567     \dim_set:Nn \l__spath_splity_dim {\tl_head:N \l__spath_split_path_tl}
1568     \tl_set:Nx \l__spath_split_path_tl {\tl_tail:N \l__spath_split_path_tl }
1569
1570   }
1571 }
1572
1573 \tl_clear:N \l__spath_split_pathb_tl
1574 \tl_put_right:NV \l__spath_split_pathb_tl \c_spath_moveto_tl
1575 \tl_put_right:Nx \l__spath_split_pathb_tl
1576 {
1577   {\dim_use:N \l__spath_splitx_dim}
1578   {\dim_use:N \l__spath_splity_dim}
1579 }
1580
1581 \fp_compare:nTF
1582 {
1583   \l__spath_split_fp == 0
1584 }
1585 {
1586   \tl_set_eq:NN \l__spath_split_intob_tl \l__spath_split_pathb_tl
1587   \tl_if_empty:NF \l__spath_split_path_tl
1588   {
1589     \tl_put_right:NV \l__spath_split_intob_tl \l__spath_tmpc_tl
1590     \tl_put_right:NV \l__spath_split_intob_tl \l__spath_split_path_tl

```

```

1591     }
1592   }
1593   {
1594
1595   \tl_case:Nn \l__spath_tmpe_tl
1596   {
1597     \c_spath_lineto_tl
1598     {
1599       \tl_put_right:NV \l__spath_split_pathb_tl \l__spath_tmpe_tl
1600       \tl_put_right:Nx \l__spath_split_pathb_tl
1601       {{ \tl_head:N \l__spath_split_path_tl }}
1602       \tl_set:Nx \l__spath_split_path_tl {\tl_tail:N \l__spath_split_path_tl }
1603
1604       \tl_put_right:Nx \l__spath_split_pathb_tl
1605       {{ \tl_head:N \l__spath_split_path_tl }}
1606       \tl_set:Nx \l__spath_split_path_tl {\tl_tail:N \l__spath_split_path_tl }
1607
1608       \spath_split_line:NNVV
1609       \l__spath_split_intoa_tl
1610       \l__spath_split_intob_tl
1611       \l__spath_split_pathb_tl
1612       \l__spath_split_fp
1613
1614       \prg_replicate:nn {3} {
1615         \tl_set:Nx \l__spath_split_intoa_tl {\tl_tail:N \l__spath_split_intoa_tl}
1616       }
1617
1618       \tl_put_right:NV \l__spath_split_patha_tl \l__spath_split_intoa_tl
1619       \tl_put_right:NV \l__spath_split_intob_tl \l__spath_split_path_tl
1620     }
1621   \c_spath_curvetoa_tl
1622   {
1623     \tl_put_right:NV \l__spath_split_pathb_tl \l__spath_tmpe_tl
1624     \tl_put_right:Nx \l__spath_split_pathb_tl
1625     {{ \tl_head:N \l__spath_split_path_tl }}
1626     \tl_set:Nx \l__spath_split_path_tl {\tl_tail:N \l__spath_split_path_tl }
1627
1628     \tl_put_right:Nx \l__spath_split_pathb_tl
1629     {{ \tl_head:N \l__spath_split_path_tl }}
1630     \tl_set:Nx \l__spath_split_path_tl {\tl_tail:N \l__spath_split_path_tl }
1631
1632     \prg_replicate:nn {2} {
1633
1634       \tl_put_right:Nx \l__spath_split_pathb_tl
1635       { \tl_head:N \l__spath_split_path_tl }
1636       \tl_set:Nx \l__spath_split_path_tl {\tl_tail:N \l__spath_split_path_tl }
1637
1638       \tl_put_right:Nx \l__spath_split_pathb_tl
1639       {{ \tl_head:N \l__spath_split_path_tl }}
1640       \tl_set:Nx \l__spath_split_path_tl {\tl_tail:N \l__spath_split_path_tl }
1641
1642       \tl_put_right:Nx \l__spath_split_pathb_tl
1643       {{ \tl_head:N \l__spath_split_path_tl }}
1644       \tl_set:Nx \l__spath_split_path_tl {\tl_tail:N \l__spath_split_path_tl }

```

```

1645     }
1646
1647     \spath_split_curve:NNVV
1648     \l__spath_split_intoa_tl
1649     \l__spath_split_intob_tl
1650     \l__spath_split_pathb_tl \l__spath_split_fp
1651
1652     \prg_replicate:nn {3} {
1653       \tl_set:Nx \l__spath_split_intoa_tl {\tl_tail:N \l__spath_split_intoa_tl}
1654     }
1655
1656     \tl_put_right:NV \l__spath_split_patha_tl \l__spath_split_intoa_tl
1657     \tl_put_right:NV \l__spath_split_intob_tl \l__spath_split_patha_tl
1658   }
1659 }
1660 }
1661
1662 \tl_gclear:N \g__spath_output_tl
1663 \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_split_patha_tl
1664 \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_split_intob_tl
1665 \group_end:
1666 }
1667 \cs_new_protected_nopar:Npn \spath_split_at:NNnn #1#2#3#4
1668 {
1669   \__spath_split_at:nn {#3}{#4}
1670   \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
1671   \tl_set:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
1672   \tl_gclear:N \g__spath_output_tl
1673 }
1674 \cs_generate_variant:Nn \spath_split_at:NNnn {NNVn, NNVV, NNnV}
1675 \cs_new_protected_nopar:Npn \spath_gsplit_at:NNnn #1#2#3#4
1676 {
1677   \__spath_split_at:nn {#3}{#4}
1678   \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
1679   \tl_gset:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
1680   \tl_gclear:N \g__spath_output_tl
1681 }
1682 \cs_generate_variant:Nn \spath_gsplit_at:NNnn {NNVn, NNVV, NNnV}
1683 \cs_new_protected_nopar:Npn \spath_split_at:Nnn #1#2#3
1684 {
1685   \__spath_split_at:nn {#2}{#3}
1686   \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}} \tl_item:Nn \g__spath_output_tl {2}}
1687   \tl_gclear:N \g__spath_output_tl
1688 }
1689 \cs_generate_variant:Nn \spath_split_at:Nnn {NVn, NVV}
1690 \cs_new_protected_nopar:Npn \spath_split_at:Nn #1#2
1691 {
1692   \spath_split_at:NVn #1#1{#2}
1693 }
1694 \cs_new_protected_nopar:Npn \spath_gsplit_at:Nnn #1#2#3
1695 {
1696   \__spath_split_at:nn {#2}{#3}
1697   \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}} \tl_item:Nn \g__spath_output_tl {2}}
1698   \tl_gclear:N \g__spath_output_tl

```

```

1699 }
1700 \cs_generate_variant:Nn \spath_gsplit_at:Nnn {NVn, NVV}
1701 \cs_new_protected_nopar:Npn \spath_gsplit_at:Nn #1#2
1702 {
1703   \spath_gsplit_at:NVn #1#1{#2}
1704 }

```

(End definition for `\spath_split_at:Nnn` and others. These functions are documented on page ??.)

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.

```

1705 \tl_new:N \l__spath_shorten_fa_tl
1706 \tl_new:N \l__spath_shorten_path_tl
1707 \tl_new:N \l__spath_shorten_last_tl
1708 \tl_new:N \l__spath_shorten_lasta_tl
1709 \tl_new:N \l__spath_shorten_lastb_tl
1710 \int_new:N \l__spath_shorten_int
1711 \fp_new:N \l__spath_shorten_x_fp
1712 \fp_new:N \l__spath_shorten_y_fp
1713 \fp_new:N \l__spath_shorten_len_fp

```

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

```

1714 \cs_new_protected_nopar:Npn \__spath_shorten_at_end:nn #1#2
1715 {
1716   \int_compare:nTF
1717   {
1718     \tl_count:n {#1} > 3
1719   }
1720   {
1721     \group_begin:
1722     \tl_set:Nn \l__spath_shorten_path_tl {#1}
1723     \tl_reverse:N \l__spath_shorten_path_tl
1724
1725     \tl_set:Nx \l__spath_shorten_fa_tl {\tl_item:Nn \l__spath_shorten_path_tl {3}}
1726
1727     \tl_clear:N \l__spath_shorten_last_tl
1728     \tl_if_eq:NNTF \l__spath_shorten_fa_tl \c_spath_curveto_tl
1729     {
1730       \int_set:Nn \l__spath_shorten_int {3}
1731     }
1732     {
1733       \int_set:Nn \l__spath_shorten_int {1}
1734     }
1735
1736     \prg_replicate:nn { \l__spath_shorten_int }
1737     {
1738       \tl_put_right:Nx \l__spath_shorten_last_tl
1739       {

```

```

1740     {\tl_head:N \l__spath_shorten_path_tl}
1741   }
1742   \tl_set:Nx \l__spath_shorten_path_tl {\tl_tail:N \l__spath_shorten_path_tl}
1743   \tl_put_right:Nx \l__spath_shorten_last_tl
1744   {
1745     {\tl_head:N \l__spath_shorten_path_tl}
1746   }
1747   \tl_set:Nx \l__spath_shorten_path_tl {\tl_tail:N \l__spath_shorten_path_tl}
1748   \tl_put_right:Nx \l__spath_shorten_last_tl
1749   {
1750     \tl_head:N \l__spath_shorten_path_tl
1751   }
1752   \tl_set:Nx \l__spath_shorten_path_tl {\tl_tail:N \l__spath_shorten_path_tl}
1753 }
1754
1755 \tl_put_right:Nx \l__spath_shorten_last_tl
1756 {
1757   {\tl_item:Nn \l__spath_shorten_path_tl {1}}
1758   {\tl_item:Nn \l__spath_shorten_path_tl {2}}
1759 }
1760 \tl_put_right:NV \l__spath_shorten_last_tl \c_spath_moveto_tl
1761
1762 \tl_reverse:N \l__spath_shorten_path_tl
1763
1764 \fp_set:Nn \l__spath_shorten_x_fp
1765 {
1766   \dim_to_fp:n {\tl_item:Nn \l__spath_shorten_last_tl {4}}
1767   -
1768   \dim_to_fp:n {\tl_item:Nn \l__spath_shorten_last_tl {1}}
1769 }
1770
1771 \fp_set:Nn \l__spath_shorten_y_fp
1772 {
1773   \dim_to_fp:n {\tl_item:Nn \l__spath_shorten_last_tl {5}}
1774   -
1775   \dim_to_fp:n {\tl_item:Nn \l__spath_shorten_last_tl {2}}
1776 }
1777
1778 \fp_set:Nn \l__spath_shorten_len_fp
1779 {
1780   sqrt( \l__spath_shorten_x_fp * \l__spath_shorten_x_fp + \l__spath_shorten_y_fp * \l__
1781 )
1782
1783 \fp_compare:nTF
1784 {
1785   \l__spath_shorten_len_fp > #2
1786 }
1787 {
1788
1789   \fp_set:Nn \l__spath_shorten_len_fp
1790   {
1791     (\l__spath_shorten_len_fp - #2)/ \l__spath_shorten_len_fp
1792   }
1793

```

```

1794 \tl_reverse:N \l__spath_shorten_last_tl
1795
1796 \tl_if_eq:NNTF \l__spath_shorten_fa_tl \c_spath_curveto_tl
1797 {
1798   \fp_set:Nn \l__spath_shorten_len_fp
1799   {
1800     1 - (1 - \l__spath_shorten_len_fp)/3
1801   }
1802   \spath_split_curve:NNVV
1803   \l__spath_shorten_lasta_tl
1804   \l__spath_shorten_lastb_tl
1805   \l__spath_shorten_last_tl
1806   \l__spath_shorten_len_fp
1807 }
1808 {
1809   \spath_split_line:NNVV
1810   \l__spath_shorten_lasta_tl
1811   \l__spath_shorten_lastb_tl
1812   \l__spath_shorten_last_tl
1813   \l__spath_shorten_len_fp
1814 }
1815
1816 \prg_replicate:nn {3}
1817 {
1818   \tl_set:Nx \l__spath_shorten_lasta_tl {\tl_tail:N \l__spath_shorten_lasta_tl}
1819 }
1820
1821 \tl_put_right:NV \l__spath_shorten_path_tl \l__spath_shorten_lasta_tl
1822
1823 }
1824 {
1825
1826   \int_compare:nT
1827   {
1828     \tl_count:N \l__spath_shorten_path_tl > 3
1829   }
1830   {
1831     \dim_set:Nn \l__spath_tmpa_dim {\fp_to_dim:n {#2 - \l__spath_shorten_len_fp } }
1832     \spath_shorten_at_end:NV \l__spath_shorten_path_tl \l__spath_tmpa_dim
1833   }
1834 }
1835
1836 \tl_gset_eq:NN \g__spath_output_tl \l__spath_shorten_path_tl
1837 \group_end:
1838 }
1839 {
1840   \tl_gset:Nn \g__spath_output_tl {#1}
1841 }
1842 }
1843 \cs_new_protected_nopar:Npn \spath_shorten_at_end:Nnn #1#2#3
1844 {
1845   \__spath_shorten_at_end:nn {#2}{#3}
1846   \tl_set_eq:NN #1 \g__spath_output_tl
1847   \tl_gclear:N \g__spath_output_tl

```

```

1848 }
1849 \cs_generate_variant:Nn \spath_shorten_at_end:Nnn {NVV, cnn, cVV, NVn}
1850 \cs_new_protected_nopar:Npn \spath_shorten_at_end:Nn #1#2
1851 {
1852   \spath_shorten_at_end:NVn #1#1{#2}
1853 }
1854 \cs_generate_variant:Nn \spath_shorten_at_end:Nn {cn, cV, NV}
1855 \cs_new_protected_nopar:Npn \spath_gshorten_at_end:Nnn #1#2#3
1856 {
1857   \__spath_shorten_at_end:nn {#2}{#3}
1858   \tl_gset_eq:NN #1 \g__spath_output_tl
1859   \tl_gclear:N \g__spath_output_tl
1860 }
1861 \cs_generate_variant:Nn \spath_gshorten_at_end:Nnn {NVV, cnn, cVV, NVn}
1862 \cs_new_protected_nopar:Npn \spath_gshorten_at_end:Nn #1#2
1863 {
1864   \spath_gshorten_at_end:NVn #1#1{#2}
1865 }
1866 \cs_generate_variant:Nn \spath_gshorten_at_end:Nn {cn, cV, NV}

```

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

\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

```

```

1867 \cs_new_protected_nopar:Npn \__spath_shorten_at_start:nn #1#2
1868 {
1869   \int_compare:nTF
1870   {
1871     \tl_count:n {#1} > 3
1872   }
1873   {
1874     \group_begin:
1875     \tl_set:Nn \l__spath_shorten_path_tl {#1}
1876
1877     \tl_set:Nx \l__spath_shorten_fa_tl {\tl_item:Nn \l__spath_shorten_path_tl {4}}
1878
1879     \tl_clear:N \l__spath_shorten_last_tl
1880
1881     \tl_if_eq:NNTF \l__spath_shorten_fa_tl \c_spath_curvetoa_tl
1882     {
1883       \int_set:Nn \l__spath_shorten_int {3}
1884     }
1885     {
1886       \int_set:Nn \l__spath_shorten_int {1}
1887     }
1888
1889     \tl_set_eq:NN \l__spath_shorten_last_tl \c_spath_movetoa_tl
1890     \tl_set:Nx \l__spath_shorten_path_tl {\tl_tail:N \l__spath_shorten_path_tl }
1891
1892     \prg_replicate:mn { \l__spath_shorten_int }
1893     {
1894       \__spath_tl_put_right_braced:Nx \l__spath_shorten_last_tl {\tl_item:Nn \l__spath_shorten
1895       \__spath_tl_put_right_braced:Nx \l__spath_shorten_last_tl {\tl_item:Nn \l__spath_shorten
1896       \tl_put_right:Nx \l__spath_shorten_last_tl {\tl_item:Nn \l__spath_shorten_path_tl {3}}
1897

```



```

1898 \prg_replicate:nn {3}
1899 {
1900 \tl_set:Nx \l__spath_shorten_path_tl {\tl_tail:N \l__spath_shorten_path_tl }
1901 }
1902 }
1903 \__spath_tl_put_right_braced:Nx \l__spath_shorten_last_tl {\tl_item:Nn \l__spath_shorten_p
1904 \__spath_tl_put_right_braced:Nx \l__spath_shorten_last_tl {\tl_item:Nn \l__spath_shorten_p
1905
1906 \fp_set:Nn \l__spath_shorten_x_fp
1907 {
1908 \dim_to_fp:n {\tl_item:Nn \l__spath_shorten_last_tl {5}}
1909 -
1910 \dim_to_fp:n {\tl_item:Nn \l__spath_shorten_last_tl {2}}
1911 }
1912
1913 \fp_set:Nn \l__spath_shorten_y_fp
1914 {
1915 \dim_to_fp:n {\tl_item:Nn \l__spath_shorten_last_tl {6}}
1916 -
1917 \dim_to_fp:n {\tl_item:Nn \l__spath_shorten_last_tl {3}}
1918 }
1919
1920 \fp_set:Nn \l__spath_shorten_len_fp
1921 {
1922 sqrt( \l__spath_shorten_x_fp * \l__spath_shorten_x_fp + \l__spath_shorten_y_fp * \l__s
1923 }
1924
1925 \fp_compare:nTF
1926 {
1927 \l__spath_shorten_len_fp > #2
1928 }
1929 {
1930
1931 \fp_set:Nn \l__spath_shorten_len_fp
1932 {
1933 #2/ \l__spath_shorten_len_fp
1934 }
1935
1936 \tl_if_eq:NNTF \l__spath_shorten_fa_tl \c_spath_curvetoa_tl
1937 {
1938 \fp_set:Nn \l__spath_shorten_len_fp
1939 {
1940 \l__spath_shorten_len_fp/3
1941 }
1942 \spath_split_curve:NNVV
1943 \l__spath_shorten_lasta_tl
1944 \l__spath_shorten_lastb_tl
1945 \l__spath_shorten_last_tl
1946 \l__spath_shorten_len_fp
1947 }
1948 {
1949 \spath_split_line:NNVV
1950 \l__spath_shorten_lasta_tl
1951 \l__spath_shorten_lastb_tl

```

```

1952     \l__spath_shorten_last_tl
1953     \l__spath_shorten_len_fp
1954   }
1955
1956   \prg_replicate:nn {2}
1957   {
1958     \tl_set:Nx \l__spath_shorten_path_tl {\tl_tail:N \l__spath_shorten_path_tl}
1959   }
1960
1961   \tl_put_left:NV \l__spath_shorten_path_tl \l__spath_shorten_lastb_tl
1962
1963   }
1964   {
1965
1966     \tl_put_left:NV \l__spath_shorten_path_tl \c_spath_moveto_tl
1967
1968     \int_compare:nT
1969     {
1970       \tl_count:N \l__spath_shorten_path_tl > 3
1971     }
1972     {
1973       \dim_set:Nn \l__spath_tmpa_dim {\fp_to_dim:n {#2 - \l__spath_shorten_len_fp } }
1974       \spath_shorten_at_start:NV \l__spath_shorten_path_tl \l__spath_tmpa_dim
1975     }
1976   }
1977
1978   \tl_gset_eq:NN \g__spath_output_tl \l__spath_shorten_path_tl
1979   \group_end:
1980   }
1981   {
1982     \tl_gset:Nn \g__spath_output_tl {#1}
1983   }
1984 }
1985 \cs_new_protected_nopar:Npn \spath_shorten_at_start:Nnn #1#2#3
1986 {
1987   \__spath_shorten_at_start:nn {#2}{#3}
1988   \tl_set_eq:NN #1 \g__spath_output_tl
1989   \tl_gclear:N \g__spath_output_tl
1990 }
1991 \cs_generate_variant:Nn \spath_shorten_at_start:Nnn {NVV, cnn, cVV, NVn}
1992 \cs_new_protected_nopar:Npn \spath_shorten_at_start:Nn #1#2
1993 {
1994   \spath_shorten_at_start:NVn #1#1{#2}
1995 }
1996 \cs_generate_variant:Nn \spath_shorten_at_start:Nn {cn, cV, NV}
1997 \cs_new_protected_nopar:Npn \spath_gshorten_at_start:Nnn #1#2#3
1998 {
1999   \__spath_shorten_at_start:nn {#2}{#3}
2000   \tl_gset_eq:NN #1 \g__spath_output_tl
2001   \tl_gclear:N \g__spath_output_tl
2002 }
2003 \cs_generate_variant:Nn \spath_gshorten_at_start:Nnn {NVV, cnn, cVV, NVn}
2004 \cs_new_protected_nopar:Npn \spath_gshorten_at_start:Nn #1#2
2005 {

```

```

2006 \spath_gshorten_at_start:N\Nn #1#1{#2}
2007 }
2008 \cs_generate_variant:Nn \spath_gshorten_at_start:Nn {cn, cV, NV}

```

(End definition for `\spath_shorten_at_start:Nnn` and others. These functions are documented on page ??.)

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.

```

2009 \cs_new_protected_nopar:Npn \__spath_point_at:nn #1#2
2010 {
2011   \group_begin:
2012   \int_set:Nn \l__spath_tmpa_int {\fp_to_int:n {floor(#2) + 1}}
2013   \fp_set:Nn \l__spath_tmpa_fp {#2 - floor(#2)}
2014
2015   \spath_segments_to_seq:Nn \l__spath_tmpa_seq {#1}
2016
2017   \int_compare:nTF
2018   {
2019     \l__spath_tmpa_int < 1
2020   }
2021   {
2022     \spath_initialpoint:Nn \l__spath_tmpc_tl {#1}
2023   }
2024   {
2025     \int_compare:nTF
2026     {
2027       \l__spath_tmpa_int > \seq_count:N \l__spath_tmpa_seq
2028     }
2029     {
2030       \spath_finalpoint:Nn \l__spath_tmpc_tl {#1}
2031     }
2032     {
2033
2034       \tl_set:Nx \l__spath_tmpa_tl {\seq_item:Nn \l__spath_tmpa_seq { \l__spath_tmpa_int} }
2035
2036       \int_compare:nTF
2037       {
2038         \tl_count:N \l__spath_tmpa_tl > 3
2039       }
2040       {
2041         \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {4}}
2042       }
2043       {
2044         \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {1}}
2045       }
2046
2047       \tl_clear:N \l__spath_tmpc_tl
2048
2049       \tl_case:Nn \l__spath_tmpb_tl
2050       {
2051         \c_spath_moveto_tl

```

```

2052 {
2053   \tl_set:Nx \l__spath_tmpc_tl
2054   {
2055     {
2056       \tl_item:Nn \l__spath_tmpa_tl {2}
2057     }
2058     {
2059       \tl_item:Nn \l__spath_tmpa_tl {3}
2060     }
2061   }
2062 }
2063
2064 \c_spath_lineto_tl
2065 {
2066   \tl_set:Nx \l__spath_tmpc_tl
2067   {
2068     {\fp_to_dim:n
2069       {
2070         (1 - \l__spath_tmpa_fp) * ( \tl_item:Nn \l__spath_tmpa_tl {2} )
2071         +
2072         \l__spath_tmpa_fp * ( \tl_item:Nn \l__spath_tmpa_tl {5} )
2073       }
2074     }
2075     {\fp_to_dim:n
2076       {
2077         (1 - \l__spath_tmpa_fp) * ( \tl_item:Nn \l__spath_tmpa_tl {3} )
2078         +
2079         \l__spath_tmpa_fp * ( \tl_item:Nn \l__spath_tmpa_tl {6} )
2080       }
2081     }
2082   }
2083 }
2084
2085 \c_spath_closepath_tl
2086 {
2087   \tl_set:Nx \l__spath_tmpc_tl
2088   {
2089     {\fp_to_dim:n
2090       {
2091         (1 - \l__spath_tmpa_fp) * ( \tl_item:Nn \l__spath_tmpa_tl {2} )
2092         +
2093         \l__spath_tmpa_fp * ( \tl_item:Nn \l__spath_tmpa_tl {5} )
2094       }
2095     }
2096     {\fp_to_dim:n
2097       {
2098         (1 - \l__spath_tmpa_fp) * ( \tl_item:Nn \l__spath_tmpa_tl {3} )
2099         +
2100         \l__spath_tmpa_fp * ( \tl_item:Nn \l__spath_tmpa_tl {6} )
2101       }
2102     }
2103   }
2104 }
2105

```

```

2106     \c_spath_curvetoa_tl
2107     {
2108         \tl_set:Nx \l__spath_tmpc_tl
2109         {
2110             {\fp_to_dim:n
2111             {
2112                 (1 - \l__spath_tmpa_fp)^3 * \tl_item:Nn \l__spath_tmpa_tl {2} + 3 * (1 - \l__
2113             }}
2114             {\fp_to_dim:n
2115             {
2116                 (1 - \l__spath_tmpa_fp)^3 * \tl_item:Nn \l__spath_tmpa_tl {3} + 3 * (1 - \l__
2117             }}
2118         }
2119     }
2120 }
2121 }
2122 }
2123
2124 \tl_gclear:N \g__spath_output_tl
2125 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpc_tl
2126 \group_end:
2127 }
2128 \cs_new_protected_nopar:Npn \spath_point_at:Nnn #1#2#3
2129 {
2130     \__spath_point_at:nn {#2}{#3}
2131     \tl_set_eq:NN #1 \g__spath_output_tl
2132     \tl_gclear:N \g__spath_output_tl
2133 }
2134 \cs_generate_variant:Nn \spath_point_at:Nnn {NVn, NVV, NnV}
2135 \cs_new_protected_nopar:Npn \spath_gpoint_at:Nnn #1#2#3
2136 {
2137     \__spath_point_at:nn {#2}{#3}
2138     \tl_gset_eq:NN #1 \g__spath_output_tl
2139     \tl_gclear:N \g__spath_output_tl
2140 }
2141 \cs_generate_variant:Nn \spath_gpoint_at:Nnn {NVn, NVV, NnV}

```

(End definition for \spath_point_at:Nnn and \spath_gpoint_at:Nnn. These functions are documented on page ??.)

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

```

2142 \cs_new_protected_nopar:Npn \__spath_tangent_at:nn #1#2
2143 {
2144     \group_begin:
2145     \int_set:Nn \l__spath_tmpa_int {\fp_to_int:n {floor(#2) + 1}}
2146     \fp_set:Nn \l__spath_tmpa_fp {#2 - floor(#2)}
2147
2148     \spath_segments_to_seq:Nn \l__spath_tmpa_seq {#1}
2149
2150     \int_compare:nTF
2151     {
2152         \l__spath_tmpa_int < 1
2153     }

```

```

2154 {
2155   \spath_initialpoint:Nn \l__spath_tmpc_tl {#1}
2156 }
2157 {
2158   \int_compare:nTF
2159   {
2160     \l__spath_tmpa_int > \seq_count:N \l__spath_tmpa_seq
2161   }
2162   {
2163     \spath_finalpoint:Nn \l__spath_tmpc_tl {#1}
2164   }
2165   {
2166
2167     \tl_set:Nx \l__spath_tmpa_tl {\seq_item:Nn \l__spath_tmpa_seq { \l__spath_tmpa_int} }
2168
2169     \int_compare:nTF
2170     {
2171       \tl_count:N \l__spath_tmpa_tl > 3
2172     }
2173     {
2174       \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {4}}
2175     }
2176     {
2177       \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {1}}
2178     }
2179
2180     \tl_clear:N \l__spath_tmpc_tl
2181
2182     \tl_case:Nn \l__spath_tmpb_tl
2183     {
2184       \c_spath_moveto_tl
2185       {
2186         \tl_set:Nx \l__spath_tmpc_tl
2187         {
2188           {
2189             \tl_item:Nn \l__spath_tmpa_tl {2}
2190           }
2191           {
2192             \tl_item:Nn \l__spath_tmpa_tl {3}
2193           }
2194         }
2195       }
2196
2197       \c_spath_lineto_tl
2198       {
2199         \tl_set:Nx \l__spath_tmpc_tl
2200         {
2201           {\fp_to_dim:n
2202           {
2203             ( \tl_item:Nn \l__spath_tmpa_tl {5} )
2204             -
2205             ( \tl_item:Nn \l__spath_tmpa_tl {2} )
2206           }
2207         }

```

```

2208         {\fp_to_dim:n
2209         {
2210             ( \tl_item:Nn \l__spath_tmpa_tl {6} )
2211             -
2212             ( \tl_item:Nn \l__spath_tmpa_tl {3} )
2213         }
2214     }
2215 }
2216 }
2217
2218 \c_spath_closepath_tl
2219 {
2220     \tl_set:Nx \l__spath_tmpc_tl
2221     {
2222         {\fp_to_dim:n
2223         {
2224             ( \tl_item:Nn \l__spath_tmpa_tl {5} )
2225             -
2226             ( \tl_item:Nn \l__spath_tmpa_tl {2} )
2227         }
2228     }
2229     {\fp_to_dim:n
2230     {
2231         ( \tl_item:Nn \l__spath_tmpa_tl {6} )
2232         -
2233         ( \tl_item:Nn \l__spath_tmpa_tl {3} )
2234     }
2235 }
2236 }
2237 }
2238
2239 \c_spath_curvetoa_tl
2240 {
2241     \tl_set:Nx \l__spath_tmpc_tl
2242     {
2243         {\fp_to_dim:n
2244         {
2245             3*(1 - \l__spath_tmpa_fp)^2 * (\tl_item:Nn \l__spath_tmpa_tl {5} - \tl_item:
2246             + 6 * (1 - \l__spath_tmpa_fp) * (\l__spath_tmpa_fp) * (\tl_item:Nn \l__spat
2247             + 3*(\l__spath_tmpa_fp)^2 * (\tl_item:Nn \l__spath_tmpa_tl {11} - \tl_item:N
2248         }
2249     }
2250     {\fp_to_dim:n
2251     {
2252         3*(1 - \l__spath_tmpa_fp)^2 * (\tl_item:Nn \l__spath_tmpa_tl {6} - \tl_item:
2253         + 6 * (1 - \l__spath_tmpa_fp) * (\l__spath_tmpa_fp) * (\tl_item:Nn \l__spat
2254         + 3*(\l__spath_tmpa_fp)^2 * (\tl_item:Nn \l__spath_tmpa_tl {12} - \tl_item:N
2255     }}
2256 }
2257 }
2258 }
2259 }
2260 }
2261

```

```

2262 \tl_gclear:N \g__spath_output_tl
2263 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpc_tl
2264 \group_end:
2265 }
2266 \cs_new_protected_nopar:Npn \spath_tangent_at:Nnn #1#2#3
2267 {
2268   \__spath_tangent_at:nn {#2}{#3}
2269   \tl_set_eq:NN #1 \g__spath_output_tl
2270   \tl_gclear:N \g__spath_output_tl
2271 }
2272 \cs_generate_variant:Nn \spath_tangent_at:Nnn {NVn, NVV, NnV}
2273 \cs_new_protected_nopar:Npn \spath_gtangent_at:Nnn #1#2#3
2274 {
2275   \__spath_tangent_at:nn {#2}{#3}
2276   \tl_gset_eq:NN #1 \g__spath_output_tl
2277   \tl_gclear:N \g__spath_output_tl
2278 }
2279 \cs_generate_variant:Nn \spath_gtangent_at:Nnn {NVn, NVV, NnV}

```

(End definition for `\spath_tangent_at:Nnn` and `\spath_gtangent_at:Nnn`. These functions are documented on page ??.)

`\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`

```

2280 \cs_new_protected_nopar:Npn \__spath_transformation_at:nn #1#2
2281 {
2282   \group_begin:
2283   \tl_clear:N \l__spath_tmpa_tl
2284   \__spath_tangent_at:nn {#1}{#2}
2285   \tl_set_eq:NN \l__spath_tmpb_tl \g__spath_output_tl
2286   \fp_set:Nn \l__spath_tmpa_fp { sqrt( (\tl_item:Nn \l__spath_tmpb_tl {1})^2 + (\tl_item:Nn \l__spath_tmpb_tl {2})^2 ) }
2287   \fp_compare:nTF { \l__spath_tmpa_fp = 0 }
2288   {
2289     \fp_set:Nn \l__spath_tmpa_fp {1}
2290     \fp_set:Nn \l__spath_tmpb_fp {0}
2291   }
2292   {
2293     \fp_set:Nn \l__spath_tmpb_fp { (\tl_item:Nn \l__spath_tmpb_tl {2}) / \l__spath_tmpa_fp }
2294     \fp_set:Nn \l__spath_tmpa_fp { (\tl_item:Nn \l__spath_tmpb_tl {1}) / \l__spath_tmpa_fp }
2295   }
2296   \tl_set:Nx \l__spath_tmpa_tl
2297   {
2298     { \fp_to_decimal:n { \l__spath_tmpa_fp } }
2299     { \fp_to_decimal:n { \l__spath_tmpb_fp } }
2300     { \fp_to_decimal:n {- \l__spath_tmpb_fp } }
2301     { \fp_to_decimal:n { \l__spath_tmpa_fp } }
2302   }
2303   \__spath_point_at:nn {#1}{#2}
2304   \tl_put_right:NV \l__spath_tmpa_tl \g__spath_output_tl
2305   \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
2306   \group_end:
2307 }
2308 \cs_new_protected_nopar:Npn \spath_transformation_at:Nnn #1#2#3
2309 {

```



```

2310 \__spath_transformation_at:nn {#2}{#3}
2311 \tl_set_eq:NN #1 \g__spath_output_tl
2312 \tl_gclear:N \g__spath_output_tl
2313 }
2314 \cs_generate_variant:Nn \spath_transformation_at:Nnn {NVn, NVV, NnV, NvV}
2315 \cs_new_protected_nopar:Npn \spath_gtransformation_at:Nnn #1#2#3
2316 {
2317 \__spath_transformation_at:nn {#2}{#3}
2318 \tl_gset_eq:NN #1 \g__spath_output_tl
2319 \tl_gclear:N \g__spath_output_tl
2320 }
2321 \cs_generate_variant:Nn \spath_gtransformation_at:Nnn {NVn, NVV, NnV}

```

(End definition for `\spath_transformation_at:Nnn` and `\spath_gtransformation_at:Nnn`. These functions are documented on page ??.)

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.

```

2322 \tl_new:N \l__spath_split_path_a_tl
2323 \tl_new:N \l__spath_split_path_b_tl
2324 \tl_new:N \l__spath_split_path_a_start_tl
2325 \tl_new:N \l__spath_split_path_b_start_tl
2326 \tl_new:N \l__spath_split_path_a_end_tl
2327 \tl_new:N \l__spath_split_path_b_end_tl
2328 \tl_new:N \l__spath_split_path_a_final_tl
2329 \tl_new:N \l__spath_split_path_b_final_tl
2330
2331 \tl_new:N \l__spath_split_prev_first_tl
2332 \tl_new:N \l__spath_split_prev_second_tl
2333
2334 \seq_new:N \l__spath_split_first_seq
2335 \seq_new:N \l__spath_split_second_seq
2336
2337 \int_new:N \l__spath_split_segment_int

```

`\spath_intersect:NN` Pass two spaths to pgf's intersection routine.

```

\spath_intersect:nn
2338 \cs_new_protected_nopar:Npn \spath_intersect:NN #1#2
2339 {
2340 \pgfintersectionofpaths%
2341 {%
2342 \pgfsetpath #1
2343 }{%
2344 \pgfsetpath #2
2345 }
2346 }
2347 \tl_new:N \l__spath_intersecta_tl
2348 \tl_new:N \l__spath_intersectb_tl
2349 \cs_new_protected_nopar:Npn \spath_intersect:nn #1#2
2350 {
2351 \tl_set:Nn \l__spath_intersecta_tl {#1}
2352 \tl_set:Nn \l__spath_intersectb_tl {#2}

```

```

2353 \spath_intersect:NN \l__spath_intersecta_tl \l__spath_intersectb_tl
2354 }

```

(End definition for `\spath_intersect:NN` and `\spath_intersect:nn`. These functions are documented on page ??.)

`\spath_split_at_intersections:NNnn` Given two paths, split them at points where they intersect and store them in the macros.

```

\spath_split_at_intersections:NN
\spath_gsplit_at_intersections:NNnn
\spath_gsplit_at_intersections:NN
2355 \cs_new_protected_nopar:Npn \__spath_split_at_intersections:nn #1#2
2356 {
2357   \group_begin:
2358
2359   % Clear some token lists and sequences
2360   \tl_clear:N \l__spath_split_path_a_final_tl
2361   \tl_clear:N \l__spath_split_path_b_final_tl
2362   \seq_clear:N \l__spath_split_first_seq
2363   \seq_clear:N \l__spath_split_second_seq
2364
2365   \pgfintersectionsortbyfirstpath
2366   % Find the intersections of these segments
2367   \tl_set:Nn \l__spath_split_path_a_tl {#1}
2368   \tl_set:Nn \l__spath_split_path_b_tl {#2}
2369   % Remove empty components
2370   % \spath_path_remove_empty_components:N \l__spath_split_path_a_tl
2371   % \spath_path_remove_empty_components:N \l__spath_split_path_b_tl
2372
2373   \spath_intersect:NN \l__spath_split_path_a_tl \l__spath_split_path_b_tl
2374
2375   % If we get intersections
2376   \int_compare:nT {\pgfintersectionsolutions > 0}
2377   {
2378     % Find the times of the intersections on each path
2379     \int_step_inline:nnnn {1} {1} {\pgfintersectionsolutions}
2380     {
2381       \pgfintersectiongetsolutiontimes{##1}{\l__spath_split_first_tl}{\l__spath_split_second_t
2382       \seq_put_left:NV \l__spath_split_first_seq \l__spath_split_first_tl
2383     }
2384   }
2385
2386   \spath_intersect:NN \l__spath_split_path_b_tl \l__spath_split_path_a_tl
2387
2388   % If we get intersections
2389   \int_compare:nT {\pgfintersectionsolutions > 0}
2390   {
2391     % Find the times of the intersections on each path
2392     \int_step_inline:nnnn {1} {1} {\pgfintersectionsolutions}
2393     {
2394       \pgfintersectiongetsolutiontimes{##1}{\l__spath_split_first_tl}{\l__spath_split_second_t
2395       \seq_put_left:NV \l__spath_split_second_seq \l__spath_split_first_tl
2396     }
2397   }
2398
2399   \tl_set:Nn \l__spath_split_prev_first_tl {-1}
2400
2401   \seq_map_inline:Nn \l__spath_split_first_seq

```

```

2402 {
2403   \tl_set:Nn \l__spath_split_first_tl {##1}
2404
2405   \tl_set_eq:NN \l__spath_tmpa_tl \l__spath_split_first_tl
2406   \int_compare:nT
2407   {
2408     \fp_to_int:n {floor( \l__spath_split_first_tl ) }
2409     =
2410     \fp_to_int:n {floor( \l__spath_split_prev_first_tl ) }
2411   }
2412   {
2413     \tl_set:Nx \l__spath_split_first_tl
2414     {
2415       \fp_eval:n {
2416         floor( \l__spath_split_first_tl )
2417         +
2418         ( \l__spath_split_first_tl - floor( \l__spath_split_first_tl ) )
2419         /
2420         ( \l__spath_split_prev_first_tl - floor( \l__spath_split_prev_first_tl ) )
2421       }
2422     }
2423   }
2424   \tl_set_eq:NN \l__spath_split_prev_first_tl \l__spath_tmpa_tl
2425
2426   \spath_split_at:NNVV \l__spath_split_path_a_start_tl \l__spath_split_path_a_end_tl \l__
2427
2428   \tl_put_left:NV \l__spath_split_path_a_final_tl \l__spath_split_path_a_end_tl
2429   \tl_set_eq:NN \l__spath_split_path_a_tl \l__spath_split_path_a_start_tl
2430
2431 }
2432
2433 \tl_set:Nn \l__spath_split_prev_second_tl {-1}
2434
2435 \seq_map_inline:Nn \l__spath_split_second_seq
2436 {
2437   \tl_set:Nn \l__spath_split_second_tl {##1}
2438
2439   \tl_set_eq:NN \l__spath_tmpa_tl \l__spath_split_second_tl
2440   \int_compare:nT
2441   {
2442     \fp_to_int:n {floor( \l__spath_split_second_tl ) }
2443     =
2444     \fp_to_int:n {floor( \l__spath_split_prev_second_tl ) }
2445   }
2446   {
2447     \tl_set:Nx \l__spath_split_second_tl
2448     {
2449       \fp_eval:n {
2450         floor( \l__spath_split_second_tl )
2451         +
2452         ( \l__spath_split_second_tl - floor( \l__spath_split_second_tl ) )
2453         /
2454         ( \l__spath_split_prev_second_tl - floor( \l__spath_split_prev_second_tl ) )
2455       }

```

```

2456     }
2457   }
2458   \tl_set_eq:NN \l__spath_split_prev_second_tl \l__spath_tmpa_tl
2459
2460   \spath_split_at:NNVV \l__spath_split_path_b_start_tl \l__spath_split_path_b_end_tl \l__
2461
2462   \tl_put_left:NV \l__spath_split_path_b_final_tl \l__spath_split_path_b_end_tl
2463   \tl_set_eq:NN \l__spath_split_path_b_tl \l__spath_split_path_b_start_tl
2464
2465 }
2466
2467 \tl_put_left:NV \l__spath_split_path_a_final_tl \l__spath_split_path_a_tl
2468 \tl_put_left:NV \l__spath_split_path_b_final_tl \l__spath_split_path_b_tl
2469
2470 \tl_if_empty:NT \l__spath_split_path_a_final_tl
2471 {
2472   \tl_set_eq:NN \l__spath_split_path_a_final_tl \l__spath_split_path_a_tl
2473 }
2474 \tl_if_empty:NT \l__spath_split_path_b_final_tl
2475 {
2476   \tl_set_eq:NN \l__spath_split_path_b_final_tl \l__spath_split_path_b_tl
2477 }
2478
2479 \spath_remove_empty_components:N \l__spath_split_path_a_final_tl
2480 \spath_remove_empty_components:N \l__spath_split_path_b_final_tl
2481
2482 \tl_gclear:N \g__spath_output_tl
2483 \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_split_path_a_final_tl
2484 \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_split_path_b_final_tl
2485 \group_end:
2486 }
2487 \cs_new_protected_nopar:Npn \spath_split_at_intersections:NNnn #1#2#3#4
2488 {
2489   \__spath_split_at_intersections:nn {#3}{#4}
2490   \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
2491   \tl_set:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
2492   \tl_gclear:N \g__spath_output_tl
2493 }
2494 \cs_generate_variant:Nn \spath_split_at_intersections:NNnn {NNVV, ccVV, ccvv}
2495 \cs_new_protected_nopar:Npn \spath_split_at_intersections:NN #1#2
2496 {
2497   \spath_split_at_intersections:NNVV #1#2#1#2
2498 }
2499 \cs_generate_variant:Nn \spath_split_at_intersections:NN {cc}
2500 \cs_new_protected_nopar:Npn \spath_gsplit_at_intersections:NNnn #1#2#3#4
2501 {
2502   \__spath_split_at_intersections:nn {#3}{#4}
2503   \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
2504   \tl_gset:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
2505   \tl_gclear:N \g__spath_output_tl
2506 }
2507 \cs_generate_variant:Nn \spath_gsplit_at_intersections:NNnn {NNVV, ccVV, ccvv}
2508 \cs_new_protected_nopar:Npn \spath_gsplit_at_intersections:NN #1#2
2509 {

```

```

2510 \spath_gsplit_at_intersections:NNVV #1#2#1#2
2511 }
2512 \cs_generate_variant:Nn \spath_gsplit_at_intersections:NN {cc}

```

(End definition for `\spath_split_at_intersections:NNnn` and others. These functions are documented on page ??.)

```

\spath_split_at_self_intersections:Nn Given a path, split it at points where it self-intersects and store in the given macro.
\spath_split_at_self_intersections:N 2513 \cs_new_protected_nopar:Npn \__spath_split_at_self_intersections:n #1
\spath_gsplit_at_self_intersections:Nn 2514 {
\spath_gsplit_at_self_intersections:N 2515 \group_begin:
2516
2517 % Copy the path
2518 \tl_set:Nn \l__spath_split_path_b_tl {#1}
2519
2520 % Open the path
2521 \spath_open:N \l__spath_split_path_b_tl
2522 % Remove empty components
2523 \spath_remove_empty_components:N \l__spath_split_path_b_tl
2524 % Make a copy for later
2525 \tl_set_eq:NN \l__spath_split_path_b_final_tl \l__spath_split_path_b_tl
2526
2527 % Clear some token lists and sequences
2528 \tl_clear:N \l__spath_split_path_a_tl
2529 \seq_clear:N \l__spath_split_first_seq
2530 \int_zero:N \l__spath_split_segment_int
2531
2532 \pgfintersectionsortbyfirstpath
2533
2534 \bool_do_until:mn
2535 {
2536   \int_compare_p:n
2537   {
2538     \tl_count:N \l__spath_split_path_b_tl < 4
2539   }
2540 }
2541 {
2542   \tl_clear:N \l__spath_split_path_a_tl
2543   \tl_put_right:Nx \l__spath_split_path_a_tl
2544   {
2545     \tl_item:Nn \l__spath_split_path_b_tl {1}
2546     {\tl_item:Nn \l__spath_split_path_b_tl {2}}
2547     {\tl_item:Nn \l__spath_split_path_b_tl {3}}
2548   }
2549
2550   \tl_set:Nx \l__spath_tmpa_tl { \tl_item:Nn \l__spath_split_path_b_tl {4} }
2551
2552   \tl_set:Nx \l__spath_split_path_b_tl {\tl_tail:N \l__spath_split_path_b_tl}
2553
2554   \int_zero:N \l__spath_tmpa_int
2555
2556   \tl_case:Nn \l__spath_tmpa_tl
2557   {
2558     \c_spath_moveto_tl

```

```

2559     {
2560         \tl_clear:N \l__spath_split_path_a_tl
2561         \tl_set:Nx \l__spath_split_path_b_tl {\tl_tail:N \l__spath_split_path_b_tl}
2562         \tl_set:Nx \l__spath_split_path_b_tl {\tl_tail:N \l__spath_split_path_b_tl}
2563         \tl_set:Nx \l__spath_split_path_b_tl {\tl_tail:N \l__spath_split_path_b_tl}
2564     }
2565     \c_spath_lineto_tl
2566     {
2567         \int_set:Nn \l__spath_tmpa_int {1}
2568     }
2569     \c_spath_curvetoa_tl
2570     {
2571         \int_set:Nn \l__spath_tmpa_int {3}
2572     }
2573 }
2574
2575 \prg_replicate:nn { \l__spath_tmpa_int }
2576 {
2577     \tl_set:Nx \l__spath_split_path_b_tl {\tl_tail:N \l__spath_split_path_b_tl}
2578     \tl_set:Nx \l__spath_split_path_b_tl {\tl_tail:N \l__spath_split_path_b_tl}
2579
2580     \tl_put_right:Nx \l__spath_split_path_a_tl
2581     {
2582         \tl_item:Nn \l__spath_split_path_b_tl {1}
2583         {\tl_item:Nn \l__spath_split_path_b_tl {2}}
2584         {\tl_item:Nn \l__spath_split_path_b_tl {3}}
2585     }
2586
2587     \tl_set:Nx \l__spath_split_path_b_tl {\tl_tail:N \l__spath_split_path_b_tl}
2588 }
2589
2590 \tl_put_left:NV \l__spath_split_path_b_tl \c_spath_moveto_tl
2591
2592 \tl_if_empty:NF \l__spath_split_path_a_tl
2593 {
2594     % Intersect the current segment with the rest of the path
2595     \spath_intersect:NN \l__spath_split_path_a_tl \l__spath_split_path_b_tl
2596
2597
2598     % If we get intersections
2599     \int_compare:nT {\pgfintersectionsolutions > 0}
2600     {
2601         % Find the times of the intersections on each path
2602         \int_step_inline:nnnn {1} {1} {\pgfintersectionsolutions}
2603         {
2604             \pgfintersectiongetsolutiontimes{##1}{\l__spath_split_first_tl}{\l__spath_split_se
2605             \fp_compare:nT
2606             {
2607                 \l__spath_split_first_tl < .99
2608             }
2609             {
2610                 \tl_set:Nx \l__spath_tmpa_tl {\fp_to_decimal:n {\l__spath_split_first_tl + \l__
2611                 \seq_put_right:NV \l__spath_split_first_seq \l__spath_tmpa_tl
2612             }

```

```

2613     }
2614   }
2615
2616   \spath_intersect:NN \l__spath_split_path_b_tl \l__spath_split_path_a_tl
2617
2618   % If we get intersections
2619   \int_compare:nT {\pgfintersectionsolutions > 0}
2620   {
2621     % Find the times of the intersections on each path
2622     \int_step_inline:nmmn {1} {1} {\pgfintersectionsolutions}
2623     {
2624       \pgfintersectiongetsolutiontimes{##1}{\l__spath_split_first_tl}{\l__spath_split_se
2625       \fp_compare:nT
2626       {
2627         \l__spath_split_first_tl > .01
2628       }
2629       {
2630         \tl_set:Nx \l__spath_tmpa_tl {\fp_to_decimal:n {\l__spath_split_first_tl + \l__
2631         \seq_put_right:NV \l__spath_split_first_seq \l__spath_tmpa_tl
2632       }
2633     }
2634   }
2635
2636   }
2637   % Increment the segment counter
2638   \int_incr:N \l__spath_split_segment_int
2639 }
2640
2641 % Sort the sequence by reverse order along the path
2642 \seq_sort:Nn \l__spath_split_first_seq
2643 {
2644   \fp_compare:nNnTF { ##1 } < { ##2 }
2645   { \sort_return_swapped: }
2646   { \sort_return_same: }
2647 }
2648
2649 % Restore the original copy of the path
2650 \tl_set_eq:NN \l__spath_split_path_b_tl \l__spath_split_path_b_final_tl
2651
2652 % Clear the token lists
2653 \tl_clear:N \l__spath_split_path_b_start_tl
2654 \tl_clear:N \l__spath_split_path_b_end_tl
2655 \tl_clear:N \l__spath_split_path_b_final_tl
2656
2657 \tl_set:Nn \l__spath_split_prev_first_tl {-1}
2658
2659 \seq_map_inline:Nn \l__spath_split_first_seq
2660 {
2661   \tl_set:Nn \l__spath_split_first_tl {##1}
2662   \tl_set_eq:NN \l__spath_tmpa_tl \l__spath_split_first_tl
2663   \int_compare:nT
2664   {
2665     \fp_to_int:n {floor( \l__spath_split_first_tl ) }
2666     =

```

```

2667     \fp_to_int:n {floor( \l__spath_split_prev_first_tl ) }
2668   }
2669   {
2670     \tl_set:Nx \l__spath_split_first_tl
2671     {
2672       \fp_eval:n {
2673         floor( \l__spath_split_first_tl )
2674         +
2675         ( \l__spath_split_first_tl - floor( \l__spath_split_first_tl ) )
2676         /
2677         ( \l__spath_split_prev_first_tl - floor( \l__spath_split_prev_first_tl ) )
2678       }
2679     }
2680   }
2681   \tl_set_eq:NN \l__spath_split_prev_first_tl \l__spath_tmpa_tl
2682
2683   \spath_split_at:NNVV \l__spath_split_path_b_start_tl \l__spath_split_path_b_end_tl \l__
2684
2685   \tl_put_left:NV \l__spath_split_path_b_final_tl \l__spath_split_path_b_end_tl
2686   \tl_set_eq:NN \l__spath_split_path_b_tl \l__spath_split_path_b_start_tl
2687
2688 }
2689
2690 \tl_put_left:NV \l__spath_split_path_b_final_tl \l__spath_split_path_b_tl
2691
2692 \tl_if_empty:NT \l__spath_split_path_b_final_tl
2693 {
2694   \tl_set_eq:NN \l__spath_split_path_b_final_tl \l__spath_split_path_b_tl
2695 }
2696
2697 \spath_remove_empty_components:N \l__spath_split_path_b_final_tl
2698
2699 \tl_gclear:N \g__spath_output_tl
2700 \tl_gset_eq:NN \g__spath_output_tl \l__spath_split_path_b_final_tl
2701 \group_end:
2702 }
2703 \cs_new_protected_nopar:Npn \spath_split_at_self_intersections:Nn #1#2
2704 {
2705   \__spath_split_at_self_intersections:n {#2}
2706   \tl_set_eq:NN #1 \g__spath_output_tl
2707   \tl_gclear:N \g__spath_output_tl
2708 }
2709 \cs_generate_variant:Nn \spath_split_at_self_intersections:Nn {NV}
2710 \cs_new_protected_nopar:Npn \spath_split_at_self_intersections:N #1
2711 {
2712   \spath_split_at_self_intersections:NV #1#1
2713 }
2714 \cs_generate_variant:Nn \spath_split_at_self_intersections:N {c}
2715 \cs_new_protected_nopar:Npn \spath_gsplit_at_self_intersections:Nn #1#2
2716 {
2717   \__spath_split_at_self_intersections:n {#2}
2718   \tl_gset_eq:NN #1 \g__spath_output_tl
2719   \tl_gclear:N \g__spath_output_tl
2720 }

```



```

2721 \cs_generate_variant:Nn \spath_gsplit_at_self_intersections:Nn {NV}
2722 \cs_new_protected_nopar:Npn \spath_gsplit_at_self_intersections:N #1
2723 {
2724   \spath_gsplit_at_self_intersections:NV #1#1
2725 }
2726 \cs_generate_variant:Nn \spath_gsplit_at_self_intersections:N {c}

```

(End definition for `\spath_split_at_self_intersections:Nn` and others. These functions are documented on page ??.)

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

```

\spath_join_component:Nn 2727 \tl_new:N \l__spath_tmpj_tl
\spath_gjoin_component:Nnn 2728 \cs_new_protected_nopar:Npn \__spath_join_component:nn #1#2
\spath_gjoin_component:Nn 2729 {
2730   \group_begin:
2731   \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
2732   \int_compare:nT
2733   {
2734     #2 == 1
2735   }
2736   {
2737     \tl_clear:N \l__spath_tmpj_tl
2738     \seq_pop_left:NN \l__spath_tmpa_seq \l__spath_tmpj_tl
2739
2740     \prg_replicate:nn {3}
2741     {
2742       \tl_set:Nx \l__spath_tmpj_tl {\tl_tail:N \l__spath_tmpj_tl}
2743     }
2744
2745     \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpj_tl
2746   }
2747   \bool_if:nT
2748   {
2749     \int_compare_p:n
2750     {
2751       #2 > 1
2752     }
2753     &&
2754     \int_compare_p:n
2755     {
2756       #2 <= \seq_count:N \l__spath_tmpa_seq
2757     }
2758   }
2759   {
2760
2761     \seq_clear:N \l__spath_tmpb_seq
2762     \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
2763     {
2764       \tl_set:Nn \l__spath_tmpj_tl {##2}
2765       \int_compare:nT {##1 = #2}
2766       {
2767         \prg_replicate:nn {3}
2768         {
2769           \tl_set:Nx \l__spath_tmpj_tl {\tl_tail:N \l__spath_tmpj_tl}

```

```

2770     }
2771   }
2772   \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpj_tl
2773 }
2774
2775 \seq_set_eq:NN \l__spath_tmpa_seq \l__spath_tmpb_seq
2776 }
2777
2778 \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpa_seq {}}
2779 \group_end:
2780 }
2781 \cs_new_protected_nopar:Npn \spath_join_component:Nnn #1#2#3
2782 {
2783   \__spath_join_component:nn {#2}{#3}
2784   \tl_set_eq:NN #1 \g__spath_output_tl
2785   \tl_gclear:N \g__spath_output_tl
2786 }
2787 \cs_generate_variant:Nn \spath_join_component:Nnn {NVn, NVV}
2788 \cs_new_protected_nopar:Npn \spath_join_component:Nn #1#2
2789 {
2790   \spath_join_component:NVn #1#1{#2}
2791 }
2792 \cs_generate_variant:Nn \spath_join_component:Nn {cn, NV, cV}
2793 \cs_new_protected_nopar:Npn \spath_gjoin_component:Nnn #1#2#3
2794 {
2795   \__spath_join_component:nn {#2}{#3}
2796   \tl_gset_eq:NN #1 \g__spath_output_tl
2797   \tl_gclear:N \g__spath_output_tl
2798 }
2799 \cs_generate_variant:Nn \spath_gjoin_component:Nnn {NVn, NVV}
2800 \cs_new_protected_nopar:Npn \spath_gjoin_component:Nn #1#2
2801 {
2802   \spath_gjoin_component:NVn #1#1{#2}
2803 }
2804 \cs_generate_variant:Nn \spath_gjoin_component:Nn {cn, NV, cV}

```

(End definition for `\spath_join_component:Nnn` and others. These functions are documented on page ??.)

`\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
2805 \cs_new_protected_nopar:Npn \__spath_spot_weld_components:n #1
2806 {
2807   \group_begin:
2808   \dim_zero:N \l__spath_move_x_dim
2809   \dim_zero:N \l__spath_move_y_dim
2810
2811   \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
2812   \seq_clear:N \l__spath_tmpb_seq
2813   \dim_set:Nn \l__spath_move_x_dim {\tl_item:nn {#1} {2} + 10 pt}
2814   \dim_set:Nn \l__spath_move_y_dim {\tl_item:nn {#1} {3} + 10 pt}
2815
2816   \int_set:Nn \l__spath_tmpa_int {\seq_count:N \l__spath_tmpa_seq}
2817

```

```

2818 \seq_map_inline:Nn \l__spath_tmpa_seq
2819 {
2820   \tl_set:Nn \l__spath_tmpa_tl {##1}
2821   \bool_if:nT
2822   {
2823     \dim_compare_p:n
2824     {
2825       \dim_abs:n {\l__spath_move_x_dim - \tl_item:Nn \l__spath_tmpa_tl {2} } < 0.01pt
2826     }
2827     &&
2828     \dim_compare_p:n
2829     {
2830       \dim_abs:n {\l__spath_move_y_dim - \tl_item:Nn \l__spath_tmpa_tl {3} } < 0.01pt
2831     }
2832   }
2833   {
2834     \prg_replicate:nn {3}
2835     {
2836       \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
2837     }
2838     \int_decr:N \l__spath_tmpa_int
2839   }
2840   \tl_reverse:N \l__spath_tmpa_tl
2841   \dim_set:Nn \l__spath_move_x_dim {\tl_item:Nn \l__spath_tmpa_tl {2}}
2842   \dim_set:Nn \l__spath_move_y_dim {\tl_item:Nn \l__spath_tmpa_tl {1}}
2843   \tl_reverse:N \l__spath_tmpa_tl
2844   \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
2845 }
2846
2847 \tl_set:Nx \l__spath_tmpa_tl {\seq_use:Nn \l__spath_tmpb_seq {} }
2848 \spath_components_to_seq:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
2849
2850
2851 \spath_initialpoint:Nn \l__spath_tmpa_tl {##1}
2852 \spath_finalpoint:Nn \l__spath_tmpb_tl {##1}
2853
2854 \bool_if:nT
2855 {
2856   \dim_compare_p:n
2857   {
2858     \dim_abs:n {\tl_item:Nn \l__spath_tmpa_tl {1} - \tl_item:Nn \l__spath_tmpb_tl {1} } <
2859   }
2860   &&
2861   \dim_compare_p:n
2862   {
2863     \dim_abs:n {\tl_item:Nn \l__spath_tmpa_tl {2} - \tl_item:Nn \l__spath_tmpb_tl {2} } <
2864   }
2865 }
2866 {
2867   \int_compare:nTF
2868   {
2869     \seq_count:N \l__spath_tmpb_seq > 1
2870   }
2871   {

```

```

2872     \seq_pop_left:NN \l__spath_tmpb_seq \l__spath_tmpb_tl
2873
2874     \prg_replicate:nn {3}
2875     {
2876       \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
2877     }
2878     \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpb_tl
2879   }
2880   {
2881     \tl_set:NV \l__spath_tmpb_tl \c_spath_closepath_tl
2882     \tl_put_right:Nx \l__spath_tmpb_tl
2883     {
2884       { \tl_item:Nn \l__spath_tmpa_tl {1} }
2885       { \tl_item:Nn \l__spath_tmpa_tl {2} }
2886     }
2887     \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpb_tl
2888   }
2889 }
2890
2891 \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpb_seq {}}
2892 \group_end:
2893 }
2894 \cs_new_protected_nopar:Npn \spath_spot_weld_components:Nn #1#2
2895 {
2896   \__spath_spot_weld_components:n {#2}
2897   \tl_set_eq:NN #1 \g__spath_output_tl
2898   \tl_gclear:N \g__spath_output_tl
2899 }
2900 \cs_generate_variant:Nn \spath_spot_weld_components:Nn {NV, cV, cn}
2901 \cs_new_protected_nopar:Npn \spath_spot_weld_components:N #1
2902 {
2903   \spath_spot_weld_components:NV #1#1
2904 }
2905 \cs_generate_variant:Nn \spath_spot_weld_components:N {c}
2906 \cs_new_protected_nopar:Npn \spath_spot_gweld_components:Nn #1#2
2907 {
2908   \__spath_spot_weld_components:n {#2}
2909   \tl_gset_eq:NN #1 \g__spath_output_tl
2910   \tl_gclear:N \g__spath_output_tl
2911 }
2912 \cs_generate_variant:Nn \spath_spot_gweld_components:Nn {NV, cV, cn}
2913 \cs_new_protected_nopar:Npn \spath_spot_gweld_components:N #1
2914 {
2915   \spath_spot_gweld_components:NV #1#1
2916 }
2917 \cs_generate_variant:Nn \spath_spot_gweld_components:N {c}

```

(End definition for \spath_spot_weld_components:Nn and others. These functions are documented on page ??.)

3.8 Exporting Commands

```

\spath_convert_to_svg:Nn Convert the soft path to an SVG document.
\spath_gconvert_to_svg:Nn 2918 \cs_new_protected_nopar:Npn \__spath_convert_to_svg:n #1

```

```

2919 {
2920   \group_begin:
2921   \tl_clear:N \l__spath_tmpa_tl
2922   \tl_put_right:Nn \l__spath_tmpa_tl
2923   {
2924     <?xml~ version="1.0"~ standalone="no"?>
2925     \iow_newline:
2926     <!DOCTYPE~ svg~ PUBLIC~ "-//W3C//DTD SVG 1.1//EN"~
2927     "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
2928     \iow_newline:
2929     <svg~ xmlns="http://www.w3.org/2000/svg"~ version="1.1"~viewBox="
2930   }
2931
2932   \spath_minbb:Nn \l__spath_tmpb_tl {#1}
2933   \spath_maxbb:Nn \l__spath_tmpc_tl {#1}
2934   \tl_put_right:Nx \l__spath_tmpa_tl
2935   {
2936     \dim_to_decimal:n
2937     {
2938       \tl_item:Nn \l__spath_tmpb_tl {1} - 10pt
2939     }
2940     \exp_not:n {~}
2941     \dim_to_decimal:n
2942     {
2943       \tl_item:Nn \l__spath_tmpb_tl {2} - 10pt
2944     }
2945     \exp_not:n {~}
2946     \dim_to_decimal:n
2947     {
2948       \tl_item:Nn \l__spath_tmpc_tl {1}
2949       -
2950       \tl_item:Nn \l__spath_tmpb_tl {1}
2951       + 20pt
2952     }
2953     \exp_not:n {~}
2954     \dim_to_decimal:n
2955     {
2956       \tl_item:Nn \l__spath_tmpc_tl {2}
2957       -
2958       \tl_item:Nn \l__spath_tmpb_tl {2}
2959       + 20pt
2960     }
2961   }
2962
2963   \tl_put_right:Nn \l__spath_tmpa_tl
2964   {
2965     ">
2966     \iow_newline:
2967     <path~ d="
2968   }
2969   \tl_set:Nn \l__spath_tmpc_tl {use:n}
2970   \tl_map_inline:Nn #1
2971   {
2972     \tl_set:Nn \l__spath_tmpb_tl {##1}

```

```

2973 \tl_case:NnF \l__spath_tmpb_tl
2974 {
2975   \c_spath_moveto_tl
2976   {
2977     \tl_put_right:Nn \l__spath_tmpa_tl {M~}
2978     \tl_set:Nn \l__spath_tmpc_tl {use:n}
2979   }
2980   \c_spath_lineto_tl
2981   {
2982     \tl_put_right:Nn \l__spath_tmpa_tl {L~}
2983     \tl_set:Nn \l__spath_tmpc_tl {use:n}
2984   }
2985   \c_spath_closepath_tl
2986   {
2987     \tl_put_right:Nn \l__spath_tmpa_tl {Z~}
2988     \tl_set:Nn \l__spath_tmpc_tl {use_none:n}
2989   }
2990   \c_spath_curvetoa_tl
2991   {
2992     \tl_put_right:Nn \l__spath_tmpa_tl {C~}
2993     \tl_set:Nn \l__spath_tmpc_tl {use:n}
2994   }
2995   \c_spath_curvetob_tl {
2996     \tl_set:Nn \l__spath_tmpc_tl {use:n}
2997   }
2998   \c_spath_curveto_tl {
2999     \tl_set:Nn \l__spath_tmpc_tl {use:n}
3000   }
3001 }
3002 {
3003   \tl_put_right:Nx \l__spath_tmpa_tl {\use:c { \l__spath_tmpc_tl } {\dim_to_decimal:n {#
3004 }
3005 }
3006 \tl_put_right:Nn \l__spath_tmpa_tl
3007 {
3008   "~ fill="none"~ stroke="black"~ />
3009   \iow_newline:
3010   </svg>
3011   \iow_newline:
3012 }
3013 \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
3014 \group_end:
3015 }
3016 \cs_new_protected_nopar:Npn \spath_convert_to_svg:Nn #1#2
3017 {
3018   \__spath_convert_to_svg:n {#2}
3019   \tl_set_eq:NN #1 \g__spath_output_tl
3020   \tl_gclear:N \g__spath_output_tl
3021 }
3022 \cs_new_protected_nopar:Npn \spath_gconvert_to_svg:Nn #1#2
3023 {
3024   \__spath_convert_to_svg:n {#2}
3025   \tl_gset_eq:NN #1 \g__spath_output_tl
3026   \tl_gclear:N \g__spath_output_tl

```

```
3027 }
```

(End definition for `\spath_convert_to_svg:Nn` and `\spath_gconvert_to_svg:Nn`. These functions are documented on page ??.)

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

```
3028 \iow_new:N \g__spath_stream
3029 \cs_new_protected_nopar:Npn \spath_export_to_svg:nn #1#2
3030 {
3031   \spath_convert_path_to_svg:Nn \l__spath_iterp_tl {#2}
3032   \iow_open:Nn \g__spath_stream {#1 .svg}
3033   \iow_now:Nx \g__spath_stream
3034   {
3035     \tl_use:N \l__spath_iterp_tl
3036   }
3037   \iow_close:N \g__spath_stream
3038 }
```

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

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

```
3039 \cs_new_protected_nopar:Npn \spath_show:n #1
3040 {
3041   \int_step_inline:nnnn {1} {3} {\tl_count:n {#1}}
3042   {
3043     \iow_term:x {
3044       \tl_item:Nn \l__spath_tmpa_tl {##1}
3045       {\tl_item:Nn \l__spath_tmpa_tl {##1+1}}
3046       {\tl_item:Nn \l__spath_tmpa_tl {##1+2}}
3047     }
3048   }
3049 }
3050 \cs_generate_variant:Nn \spath_show:n {V, v}
```

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

3.8.1 PGF and TikZ Interface Functions

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

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

`\spath_gget_current_path:N`

```
3051 \cs_new_protected_nopar:Npn \spath_get_current_path:N #1
3052 {
3053   \pgfsyssoftpath@getcurrentpath #1
3054 }
3055 \cs_new_protected_nopar:Npn \spath_gget_current_path:N #1
3056 {
3057   \pgfsyssoftpath@getcurrentpath #1
3058   \tl_gset_eq:NN #1 #1
3059 }
```

(End definition for `\spath_get_current_path:N` and `\spath_gget_current_path:N`. These functions are documented on page ??.)

`\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.

```

3060 \cs_new_protected_nopar:Npn \spath_protocol_path:n #1
3061 {
3062   \spath_minbb:Nn \l__spath_tmpa_tl {#1}
3063   \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl
3064
3065   \spath_maxbb:Nn \l__spath_tmpa_tl {#1}
3066   \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl
3067 }
3068 \cs_generate_variant:Nn \spath_protocol_path:n {V}

```

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

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

```

\spath_set_current_path:N
3069 \cs_new_protected_nopar:Npn \spath_set_current_path:n #1
3070 {
3071   \spath_protocol_path:n {#1}
3072   \tl_set:Nn \l__spath_tmpa_tl {#1}
3073   \pgfsyssoftpath@setcurrentpath\l__spath_tmpa_tl
3074 }
3075 \cs_new_protected_nopar:Npn \spath_set_current_path:N #1
3076 {
3077   \spath_protocol_path:V #1
3078   \pgfsyssoftpath@setcurrentpath #1
3079 }
3080 \cs_generate_variant:Nn \spath_set_current_path:N {c}

```

(End definition for `\spath_set_current_path:n` and `\spath_set_current_path:N`. These functions are documented on page ??.)

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

```

3081 \cs_new_protected_nopar:Npn \spath_use_path:nn #1#2
3082 {
3083   \spath_set_current_path:n {#1}
3084   \pgfusepath{#2}
3085 }

```

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

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

```

3086 \cs_new_protected_nopar:Npn \spath_tikz_path:nn #1#2
3087 {
3088   \path[#1] \pgfextra{
3089     \spath_set_current_path:n {#2}
3090     \tl_put_right:Nn \tikz@preactions {\def\tikz@actions@path{#2}}
3091   };
3092 }
3093 \cs_generate_variant:Nn \spath_tikz_path:nn {Vn, VV, nv, Vv, nV}

```

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

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

```
3094 \cs_new_protected_nopar:Npn \spath_set_tikz_coords:n #1
3095 {
3096   \spath_finalpoint:Nn \l__spath_tmpa_tl {#1}
3097   \tl_set:Nx \l__spath_tmpa_tl
3098   {
3099     \exp_not:c {tikz@lastx}=\tl_item:Nn \l__spath_tmpa_tl {1}
3100     \exp_not:c {tikz@lasty}=\tl_item:Nn \l__spath_tmpa_tl {2}
3101     \exp_not:c {tikz@lastxsaved}=\tl_item:Nn \l__spath_tmpa_tl {1}
3102     \exp_not:c {tikz@lastysaved}=\tl_item:Nn \l__spath_tmpa_tl {2}
3103   }
3104   \tl_use:N \l__spath_tmpa_tl
3105 }
3106 \cs_generate_variant:Nn \spath_set_tikz_coords:n {V, v}
```

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

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.

```
3107 \RequirePackage{spath3}
3108 \RequirePackage{expl3}
3109 \ExplSyntaxOn
3110
3111 \tl_new:N \l__spath_current_tl
3112 \tl_new:N \l__spath_reverse_tl
3113 \tl_new:N \l__spath_prefix_tl
3114 \tl_new:N \l__spath_suffix_tl
3115 \tl_new:N \g__spath_smuggle_tl
3116 \seq_new:N \g__spath_tmpa_seq
3117 \seq_new:N \g__spath_tmpb_seq
3118 \bool_new:N \l__spath_draft_bool
```

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.

```
3119 \tl_set:Nn \l__spath_prefix_tl {tikz@intersect@path@name@}
3120 \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.

```
3121 \tl_new:N \g__spath_tikzfinish_tl
3122 \cs_new_protected_nopar:Npn \spath_at_end_of_path:
3123 {
3124   \tl_use:N \g__spath_tikzfinish_tl
3125   \tl_gclear:N \g__spath_tikzfinish_tl
3126 }
3127 \tl_put_right:Nn \tikz@finish {\spath_at_end_of_path:}
3128
3129 \cs_new_protected_nopar:Npn \spath_save_path:Nn #1#2
```

```

3130 {
3131   \tl_gput_right:Nn \g__spath_tikzfinish_tl
3132   {
3133     \tl_clear_new:N #1
3134     \tl_set:Nn #1 {#2}
3135   }
3136 }
3137 \cs_generate_variant:Nn \spath_save_path:Nn {cn, NV, cV}
3138
3139 \cs_new_protected_nopar:Npn \spath_gsave_path:Nn #1#2
3140 {
3141   \tl_gput_right:Nn \g__spath_tikzfinish_tl
3142   {
3143     \tl_gclear_new:N #1
3144     \tl_gset:Nn #1 {#2}
3145   }
3146 }
3147 \cs_generate_variant:Nn \spath_gsave_path:Nn {cn, NV, cV}

```

Now we define all of our keys.

```

3148 \tikzset{

```

We're in the `spath` key family.

```

3149   spath/.is~family,
3150   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.

```

3151   set~ prefix/.store~ in=\l__spath_prefix_tl,
3152   prefix/.is~choice,
3153   prefix/default/.style={
3154     /tikz/spath/set~ prefix=tikz@intersect@path@name@
3155   },
3156   set~ suffix/.store~ in=\l__spath_suffix_tl,
3157   suffix/.is~choice,
3158   suffix/default/.style={
3159     /tikz/spath/set~ suffix={}
3160   },
3161   set~ name/.style={
3162     /tikz/spath/prefix=#1,
3163     /tikz/spath/suffix=#1
3164   },

```

Keys for saving and cloning a soft path.

```

3165   save/.code={
3166     \tikz@addmode{
3167       \spath_get_current_path:N \l__spath_tmpa_tl
3168       \spath_bake_round:N \l__spath_tmpa_tl \l__spath_tmpa_tl
3169       \spath_save_path:cV {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} \
3170     }
3171   },
3172   clone/.code~ 2~ args={
3173     \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl} \
3174     {

```

```

3175     \tl_clear_new:c
3176     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3177     \tl_set_eq:cc
3178     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3179     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
3180   }
3181 },
3182 clone~ global/.code~ 2~ args={
3183   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
3184   {
3185     \tl_gclear_new:c
3186     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3187     \tl_gset_eq:cc
3188     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3189     {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
3190   }
3191 },
3192 save~ global/.code={
3193   \tikz@addmode{
3194     \spath_get_current_path:N \l__spath_tmpa_tl
3195     \spath_bake_round:NV \l__spath_tmpa_tl \l__spath_tmpa_tl
3196     \spath_gsave_path:cV
3197     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3198     \l__spath_tmpa_tl
3199   }
3200 },

```

Saves a soft path to the aux file.

```

3201 save~ to~ aux/.code={
3202   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3203   {
3204     \spath_save_to_aux:c
3205     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3206   }
3207 },

```

Restores a soft path to the current path.

```

3208 restore/.code={
3209   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3210   {
3211     \spath_set_current_path:c
3212     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3213     \spath_set_tikz_coords:v
3214     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3215   }
3216 },

```

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

```

3217 show~current~path/.code={
3218   \tikz@addmode{
3219     \pgfsyssoftpath@getcurrentpath\l__spath_tmpa_tl
3220     \iow_term:n {---~ current~ soft~ path~ ---}
3221     \spath_show:V \l__spath_tmpa_tl
3222   }
3223 },

```

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

```
3224 show/.code={
3225   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3226   {
3227     \iow_term:n {---- soft~ path~ #1~ ---}
3228     \spath_show:v {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3229   }
3230 },
```

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

```
3231 append/.code={
3232   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3233   {
3234     \spath_get_current_path:N \l__spath_current_tl
3235     \spath_weld:Nv
3236     \l__spath_current_tl
3237     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3238     \spath_set_current_path:N \l__spath_current_tl
3239     \spath_set_tikz_coords:V \l__spath_current_tl
3240   }
3241 },
```

Joins the second named path to the first.

```
3242 join~ with/.code~ 2~ args={
3243   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3244   {
3245     \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
3246     {
3247       \spath_append:cv
3248       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3249       {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
3250     }
3251   }
3252 },
```

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

```
3253 spot~ weld/.code={
3254   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3255   {
3256     \spath_spot_weld_components:c
3257     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3258   }
3259 },
```

Reverses the named path.

```
3260 reverse/.code={
3261   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3262   {
3263     \spath_reverse:c
3264     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3265   }
3266 },
```

Appends the reversal of the path to the current path.

```

3267 append~ reverse/.code={
3268   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3269   {
3270     \spath_reverse:Nv
3271     \l__spath_reverse_tl
3272     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3273     \spath_get_current_path:N \l__spath_current_tl
3274     \spath_weld:Nv \l__spath_current_tl \l__spath_reverse_tl
3275     \spath_set_current_path:N \l__spath_current_tl
3276     \spath_set_tikz_coords:V \l__spath_current_tl
3277   }
3278 },

```

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

```

3279 insert/.code={
3280   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3281   {
3282     \spath_get_current_path:N \l__spath_current_tl
3283     \spath_append:Nv
3284     \l__spath_current_tl
3285     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3286     \spath_set_current_path:N \l__spath_current_tl
3287     \spath_set_tikz_coords:V \l__spath_current_tl
3288   }
3289 },

```

Inserts the reverse of the named path.

```

3290 insert~ reverse/.code={
3291   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3292   {
3293     \spath_reverse:Nv
3294     \l__spath_reverse_tl
3295     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3296     \spath_get_current_path:N \l__spath_current_tl
3297     \spath_append:Nv \l__spath_current_tl \l__spath_reverse_tl
3298     \spath_set_current_path:N \l__spath_current_tl
3299     \spath_set_tikz_coords:V \l__spath_current_tl
3300   }
3301 },

```

These keys shorten the path.

```

3302 shorten~ at~ end/.code~ 2~ args={
3303   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3304   {
3305     \spath_shorten_at_end:cn
3306     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
3307   }
3308 },
3309 shorten~ at~ start/.code~ 2~ args ={
3310   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3311   {
3312     \spath_shorten_at_start:cn

```

```

3313     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
3314   }
3315 },
3316 shorten~ at~ both~ ends/.code~ 2~ args={
3317   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3318   {
3319     \spath_shorten_at_end:cn
3320     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
3321     \spath_shorten_at_start:cn
3322     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl} {#2}
3323   }
3324 },

```

This translates the named path.

```

3325 translate/.code~ n~ args={3}{
3326   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3327   {
3328     \spath_translate:cn
3329     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}{#2}{#3}
3330   }
3331 },

```

Exports the path as an SVG file.

```

3332 export~ to~ svg/.code={
3333   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3334   {
3335     \spath_export_to_svg:c
3336     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3337   }
3338 },

```

Transforms the named path using TikZ transformation specifications.

```

3339 transform/.code~ 2~ args={
3340   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3341   {
3342     \group_begin:
3343     \pgftransformreset
3344     \tikzset{#2}
3345     \pgfgettransform \l__spath_tmpa_tl
3346     \tl_gset:Nn \g__spath_smuggle_tl
3347     {
3348       \spath_transform:cnnnnn
3349       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3350     }
3351     \tl_gput_right:NV \g__spath_smuggle_tl \l__spath_tmpa_tl
3352     \group_end:
3353     \tl_use:N \g__spath_smuggle_tl
3354   }
3355 },
3356 transform~global/.code~ 2~ args={
3357   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3358   {
3359     \group_begin:
3360     \pgftransformreset

```

```

3361 \tikzset{#2}
3362 \pgfgettransform \l__spath_tmpa_tl
3363 \tl_gset:Nn \g__spath_smuggle_tl
3364 {
3365   \spath_gtransform:cnnnnn
3366   {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3367 }
3368 \tl_gput_right:NV \g__spath_smuggle_tl \l__spath_tmpa_tl
3369 \group_end:
3370 \tl_use:N \g__spath_smuggle_tl
3371 }
3372 },

```

Splits two paths at their mutual intersections.

```

3373 split~ at~ intersections/.code~ n~ args={2}{
3374   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3375   {
3376     \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
3377     {
3378       \spath_split_at_intersections:cc
3379       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3380       {\tl_use:N \l__spath_prefix_tl #2 \tl_use:N \l__spath_suffix_tl}
3381     }
3382   }
3383 },

```

Splits a path at its self-intersections.

```

3384 split~ at~ self~ intersections/.code={
3385   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3386   {
3387     \spath_split_at_self_intersections:c
3388     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3389   }
3390 },

```

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

```

3391 get~ components~ of/.code~ 2~ args={
3392   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3393   {
3394     \clist_clear_new:N #2
3395     \spath_components_to_seq:Nv
3396     \l__spath_tmpa_seq
3397     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3398     \seq_map_inline:Nn \l__spath_tmpa_seq
3399     {
3400       \tl_new:c
3401       {\tl_use:N \l__spath_prefix_tl anonymous_\int_use:N \g__spath_anon_int \tl_use:N \l__
3402       \tl_set:cn
3403       {\tl_use:N \l__spath_prefix_tl anonymous_\int_use:N \g__spath_anon_int \tl_use:N \l__
3404       \clist_put_right:Nx #2 {anonymous_\int_use:N \g__spath_anon_int}
3405       \int_gincr:N \g__spath_anon_int
3406     }
3407   }
3408 },

```

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

```

3409 render~ components/.code={
3410   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3411   {
3412     \group_begin:
3413     \spath_components_to_seq:Nv
3414     \l__spath_tmpa_seq
3415     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3416     \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
3417     {
3418       \spath_tikz_path:nn
3419       {
3420         every~ spath~ component/.try,
3421         spath ~component~ ##1/.try,
3422         spath ~component/.try={##1},
3423         every~ #1~ component/.try,
3424         #1 ~component~ ##1/.try,
3425         #1 ~component/.try={##1},
3426       }
3427       {
3428         ##2
3429       }
3430     }
3431     \group_end:
3432   }
3433 },

```

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.

```

3434 insert~ gaps~ after~ components/.code~ n~ args={3}{
3435   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3436   {
3437     \group_begin:
3438     \seq_gclear:N \g__spath_tmpa_seq
3439     \seq_gclear:N \g__spath_tmpb_seq
3440     \foreach \l__spath_tmpa_tl in {#3}
3441     {
3442       \seq_gput_right:Nv \g__spath_tmpa_seq \l__spath_tmpa_tl
3443       \seq_gput_right:Nx \g__spath_tmpb_seq {\int_eval:n { \l__spath_tmpa_tl + 1 }}
3444     }
3445     \spath_components_to_seq:Nv
3446     \l__spath_tmpa_seq
3447     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3448     \seq_clear:N \l__spath_tmpb_seq
3449     \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
3450     {
3451       \tl_set:Nn \l__spath_tmpa_tl {##2}
3452       \seq_if_in:NnT \g__spath_tmpa_seq {##1}
3453       {
3454         \spath_shorten_at_end:Nn \l__spath_tmpa_tl {#2/2}
3455       }
3456       \seq_if_in:NnT \g__spath_tmpb_seq {##1}
3457       {

```



```

3458     \spath_shorten_at_start:Nn \l__spath_tmpa_tl {#2/2}
3459   }
3460   \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
3461 }
3462 \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpb_seq {}}
3463 \group_end:
3464 \tl_set_eq:cn
3465 {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3466 \g__spath_output_tl
3467 \tl_gclear:N \g__spath_output_tl
3468 }
3469 },

```

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

```

3470 join~ components/.code~ 2~ args={
3471   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3472   {
3473     \seq_gclear:N \g__spath_tmpa_seq
3474     \foreach \l__spath_tmpa_tl in {#2}
3475     {
3476       \seq_gput_right:NV \g__spath_tmpa_seq \l__spath_tmpa_tl
3477     }
3478     \seq_gsort:Nn \g__spath_tmpa_seq
3479     {
3480       \int_compare:nNnTF {##1} > {##2}
3481       { \sort_return_same: }
3482       { \sort_return_swapped: }
3483     }
3484     \seq_map_inline:Nn \g__spath_tmpa_seq
3485     {
3486       \spath_join_component:cn
3487       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}{##1}
3488     }
3489   }
3490 },

```

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

```

3491 remove~ empty~ components/.code={
3492   \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3493   {
3494     \spath_remove_empty_components:c
3495     {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3496   }
3497 },

```

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.

```

3498 draft~ mode/.is~ choice,
3499 draft~ mode/true/.code={
3500   \bool_set_true:N \l__spath_draft_bool
3501 },
3502 draft~ mode/false/.code={
3503   \bool_set_false:N \l__spath_draft_bool

```

```

3504 },
3505 maybe~ spot~ weld/.code={
3506   \bool_if:NF \l__spath_draft_bool
3507   {
3508     \tl_if_exist:cT {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3509     {
3510       \spath_spot_weld_components:c
3511       {\tl_use:N \l__spath_prefix_tl #1 \tl_use:N \l__spath_suffix_tl}
3512     }
3513   }
3514 },

```

Set the transformation to lie along a path.

```

3515 transform~ to/.code~ 2~ args={
3516   \group_begin:
3517   \tl_if_exist:cTF
3518   {
3519     \tl_use:N \l__spath_prefix_tl
3520     #1
3521     \tl_use:N \l__spath_suffix_tl
3522   }
3523   {
3524     \spath_reallength:Nv
3525     \l__spath_tmpa_int
3526     {
3527       \tl_use:N \l__spath_prefix_tl
3528       #1
3529       \tl_use:N \l__spath_suffix_tl
3530     }
3531
3532     \tl_set:Nx \l__spath_tmpb_tl
3533     {\fp_to_decimal:n {(#2) * (\l__spath_tmpa_int)}}
3534     \spath_transformation_at:NvV \l__spath_tmpc_tl
3535     {
3536       \tl_use:N \l__spath_prefix_tl
3537       #1
3538       \tl_use:N \l__spath_suffix_tl
3539     }
3540     \l__spath_tmpb_tl
3541     \tl_gset_eq:NN \g__spath_smuggle_tl \l__spath_tmpc_tl
3542   }
3543   {
3544     \tl_gset_eq:NN \g__spath_smuggle_tl { {1}{0}{0}{1}{0pt}{0pt} }
3545   }
3546   \group_end:
3547   \exp_last_unbraced:NV \pgfsettransformentries \g__spath_smuggle_tl
3548   \tl_gclear:N \g__spath_smuggle_tl
3549 },

```

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

```

3550 upright~ transform~ to/.code~ 2~ args={
3551   \group_begin:
3552   \tl_if_exist:cTF

```

```

3553 {
3554   \tl_use:N \l__spath_prefix_tl
3555   #1
3556   \tl_use:N \l__spath_suffix_tl
3557 }
3558 {
3559   \spath_reallength:Nv
3560   \l__spath_tmpa_int
3561   {
3562     \tl_use:N \l__spath_prefix_tl
3563     #1
3564     \tl_use:N \l__spath_suffix_tl
3565   }
3566
3567   \tl_set:Nx \l__spath_tmpb_tl
3568   {\fp_to_decimal:n {(#2) * (\l__spath_tmpa_int)}}
3569   \spath_transformation_at:NvV \l__spath_tmpc_tl
3570   {
3571     \tl_use:N \l__spath_prefix_tl
3572     #1
3573     \tl_use:N \l__spath_suffix_tl
3574   }
3575   \l__spath_tmpb_tl
3576   \tl_gset_eq:NN \g__spath_smuggle_tl \l__spath_tmpc_tl
3577 }
3578 {
3579   \tl_gset_eq:NN \g__spath_smuggle_tl { {1}{0}{0}{1}{0pt}{0pt} }
3580 }
3581 \fp_compare:nT { \tl_item:Nn \g__spath_smuggle_tl {4} < 0 }
3582 {
3583   \tl_gset:Nx \g__spath_smuggle_tl
3584   {
3585     { \fp_eval:n { - (\tl_item:Nn \g__spath_smuggle_tl {1}) } }
3586     { \fp_eval:n { - (\tl_item:Nn \g__spath_smuggle_tl {2}) } }
3587     { \fp_eval:n { - (\tl_item:Nn \g__spath_smuggle_tl {3}) } }
3588     { \fp_eval:n { - (\tl_item:Nn \g__spath_smuggle_tl {4}) } }
3589     { \tl_item:Nn \g__spath_smuggle_tl {5} }
3590     { \tl_item:Nn \g__spath_smuggle_tl {6} }
3591   }
3592 }
3593 \group_end:
3594 \exp_last_unbraced:Nv \pgfsettransformentries \g__spath_smuggle_tl
3595 \tl_gclear:N \g__spath_smuggle_tl
3596 },

```

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

```

3597 knot/.style~ n~ args={3}{
3598   spath/.cd,
3599   split~ at~ self~ intersections=#1,
3600   insert~ gaps~ after~ components={#1}{#2}{#3},
3601   maybe~ spot~ weld=#1,
3602   render~ components=#1
3603 },
3604 }

```

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

```

3605 \tikzdeclarecoordinatesystem{spath}{%
3606   \group_begin:
3607   \tl_set:Nn \l__spath_tmpa_tl {#1}
3608   \tl_trim_spaces:N \l__spath_tmpa_tl
3609
3610   \seq_set_split:NnV \l__spath_tmpa_seq {~} \l__spath_tmpa_tl
3611   \seq_pop_right:NN \l__spath_tmpa_seq \l__spath_tmpb_tl
3612
3613   \tl_set:Nx \l__spath_tmpa_tl { \seq_use:Nn \l__spath_tmpa_seq {~} }
3614   \tl_if_exist:cTF
3615   {
3616     \tl_use:N \l__spath_prefix_tl
3617     \tl_use:N \l__spath_tmpa_tl
3618     \tl_use:N \l__spath_suffix_tl
3619   }
3620   {
3621
3622     \tl_set_eq:Nc
3623     \l__spath_tmpa_tl
3624     {
3625       \tl_use:N \l__spath_prefix_tl
3626       \tl_use:N \l__spath_tmpa_tl
3627       \tl_use:N \l__spath_suffix_tl
3628     }
3629
3630     \tl_if_empty:NTF \l__spath_tmpa_tl
3631     {
3632       \tl_gset_eq:NN \g__spath_smuggle_tl \pgfpointorigin
3633     }
3634     {
3635       \spath_reallength:NV \l__spath_tmpa_int \l__spath_tmpa_tl
3636       \tl_set:Nx \l__spath_tmpb_tl
3637       {\fp_to_decimal:n {(\l__spath_tmpb_tl) * (\l__spath_tmpa_int)}}
3638       \spath_point_at:NVV \l__spath_tmpc_tl \l__spath_tmpa_tl \l__spath_tmpb_tl
3639
3640       \tl_clear:N \l__spath_tmpd_tl
3641       \tl_put_right:Nn \l__spath_tmpd_tl {\pgf@x=}
3642       \tl_put_right:Nx \l__spath_tmpd_tl {\tl_item:Nn \l__spath_tmpc_tl {1}}
3643       \tl_put_right:Nn \l__spath_tmpd_tl {\relax}
3644       \tl_put_right:Nn \l__spath_tmpd_tl {\pgf@y=}
3645       \tl_put_right:Nx \l__spath_tmpd_tl {\tl_item:Nn \l__spath_tmpc_tl {2}}
3646       \tl_put_right:Nn \l__spath_tmpd_tl {\relax}
3647       \tl_gset_eq:NN \g__spath_smuggle_tl \l__spath_tmpd_tl
3648     }
3649   }
3650   {
3651     \tl_gset_eq:NN \g__spath_smuggle_tl \pgfpointorigin
3652   }
3653   \group_end:
3654   \tl_use:N \g__spath_smuggle_tl
3655 }
3656
3657 \ExplSyntaxOff

```

5 The Calligraphy Package

3658 <@@=cal>

5.1 Initialisation

```
3659 \RequirePackage{spath3}
3660 \ExplSyntaxOn
3661
3662 \tl_new:N \l__cal_tmpa_tl
3663 \tl_new:N \l__cal_tmpb_tl
3664 \tl_new:N \l__cal_tmp_path_tl
3665 \tl_new:N \l__cal_tmp_rpath_tl
3666 \tl_new:N \l__cal_tmp_rpathb_tl
3667 \tl_new:N \l__cal_tmp_patha_tl
3668
3669 \seq_new:N \l__cal_tmpa_seq
3670
3671 \int_new:N \l__cal_tmpa_int
3672 \int_new:N \l__cal_tmpb_int
3673 \int_new:N \g__cal_path_component_int
3674 \int_new:N \g__cal_label_int
3675
3676 \fp_new:N \l__cal_tmpa_fp
3677 \fp_new:N \l__cal_tmpb_fp
3678 \fp_new:N \l__cal_tmpe_fp
3679 \fp_new:N \l__cal_tmpe_fp
3680 \fp_new:N \l__cal_tmpe_fp
3681
3682 \dim_new:N \l__cal_tmpa_dim
3683 \dim_new:N \l__cal_tmpb_dim
3684 \dim_new:N \l__cal_tmpe_dim
3685 \dim_new:N \l__cal_tmpe_dim
3686 \dim_new:N \l__cal_tmpe_dim
3687 \dim_new:N \l__cal_tmpe_dim
3688 \dim_new:N \l__cal_tmpe_dim
3689 \dim_new:N \l__cal_tmpe_dim
3690
3691 \bool_new:N \l__cal_annotate_bool
3692 \bool_new:N \l__cal_taper_start_bool
3693 \bool_new:N \l__cal_taper_end_bool
3694 \bool_new:N \l__cal_taperable_bool
3695
3696 \dim_new:N \l__cal_taper_width_dim
3697 \dim_new:N \l__cal_line_width_dim
3698
3699 \bool_set_true:N \l__cal_taper_start_bool
3700 \bool_set_true:N \l__cal_taper_end_bool
3701
3702 \cs_generate_variant:Nn \tl_put_right:Nn {Nv}
3703
3704 \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.

```
3705 \tikzset{
3706   define-pen/.code={
3707     \tikzset{pen-name=#1}
3708     \pgf@relevantforpicturesizefalse
3709     \tikz@addmode{
3710       \pgfsyssoftpath@getcurrentpath\l__cal_tmpa_tl
3711       \spath_components_to_seq:NV \l__cal_tmpa_seq \l__cal_tmpa_tl
3712       \seq_gclear_new:c {g__cal_pen_\pgfkeysvalueof{/tikz/pen-name}_seq}
3713       \seq_gset_eq:cN {g__cal_pen_\pgfkeysvalueof{/tikz/pen-name}_seq} \l__cal_tmpa_seq
3714       \pgfusepath{discard}%
3715     }
3716   },
3717   define-pen/.default={default},
3718   use-pen/.code={
3719     \tikzset{pen-name=#1}
3720     \int_gzero:N \g__cal_path_component_int
3721     \cs_set_eq:NN \pgfpathmoveto \cal_moveto:n
3722     \tikz@addmode{
3723       \pgfsyssoftpath@getcurrentpath\l__cal_tmpa_tl
3724       \spath_components_to_seq:NV \l__cal_tmpa_seq \l__cal_tmpa_tl
3725       \tl_if_exist:cTF {g__cal_pen_\pgfkeysvalueof{/tikz/pen-name}_seq}
3726       {
3727         \cal_path_create:Nc \l__cal_tmpa_seq {g__cal_pen_\pgfkeysvalueof{/tikz/pen-name}_seq}
3728       }
3729       {
3730         \msg_warning:nxx { calligraphy } { undefined pen } { \pgfkeysvalueof{/tikz/pen-name} }
3731       }
3732     }
3733   },
3734   use-pen/.default={default},
3735   pen-name/.initial={default},
3736   copperplate/.style={pen-name=copperplate},
3737   pen-colour/.initial={black},
3738   weight/.is-choice,
3739   weight/heavy/.style={
3740     line-width=\pgfkeysvalueof{/tikz/heavy-line-width},
3741     taper-width=\pgfkeysvalueof{/tikz/light-line-width},
3742   },
3743   weight/light/.style={
3744     line-width=\pgfkeysvalueof{/tikz/light-line-width},
3745     taper-width=0pt,
3746   },
3747   heavy/.style={
3748     weight=heavy
3749   },
3750   light/.style={
3751     weight=light
3752   },
3753   heavy-line-width/.initial=2pt,
3754   light-line-width/.initial=1pt,
3755   taper/.is-choice,
```

```

3756 taper/.default=both,
3757 taper/none/.style={
3758   taper~start=false,
3759   taper~end=false,
3760 },
3761 taper/both/.style={
3762   taper~start=true,
3763   taper~end=true,
3764 },
3765 taper/start/.style={
3766   taper~start=true,
3767   taper~end=false,
3768 },
3769 taper/end/.style={
3770   taper~start=false,
3771   taper~end=true,
3772 },
3773 taper~start/.code={
3774   \tl_if_eq:nnTF {#1} {true}
3775   {
3776     \bool_set_true:N \l__cal_taper_start_bool
3777   }
3778   {
3779     \bool_set_false:N \l__cal_taper_start_bool
3780   }
3781 },
3782 taper~start/.default={true},
3783 taper~end/.code={
3784   \tl_if_eq:nnTF {#1} {true}
3785   {
3786     \bool_set_true:N \l__cal_taper_end_bool
3787   }
3788   {
3789     \bool_set_false:N \l__cal_taper_end_bool
3790   }
3791 },
3792 taper~end/.default={true},
3793 taper~width/.code={\dim_set:Nn \l__cal_taper_width_dim {#1}},
3794 nib~style/.code~2~args={
3795   \tl_clear_new:c {l__cal_nib_style_#1}
3796   \tl_set:cn {l__cal_nib_style_#1} {#2}
3797 },
3798 stroke~style/.code~2~args={
3799   \tl_clear_new:c {l__cal_stroke_style_#1}
3800   \tl_set:cn {l__cal_stroke_style_#1} {#2}
3801 },
3802 this~stroke~style/.code={
3803   \tl_clear_new:c {l__cal_stroke_inline_style_ \int_use:N \g__cal_path_component_int}
3804   \tl_set:cn {l__cal_stroke_inline_style_ \int_use:N \g__cal_path_component_int} {#1}
3805 },
3806 annotate/.style={
3807   annotate~if,
3808   annotate~reset,
3809   annotation~style/.update~value={#1},

```

```

3810 },
3811 annotate-if/.default={true},
3812 annotate-if/.code={
3813   \tl_if_eq:nnTF {#1} {true}
3814   {
3815     \bool_set_true:N \l__cal_annotate_bool
3816   }
3817   {
3818     \bool_set_false:N \l__cal_annotate_bool
3819   }
3820 },
3821 annotate-reset/.code={
3822   \int_gzero:N \g__cal_label_int
3823 },
3824 annotation~style/.initial={draw,->},
3825 annotation~shift/.initial={{0,lex}},
3826 every~annotation-node/.initial={anchor=south-west},
3827 annotation~node~style/.code~2~args={
3828   \tl_clear_new:c {l__cal_annotation_style_ #1 _tl}
3829   \tl_set:cn {l__cal_annotation_style_ #1 _tl}{#2}
3830 },
3831 tl~use:N/.code={
3832   \exp_args:NV \pgfkeysalso #1
3833 },
3834 tl~use:c/.code={
3835   \tl_if_exist:cT {#1}
3836   {
3837     \exp_args:Nv \pgfkeysalso {#1}
3838   }
3839 },
3840 /handlers/.update~style/.code={
3841   \tl_if_eq:nnF {#1} {\pgfkeysnovalue}
3842   {
3843     \pgfkeys{\pgfkeyscurrentpath/.code=\pgfkeysalso{#1}}
3844   }
3845 },
3846 /handlers/.update~value/.code={
3847   \tl_if_eq:nnF {#1} {\pgfkeysnovalue}
3848   {
3849     \pgfkeyssetvalue{\pgfkeyscurrentpath}{#1}
3850   }
3851 },
3852 }

```

Some wrappers around the TikZ keys.

```

3853 \NewDocumentCommand \pen { 0{} }
3854 {
3855   \path[define~ pen,every~ calligraphy~ pen/.try,#1]
3856 }
3857
3858 \NewDocumentCommand \definepen { 0{} }
3859 {
3860   \tikz \path[define~ pen,every~ calligraphy~ pen/.try,#1]
3861 }
3862

```



```

3863 \NewDocumentCommand \calligraphy { 0{} }
3864 {
3865   \path[use~ pen,every~ calligraphy/.try,#1]
3866 }

```

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

```

3867 \cs_new_protected_nopar:Npn \cal_path_create:NN #1#2
3868 {
3869   \int_zero:N \l__cal_tmpa_int
3870   \seq_map_inline:Nn #1
3871   {
3872     \int_compare:nT {\tl_count:n {##1} > 3}
3873     {
3874
3875       \int_incr:N \l__cal_tmpa_int
3876       \int_zero:N \l__cal_tmpb_int
3877
3878       \tl_set:Nn \l__cal_tmp_path_tl {##1}
3879       \spath_open:N \l__cal_tmp_path_tl
3880       \spath_reverse:NV \l__cal_tmp_rpath_tl \l__cal_tmp_path_tl
3881
3882       \seq_map_inline:Nn #2
3883       {
3884         \int_incr:N \l__cal_tmpb_int
3885         \group_begin:
3886         \pgfsys@beginscope
3887         \cal_apply_style:c {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_int}
3888         \cal_apply_style:c {l__cal_stroke_inline_style_ \int_use:N \l__cal_tmpa_int}
3889         \cal_apply_style:c {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}
3890
3891         \spath_initialpoint:Nn \l__cal_tmpa_tl {###1}
3892         \tl_set_eq:NN \l__cal_tmp_patha_tl \l__cal_tmp_path_tl
3893         \spath_translate:NV \l__cal_tmp_patha_tl \l__cal_tmpa_tl
3894
3895         \int_compare:nTF {\tl_count:n {###1} == 3}
3896         {
3897           \cal_at_least_three:N \l__cal_tmp_patha_tl
3898           \spath_protocol_path:V \l__cal_tmp_patha_tl
3899
3900           \tikz@options
3901           \dim_set:Nn \l__cal_line_width_dim {\pgflinewidth}
3902           \cal_maybe_taper:N \l__cal_tmp_patha_tl
3903         }
3904         {
3905           \spath_weld:Nn \l__cal_tmp_patha_tl {###1}
3906           \spath_weld:NV \l__cal_tmp_patha_tl \l__cal_tmp_rpath_tl
3907           \spath_reverse:Nn \l__cal_tmp_rpathb_tl {###1}
3908           \spath_weld:NV \l__cal_tmp_patha_tl \l__cal_tmp_rpathb_tl
3909
3910           \tl_clear:N \l__cal_tmpa_tl

```

```

3911     \tl_set:Nn \l__cal_tmpa_tl {fill=\pgfkeysvalueof{/tikz/pen~colour},draw=none}
3912     \tl_if_exist:cT {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_int}
3913     {
3914         \tl_put_right:Nv \l__cal_tmpa_tl {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_i
3915     }
3916     \tl_if_exist:cT {l__cal_stroke_inline_style_ \int_use:N \l__cal_tmpa_int}
3917     {
3918         \tl_put_right:Nn \l__cal_tmpa_tl {,}
3919         \tl_put_right:Nv \l__cal_tmpa_tl {l__cal_stroke_inline_style_ \int_use:N \l__cal
3920     }
3921     \tl_if_exist:cT {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}
3922     {
3923         \tl_put_right:Nn \l__cal_tmpa_tl {,}
3924         \tl_put_right:Nv \l__cal_tmpa_tl {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}
3925     }
3926     \spath_tikz_path:VV \l__cal_tmpa_tl \l__cal_tmp_patha_tl
3927
3928 }
3929 \pgfsys@endscope
3930 \group_end:
3931 }
3932
3933 \bool_if:NT \l__cal_annotate_bool
3934 {
3935     \seq_get_right:NN #2 \l__cal_tmpa_tl
3936     \spath_finalpoint:NV \l__cal_tmpa_tl \l__cal_tmpa_tl
3937     \spath_translate:NV \l__cal_tmp_path_tl \l__cal_tmpa_tl
3938     \tikz@scan@one@point\pgfutil@firstofone\pgfkeysvalueof{/tikz/annotation~shift}
3939
3940     \spath_translate:Nnn \l__cal_tmp_path_tl {\pgf@x} {\pgf@y}
3941
3942     \pgfkeysgetvalue{/tikz/annotation~style}{\l__cal_tmpa_tl}
3943     \spath_tikz_path:VV \l__cal_tmpa_tl \l__cal_tmp_path_tl
3944
3945     \spath_finalpoint:NV \l__cal_tmpa_tl \l__cal_tmp_path_tl
3946
3947     \exp_last_unbraced:NV \pgfqpoint \l__cal_tmpa_tl
3948     \begin{scope}[reset~ cm]
3949     \node[every~annotation~node/.try,tl~use:c = {l__cal_annotation_style_ \int_use:N \l
3950     \end{scope}
3951 }
3952 }
3953 }
3954 }
3955 \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.

```

3956 \cs_new_eq:NN \cal_orig_moveto:n \pgfpathmoveto
3957 \cs_new_nopar:Npn \cal_moveto:n #1
3958 {
3959     \int_gincr:N \g__cal_path_component_int

```

```

3960 \cal_orig_moveto:n {#1}
3961 }

```

(End definition for \cal_moveto:n.)

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

```

3962 \cs_new_nopar:Npn \cal_apply_style:N #1
3963 {
3964   \tl_if_exist:NT #1 {
3965     \exp_args:NV \tikzset #1
3966   }
3967 }
3968 \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.

```

3969 \cs_new_protected_nopar:Npn \cal_at_least_three:Nn #1#2
3970 {
3971   \spath_reallength:Nn \l__cal_tmpa_int {#2}
3972   \tl_clear:N \l__cal_tmpb_tl
3973   \tl_set:Nn \l__cal_tmpb_tl {#2}
3974   \int_compare:nTF {\l__cal_tmpa_int = 1}
3975   {
3976     \spath_split_at:Nn \l__cal_tmpb_tl {2/3}
3977     \spath_split_at:Nn \l__cal_tmpb_tl {1/2}
3978   }
3979   {
3980     \int_compare:nT {\l__cal_tmpa_int = 2}
3981     {
3982       \spath_split_at:Nn \l__cal_tmpb_tl {1.5}
3983       \spath_split_at:Nn \l__cal_tmpb_tl {.5}
3984     }
3985   }
3986   \tl_set_eq:NN #1 \l__cal_tmpb_tl
3987 }
3988 \cs_generate_variant:Nn \cal_at_least_three:Nn {NV}
3989 \cs_new_protected_nopar:Npn \cal_at_least_three:N #1
3990 {
3991   \cal_at_least_three:NV #1#1
3992 }
3993 \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.

```

3994 \cs_new_protected_nopar:Npn \cal_maybe_taper:N #1
3995 {
3996   \tl_set_eq:NN \l__cal_tmpa_tl #1
3997
3998   \bool_if:NT \l__cal_taper_start_bool
3999   {
4000
4001     \dim_set:Nn \l__cal_tmpa_dim {\tl_item:Nn \l__cal_tmpa_tl {2}}

```

```

4002 \dim_set:Nn \l__cal_tmpb_dim {\tl_item:Nn \l__cal_tmpa_tl {3}}
4003 \tl_set:Nx \l__cal_tmpb_tl {\tl_item:Nn \l__cal_tmpa_tl {4}}
4004
4005 \tl_case:NnF \l__cal_tmpb_tl
4006 {
4007   \c_spath_lineto_tl
4008   {
4009
4010     \bool_set_true:N \l__cal_taperable_bool
4011     \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {5}}
4012     \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {6}}
4013     \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpa_dim + \l__cal_tmph_dim)/3}
4014     \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpe_dim + \l__cal_tmph_dim)/3}
4015     \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmpa_dim + 2\l__cal_tmpe_dim)/3}
4016     \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmpe_dim + 2\l__cal_tmph_dim)/3}
4017     \prg_replicate:nn {4}
4018     {
4019       \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
4020     }
4021     \tl_put_left:NV \l__cal_tmpa_tl \c_spath_moveto_tl
4022   }
4023   \c_spath_curvetoa_tl
4024   {
4025     \bool_set_true:N \l__cal_taperable_bool
4026     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {5}}
4027     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {6}}
4028     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {8}}
4029     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {9}}
4030     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {11}}
4031     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {12}}
4032     \prg_replicate:nn {10}
4033     {
4034       \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
4035     }
4036     \tl_put_left:NV \l__cal_tmpa_tl \c_spath_moveto_tl
4037   }
4038 }
4039 {
4040   \bool_set_false:N \l__cal_taperable_bool
4041 }
4042
4043 \bool_if:NT \l__cal_taperable_bool
4044 {
4045   \__cal_taper_aux:
4046 }
4047
4048 }
4049
4050 \bool_if:NT \l__cal_taper_end_bool
4051 {
4052
4053   \dim_set:Nn \l__cal_tmpa_dim {\tl_item:Nn \l__cal_tmpa_tl {-2}}
4054   \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-1}}
4055   \tl_set:Nx \l__cal_tmpe_tl {\tl_item:Nn \l__cal_tmpa_tl {-3}}

```

```

4056
4057 \tl_case:NnF \l__cal_tmpb_tl
4058 {
4059   \c_spath_lineto_tl
4060   {
4061
4062     \bool_set_true:N \l__cal_taperable_bool
4063     \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {-5}}
4064     \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {-4}}
4065     \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpa_dim + \l__cal_tmpg_dim)/3}
4066     \dim_set:Nn \l__cal_tmpe_dim {(2\l__cal_tmpe_dim + \l__cal_tmph_dim)/3}
4067     \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmpe_dim + 2\l__cal_tmpe_dim)/3}
4068     \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmpe_dim + 2\l__cal_tmpe_dim)/3}
4069     \tl_reverse:N \l__cal_tmpa_tl
4070     \prg_replicate:nn {3}
4071     {
4072       \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
4073     }
4074     \tl_reverse:N \l__cal_tmpa_tl
4075   }
4076   \c_spath_curveto_tl
4077   {
4078     \bool_set_true:N \l__cal_taperable_bool
4079     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-5}}
4080     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-4}}
4081     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-8}}
4082     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-7}}
4083     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-11}}
4084     \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-10}}
4085     \tl_reverse:N \l__cal_tmpa_tl
4086     \prg_replicate:nn {9}
4087     {
4088       \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
4089     }
4090     \tl_reverse:N \l__cal_tmpa_tl
4091   }
4092 }
4093 {
4094   \bool_set_false:N \l__cal_taperable_bool
4095 }
4096
4097 \bool_if:NT \l__cal_taperable_bool
4098 {
4099   \__cal_taper_aux:
4100 }
4101
4102 }
4103
4104 \pgfsyssoftpath@setcurrentpath\l__cal_tmpa_tl
4105 \pgfsetstrokecolor{\pgfkeysvalueof{/tikz/pen~colour}}
4106 \pgfusepath{stroke}
4107
4108 }

```

(End definition for \cal_maybe_taper:N.)

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

```

4109 \cs_new_protected_nopar:Npn \__cal_taper_aux:
4110 {
4111   \tl_clear:N \l__cal_tmpb_tl
4112   \tl_put_right:NV \l__cal_tmpb_tl \c_spath_moveto_tl
4113
4114   \fp_set:Nn \l__cal_tmpe_dim
4115   {
4116     \l__cal_tmpe_dim - \l__cal_tmpe_dim
4117   }
4118   \fp_set:Nn \l__cal_tmpe_dim
4119   {
4120     \l__cal_tmpe_dim - \l__cal_tmpe_dim
4121   }
4122   \fp_set:Nn \l__cal_tmpe_dim
4123   {
4124     (\l__cal_tmpe_dim^2 + \l__cal_tmpe_dim^2)^.5
4125   }
4126
4127   \fp_set:Nn \l__cal_tmpe_dim { .5*\l__cal_taper_width_dim * \l__cal_tmpe_dim / \l__cal_tmpe_dim }
4128   \fp_set:Nn \l__cal_tmpe_dim { .5*\l__cal_taper_width_dim * \l__cal_tmpe_dim / \l__cal_tmpe_dim }
4129
4130   \fp_set:Nn \l__cal_tmpe_dim
4131   {
4132     \l__cal_tmpe_dim - \l__cal_tmpe_dim
4133   }
4134   \fp_set:Nn \l__cal_tmpe_dim
4135   {
4136     \l__cal_tmpe_dim - \l__cal_tmpe_dim
4137   }
4138   \fp_set:Nn \l__cal_tmpe_dim
4139   {
4140     (\l__cal_tmpe_dim^2 + \l__cal_tmpe_dim^2)^.5
4141   }
4142
4143   \fp_set:Nn \l__cal_tmpe_dim { .5*\l__cal_line_width_dim * \l__cal_tmpe_dim / \l__cal_tmpe_dim }
4144   \fp_set:Nn \l__cal_tmpe_dim { .5*\l__cal_line_width_dim * \l__cal_tmpe_dim / \l__cal_tmpe_dim }
4145
4146   \tl_put_right:Nx \l__cal_tmpb_tl
4147   {
4148     {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim }}
4149     {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim }}
4150   }
4151
4152   \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetoa_tl
4153
4154   \tl_put_right:Nx \l__cal_tmpb_tl
4155   {
4156     {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim }}
4157     {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \l__cal_tmpe_dim }}
4158   }
4159
4160   \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetob_tl
4161

```

```

4162 \tl_put_right:Nx \l__cal_tmpb_tl
4163 {
4164   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim}}
4165   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim}}
4166 }
4167
4168 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curveto_tl
4169
4170 \tl_put_right:Nx \l__cal_tmpb_tl
4171 {
4172   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim}}
4173   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim}}
4174 }
4175
4176 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetoa_tl
4177
4178 \tl_put_right:Nx \l__cal_tmpb_tl
4179 {
4180   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim - \fp_to_dim:n {1.32 * \l__cal_tmpe_dim}}}
4181   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpe_dim + \fp_to_dim:n {1.32 * \l__cal_tmpe_dim}}}
4182 }
4183
4184 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetob_tl
4185
4186 \tl_put_right:Nx \l__cal_tmpb_tl
4187 {
4188   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim - \fp_to_dim:n {1.32 * \l__cal_tmpe_dim}}}
4189   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim + \fp_to_dim:n {1.32 * \l__cal_tmpe_dim}}}
4190 }
4191
4192 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curveto_tl
4193
4194 \tl_put_right:Nx \l__cal_tmpb_tl
4195 {
4196   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim}}
4197   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim}}
4198 }
4199
4200 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetoa_tl
4201
4202 \tl_put_right:Nx \l__cal_tmpb_tl
4203 {
4204   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim}}
4205   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim}}
4206 }
4207
4208 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetob_tl
4209
4210 \tl_put_right:Nx \l__cal_tmpb_tl
4211 {
4212   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim}}
4213   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpe_dim}}
4214 }
4215

```

```

4216 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curveto_tl
4217
4218 \tl_put_right:Nx \l__cal_tmpb_tl
4219 {
4220   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim}}
4221   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim}}
4222 }
4223
4224 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetoa_tl
4225
4226 \tl_put_right:Nx \l__cal_tmpb_tl
4227 {
4228   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim + \fp_to_dim:n{ 1.32 * \l__cal_tmpa_dim}}
4229   {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim - \fp_to_dim:n{ 1.32 * \l__cal_tmpb_dim}}
4230 }
4231
4232 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curvetob_tl
4233
4234 \tl_put_right:Nx \l__cal_tmpb_tl
4235 {
4236   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim + \fp_to_dim:n{ 1.32 * \l__cal_tmpa_dim}}
4237   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim - \fp_to_dim:n{ 1.32 * \l__cal_tmpb_dim}}
4238 }
4239
4240 \tl_put_right:NV \l__cal_tmpb_tl \c_spath_curveto_tl
4241
4242 \tl_put_right:Nx \l__cal_tmpb_tl
4243 {
4244   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim}}
4245   {\dim_eval:n { \fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim}}
4246 }
4247
4248 \pgfsyssoftpath@setcurrentpath\l__cal_tmpb_tl
4249 \pgfsetfillcolor{\pgfkeysvalueof{/tikz/pen~colour}}
4250 \pgfusepath{fill}
4251 }

```

(End definition for _cal_taper_aux:.)

Defines a copperplate pen.

```

4252 \tl_set:Nn \l__cal_tmpa_tl {\pgfsyssoftpath@movetotoken{0pt}{0pt}}
4253 \spath_components_to_seq:NV \l__cal_tmpa_seq \l__cal_tmpa_tl
4254 \seq_gclear_new:N \g__cal_pen_copperplate_seq
4255 \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.

```

4256 \DeclareDocumentCommand \CopperplatePath { m }
4257 {
4258   \spath_components_to_seq:NV \l__cal_tmpa_seq #1
4259   \cal_path_create:NN \l__cal_tmpa_seq \g__cal_pen_copperplate_seq
4260 }

```

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

```

4261 \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.

```

4262 \expandafter\ifx\csname pgfdeclaredecoration\endcsname\relax
4263 \else
4264 \pgfdeclaredecoration{calligraphic brace}{brace}
4265 {
4266   \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]
4267   {
4268     \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}
4269     \pgfpathmoveto{\pgfpointorigin}
4270     \pgfpathcurveto
4271     {\pgfpoint{.15\pgfdecorationsegmentamplitude}{.3\pgfdecorationsegmentamplitude}}
4272     {\pgfpoint{.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
4273     {\pgfpoint{\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
4274     {
4275       \pgftransformxshift{+\pgfdecorationsegmentaspect\pgfdecoratedremainingdistance}
4276       \pgfpathlineto{\pgfpoint{-\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamp
4277       \pgfpathcurveto
4278       {\pgfpoint{-.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
4279       {\pgfpoint{-.15\pgfdecorationsegmentamplitude}{.7\pgfdecorationsegmentamplitude}}
4280       {\pgfpoint{0\pgfdecorationsegmentamplitude}{1\pgfdecorationsegmentamplitude}}
4281       \pgfpathmoveto{\pgfpoint{0\pgfdecorationsegmentamplitude}{1\pgfdecorationsegmentamp
4282       \pgfpathcurveto
4283       {\pgfpoint{.15\pgfdecorationsegmentamplitude}{.7\pgfdecorationsegmentamplitude}}
4284       {\pgfpoint{.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
4285       {\pgfpoint{\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
4286     }
4287     {
4288       \pgftransformxshift{+\pgfdecoratedremainingdistance}
4289       \pgfpathlineto{\pgfpoint{-\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamp
4290       \pgfpathcurveto
4291       {\pgfpoint{-.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
4292       {\pgfpoint{-.15\pgfdecorationsegmentamplitude}{.3\pgfdecorationsegmentamplitude}}
4293       {\pgfpoint{0pt}{0pt}}
4294     }
4295     \tikzset{
4296       taper width=.5\pgflinewidth,
4297       taper
4298     }%
4299     \pgfsyssoftpath@getcurrentpath\cal@tmp@path
4300     \CopperplatePath{\cal@tmp@path}
4301   }
4302   \state{final}{}
4303 }

```

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

```

4304 \pgfdeclaredecoration{calligraphic straight parenthesis}{brace}
4305 {
4306   \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]
4307   {

```

```

4308 \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}
4309 \pgfpathmoveto{\pgfpointorigin}
4310 \pgfpathcurveto
4311 {\pgfqqpoint{.76604\pgfdecorationsegmentamplitude}{.64279\pgfdecorationsegmentamplitude}}
4312 {\pgfqqpoint{2.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
4313 {\pgfqqpoint{3.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
4314 {
4315 \pgftransformxshift{+\pgfdecoratedremainingdistance}
4316 \pgfpathlineto{\pgfqqpoint{-3.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
4317 \pgfpathcurveto
4318 {\pgfqqpoint{-2.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
4319 {\pgfqqpoint{-.76604\pgfdecorationsegmentamplitude}{.64279\pgfdecorationsegmentamplitude}}
4320 {\pgfqqpoint{Opt}{Opt}}
4321 }
4322 \tikzset{
4323 taper width=.5\pgflinewidth,
4324 taper
4325 }%
4326 \pgfsyssoftpath@getcurrentpath\cal@tmp@path
4327 \CopperplatePath{\cal@tmp@path}
4328 }
4329 \state{final}{}%
4330 }

```

The third is a curved parenthesis.

```

4331 \pgfdeclaredecoration{calligraphic curved parenthesis}{brace}
4332 {
4333 \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]
4334 {
4335 \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}
4336 \pgfpathmoveto{\pgfpointorigin}
4337 \pgf@xa=\pgfdecoratedremainingdistance\relax
4338 \advance\pgf@xa by -1.5890\pgfdecorationsegmentamplitude\relax
4339 \edef\cgrphy@xa{the\pgf@xa}
4340 \pgfpathcurveto
4341 {\pgfqqpoint{1.5890\pgfdecorationsegmentamplitude}{1.3333\pgfdecorationsegmentamplitude}}
4342 {\pgfqqpoint{\cgrphy@xa}{1.3333\pgfdecorationsegmentamplitude}}
4343 {\pgfqqpoint{\pgfdecoratedremainingdistance}{Opt}}
4344 \tikzset{
4345 taper width=.5\pgflinewidth,
4346 taper
4347 }%
4348 \pgfsyssoftpath@getcurrentpath\cal@tmp@path
4349 \CopperplatePath{\cal@tmp@path}
4350 }
4351 \state{final}{}%
4352 }

```

End the conditional for if pgfdecoration module is loaded

```

4353 \fi

```

6 Drawing Knots

```

4354 <@@=knot>

```

6.1 Initialisation

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

```
4355 \RequirePackage{spath3}
4356 \usetikzlibrary{intersections,spath3}
4357
4358 \ExplSyntaxOn
4359
4360 \tl_new:N \l__knot_tmpa_tl
4361 \tl_new:N \l__knot_tmpb_tl
4362 \tl_new:N \l__knot_tmpc_tl
4363 \tl_new:N \l__knot_tmpd_tl
4364 \tl_new:N \l__knot_tmpg_tl
4365 \tl_new:N \l__knot_redraws_tl
4366 \tl_new:N \l__knot_clip_width_tl
4367 \tl_new:N \l__knot_name_tl
4368 \tl_new:N \l__knot_node_tl
4369 \tl_new:N \l__knot_aux_tl
4370 \tl_new:N \l__knot_auxa_tl
4371 \tl_new:N \l__knot_prefix_tl
4372
4373 \seq_new:N \l__knot_segments_seq
4374
4375 \int_new:N \l__knot_tmpa_int
4376 \int_new:N \l__knot_strands_int
4377 \int_new:N \g__knot_intersections_int
4378 \int_new:N \g__knot_filaments_int
4379 \int_new:N \l__knot_component_start_int
4380
4381 \fp_new:N \l__knot_tmpa_fp
4382 \fp_new:N \l__knot_tmpb_fp
4383
4384 \dim_new:N \l__knot_tmpa_dim
4385 \dim_new:N \l__knot_tmpb_dim
4386 \dim_new:N \l__knot_tolerance_dim
4387 \dim_new:N \l__knot_clip_bg_radius_dim
4388 \dim_new:N \l__knot_clip_draw_radius_dim
4389
4390 \bool_new:N \l__knot_draft_bool
4391 \bool_new:N \l__knot_ignore_ends_bool
4392 \bool_new:N \l__knot_self_intersections_bool
4393 \bool_new:N \l__knot_splits_bool
4394 \bool_new:N \l__knot_super_draft_bool
4395
4396 \bool_new:N \l__knot_prepend_prev_bool
4397 \bool_new:N \l__knot_append_next_bool
4398 \bool_new:N \l__knot_skip_bool
4399 \bool_new:N \l__knot_save_bool
4400
4401 \seq_new:N \g__knot_nodes_seq
4402
4403 \bool_set_true:N \l__knot_ignore_ends_bool
```

Configuration is via TikZ keys and styles.

```

4404 \tikzset{
4405   spath/prefix/knot/.style={
4406     spath/set~ prefix=knot strand,
4407   },
4408   spath/suffix/knot/.style={
4409     spath/set~ suffix={},
4410   },
4411   knot/.code={
4412     \tl_if_eq:nnTF {#1} {none}
4413     {
4414       \tikz@addmode{\tikz@mode@doublefalse}
4415     }
4416     {
4417       \tikz@addmode{\tikz@mode@doubletrue}
4418       \tl_if_eq:nnTF {\pgfkeysnovalue} {#1}
4419       {
4420         \tikz@addoption{\pgfsetinnerstrokecolor{.}}
4421       }
4422       {
4423         \pgfsetinnerstrokecolor{#1}
4424       }
4425       \tikz@addoption{
4426         \pgfsetstrokecolor{knotbg}
4427       }
4428       \tl_set:Nn \tikz@double@setup{
4429         \pgfsetinnerlinewidth{\pgflinewidth}
4430         \pgfsetlinewidth{\dim_eval:n {\tl_use:N \l__knot_gap_tl \pgflinewidth}}
4431       }
4432     }
4433   },
4434   knot~ gap/.store~ in=\l__knot_gap_tl,
4435   knot~ gap=3,
4436   knot~ diagram/.is~family,
4437   knot~ diagram/.unknown/.code={
4438     \tl_set_eq:NN \l__knot_tmpa_tl \pgfkeyscurrentname
4439     \pgfkeysalso{
4440       /tikz/\l__knot_tmpa_tl=#1
4441     }
4442   },
4443   background~ colour/.code={%
4444     \colorlet{knotbg}{#1}%
4445   },
4446   background~ color/.code={%
4447     \colorlet{knotbg}{#1}%
4448   },
4449   background~ colour=white,
4450   knot~ diagram,
4451   name/.store~ in=\l__knot_name_tl,
4452   name={knot},
4453   save~ intersections/.is~ choice,
4454   save~ intersections/.default=true,
4455   save~ intersections/true/.code={
4456     \bool_set_true:N \l__knot_save_bool
4457   },

```

```

4458 save~ intersections/false/.code={
4459   \bool_set_false:N \l__knot_save_bool
4460 },
4461 every~ strand/.style={draw},
4462 ignore~ endpoint~ intersections/.code={
4463   \tl_if_eq:nnTF {#1} {true}
4464   {
4465     \bool_set_true:N \l__knot_ignore_ends_bool
4466   }
4467   {
4468     \bool_set_false:N \l__knot_ignore_ends_bool
4469   }
4470 },
4471 ignore~ endpoint~ intersections/.default=true,
4472 consider~ self~ intersections/.is~choice,
4473 consider~ self~ intersections/true/.code={
4474   \bool_set_true:N \l__knot_self_intersections_bool
4475   \bool_set_true:N \l__knot_splits_bool
4476 },
4477 consider~ self~ intersections/false/.code={
4478   \bool_set_false:N \l__knot_self_intersections_bool
4479   \bool_set_false:N \l__knot_splits_bool
4480 },
4481 consider~ self~ intersections/no~ splits/.code={
4482   \bool_set_true:N \l__knot_self_intersections_bool
4483   \bool_set_false:N \l__knot_splits_bool
4484 },
4485 consider~ self~ intersections/.default={true},
4486 clip~ radius/.code={
4487   \dim_set:Nn \l__knot_clip_bg_radius_dim {#1}
4488   \dim_set:Nn \l__knot_clip_draw_radius_dim {#1+2pt}
4489 },
4490 clip~ draw~ radius/.code={
4491   \dim_set:Nn \l__knot_clip_draw_radius_dim {#1}
4492 },
4493 clip~ background~ radius/.code={
4494   \dim_set:Nn \l__knot_clip_bg_radius_dim {#1}
4495 },
4496 clip~ radius=10pt,
4497 end~ tolerance/.code={
4498   \dim_set:Nn \l__knot_tolerance_dim {#1}
4499 },
4500 end~ tolerance=14pt,
4501 clip/.style={
4502   clip
4503 },
4504 background~ clip/.style={
4505   clip
4506 },
4507 clip~ width/.code={
4508   \tl_set:Nn \l__knot_clip_width_tl {#1}
4509 },
4510 clip~ width=3,
4511 flip~ crossing/.code={%

```

```

4512     \tl_clear_new:c {l__knot_crossing_#1}
4513     \tl_set:cn {l__knot_crossing_#1} {x}
4514 },
4515 ignore~ crossing/.code={%
4516     \tl_clear_new:c {l__knot_ignore_crossing_#1}
4517     \tl_set:cn {l__knot_ignore_crossing_#1} {x}
4518 },
4519 draft~ mode/.is~ choice,
4520 draft~ mode/off/.code={%
4521     \bool_set_false:N \l__knot_draft_bool
4522     \bool_set_false:N \l__knot_super_draft_bool
4523 },
4524 draft~ mode/crossings/.code={%
4525     \bool_set_true:N \l__knot_draft_bool
4526     \bool_set_false:N \l__knot_super_draft_bool
4527 },
4528 draft~ mode/strands/.code={%
4529     \bool_set_true:N \l__knot_draft_bool
4530     \bool_set_true:N \l__knot_super_draft_bool
4531 },
4532 draft/.is~ family,
4533 draft,
4534 crossing~ label/.style={
4535     overlay,
4536     fill=white,
4537     fill~ opacity=.5,
4538     text~ opacity=1,
4539     text=blue,
4540     pin~ edge={blue,<-}
4541 },
4542 strand~ label/.style={
4543     overlay,
4544     circle,
4545     draw=purple,
4546     fill=white,
4547     fill~ opacity=.5,
4548     text~ opacity=1,
4549     text=purple,
4550     inner~ sep=0pt
4551 },
4552 }

```

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

```

4553 \cs_new_nopar:Npn \knot_apply_style:N #1
4554 {
4555     \tl_if_exist:NT #1 {
4556         \exp_args:NV \tikzset #1
4557     }
4558 }
4559 \cs_generate_variant:Nn \knot_apply_style:N {c}

```

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

```

4560 \NewDocumentCommand \flipcrossings {m}

```

```

4561 {
4562   \tikzset{knot~ diagram/flip~ crossing/.list={#1}}%
4563 }

```

(End definition for \flipcrossings.)

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

```

4564 \NewDocumentCommand \strand { 0{} }
4565 {
4566   \int_incr:N \l__knot_strands_int
4567   \tl_clear_new:c {l__knot_options_strand \int_use:N \l__knot_strands_int}
4568   \tl_set:cn {l__knot_options_strand \int_use:N \l__knot_strands_int} {#1}
4569   \path[#1,spath/set~ name=knot,spath/save=\int_use:N \l__knot_strands_int]
4570 }

```

(End definition for \strand.)

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

```

4571 \NewDocumentEnvironment{knot} { 0{} }
4572 {
4573   \knot_initialise:n {#1}
4574 }
4575 {
4576   \knot_render:
4577 }

```

(End definition for knot.)

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

```

4578 \cs_new_protected_nopar:Npn \knot_initialise:n #1
4579 {
4580   \tikzset{knot~ diagram/.cd,every~ knot~ diagram/.try,#1}
4581   \int_zero:N \l__knot_strands_int
4582   \tl_clear:N \l__knot_redraws_tl
4583   \seq_gclear:N \g__knot_nodes_seq
4584 }

```

(End definition for \knot_initialise:n.)

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

```

4585 \cs_new_protected_nopar:Npn \knot_render:
4586 {

```

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

```

4587   \pgfscope
4588   \pgftransformreset

```

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

```

4589   \int_step_function:nnnN {1} {1} {\l__knot_strands_int} \knot_draw_strand:n

```

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

```

4590   \bool_if:NF \l__knot_super_draft_bool
4591   {

```

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.

```
4592 \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.

```
4593 \bool_if:NTF \l__knot_self_intersections_bool
4594 {
4595   \knot_split_strands:
4596   \int_set_eq:NN \l__knot_tmpa_int \g__knot_filaments_int
4597   \tl_set:Nn \l__knot_prefix_tl {filament}
4598 }
4599 {
4600   \int_set_eq:NN \l__knot_tmpa_int \l__knot_strands_int
4601   \tl_set:Nn \l__knot_prefix_tl {strand}
4602 }
```

Initialise the intersection count.

```
4603 \int_gzero:N \g__knot_intersections_int
```

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

```
4604 \tl_clear:N \l__knot_node_tl
4605 \bool_if:NT \l__knot_draft_bool
4606 {
4607   \tl_set:Nn \l__knot_node_tl {
4608     \exp_not:N \node[coordinate,
4609       pin={\node~ contents={\int_use:N \g__knot_intersections_int},knot~ diagram/draft/
4610       }]}
4611   }
4612 }
```

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

```
4613 \int_step_variable:nnnNn {1} {1} {\l__knot_tmpa_int - 1} \l__knot_tmpa_tl
4614 {
4615   \int_step_variable:nnnNn {\tl_use:N \l__knot_tmpa_tl + 1} {1}      {\l__knot_tmpa_int}
4616   {
4617     \knot_intersections:VV \l__knot_tmpa_tl \l__knot_tmpb_tl
4618   }
4619 }
```

If any redraws were requested, do them here.

```
4620 \tl_use:N \l__knot_redraws_tl
```

Draw the crossing nodes

```
4621 \seq_use:Nn \g__knot_nodes_seq {}
4622 }
```

Close the scope

```
4623 \endpgfscope
4624 }
```

(End definition for \knot_render:.)

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

```

4625 \cs_new_protected_nopar:Npn \knot_draw_strand:n #1
4626 {
4627   \pgfscope
4628   \group_begin:
4629   \spath_bake_round:c {knot strand #1}
4630   \tl_set:Nn \l__knot_tmpa_tl {knot~ diagram/every~ strand/.try,}
4631   \tl_put_right:Nv \l__knot_tmpa_tl {l__knot_options_strand #1}
4632   \tl_put_right:Nn \l__knot_tmpa_tl {,knot~ diagram/only~ when~ rendering/.try,only~ when~ r
4633   \spath_tikz_path:Vv \l__knot_tmpa_tl {knot strand #1}
4634   \group_end:
4635   \endpgfscope
4636 }
4637 \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.

```

4638 \cs_new_protected_nopar:Npn \knot_draw_labels:n #1
4639 {
4640   \bool_if:NT \l__knot_draft_bool
4641   {
4642     \spath_finalpoint:Nv \l__knot_tmpb_tl {knot strand #1}
4643     \dim_set:Nn \l__knot_tmpa_dim {\tl_item:Nn \l__knot_tmpb_tl {1}}
4644     \dim_set:Nn \l__knot_tmpb_dim {\tl_item:Nn \l__knot_tmpb_tl {2}}
4645     \node[knot~ diagram/draft/strand~label] at (\l__knot_tmpa_dim,\l__knot_tmpb_dim) {#1};
4646     \spath_initialpoint:Nv \l__knot_tmpb_tl {knot strand #1}
4647     \dim_set:Nn \l__knot_tmpa_dim {\tl_item:Nn \l__knot_tmpb_tl {1}}
4648     \dim_set:Nn \l__knot_tmpb_dim {\tl_item:Nn \l__knot_tmpb_tl {2}}
4649     \node[knot~ diagram/draft/strand~label] at (\l__knot_tmpa_dim,\l__knot_tmpb_dim) {#1};
4650   }
4651   \bool_if:nT {
4652     \l__knot_self_intersections_bool
4653     &&
4654     \l__knot_splits_bool
4655   }
4656   {
4657     \tl_clear:N \l__knot_tmpa_tl
4658     \spath_segments_to_seq:Nv \l__knot_segments_seq {knot strand #1}
4659     \seq_map_function:NN \l__knot_segments_seq \knot_split_self_intersects:N
4660     \tl_set_eq:cN {knot strand #1} \l__knot_tmpa_tl
4661   }
4662 }

```

(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!

```

4663 \cs_new_protected_nopar:Npn \knot_split_self_intersects:N #1
4664 {
4665   \tl_set:Nx \l__knot_tmpc_tl {\tl_item:nn {#1} {4}}

```

```

4666 \tl_case:NnF \l__knot_tmpc_tl
4667 {
4668   \c_spath_curvetoa_tl
4669   {
4670     \fp_set:Nn \l__knot_tmpa_fp
4671     {
4672       (\tl_item:nn {#1} {3} - 3 * \tl_item:nn {#1} {6} + 3 * \tl_item:nn {#1} {9} - \tl_it
4673       *
4674       (3 * \tl_item:nn {#1} {8} - 3 * \tl_item:nn {#1} {11}))
4675       -
4676       (\tl_item:nn {#1} {2} - 3 * \tl_item:nn {#1} {5} + 3 * \tl_item:nn {#1} {8} - \tl_it
4677       *
4678       (3 * \tl_item:nn {#1} {9} - 3 * \tl_item:nn {#1} {12}))
4679     }
4680     \fp_set:Nn \l__knot_tmpb_fp
4681     {
4682       (\tl_item:nn {#1} {2} - 3 * \tl_item:nn {#1} {5} + 3 * \tl_item:nn {#1} {8} - \tl_it
4683       *
4684       (3 * \tl_item:nn {#1} {6} - 6 * \tl_item:nn {#1} {9} + 3 * \tl_item:nn {#1} {12}))
4685       -
4686       (\tl_item:nn {#1} {3} - 3 * \tl_item:nn {#1} {6} + 3 * \tl_item:nn {#1} {9} - \tl_it
4687       *
4688       (3 * \tl_item:nn {#1} {5} - 6 * \tl_item:nn {#1} {8} + 3 * \tl_item:nn {#1} {11}))
4689     }
4690     \fp_compare:nTF
4691     {
4692       \l__knot_tmpb_fp != 0
4693     }
4694     {
4695       \fp_set:Nn \l__knot_tmpa_fp { .5 * \l__knot_tmpa_fp / \l__knot_tmpb_fp }
4696       \fp_compare:nTF
4697       {
4698         0 < \l__knot_tmpa_fp && \l__knot_tmpa_fp < 1
4699       }
4700       {
4701         \spath_split_curve:NNnV \l__knot_tmpc_tl \l__knot_tmpd_tl {#1} \l__knot_tmpa_fp
4702         \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
4703         \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
4704         \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
4705         \tl_set:Nx \l__knot_tmpd_tl {\tl_tail:N \l__knot_tmpd_tl}
4706         \tl_set:Nx \l__knot_tmpd_tl {\tl_tail:N \l__knot_tmpd_tl}
4707         \tl_set:Nx \l__knot_tmpd_tl {\tl_tail:N \l__knot_tmpd_tl}
4708         \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpc_tl
4709         \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpd_tl
4710       }
4711       {
4712         \tl_set:Nn \l__knot_tmpc_tl {#1}
4713         \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
4714         \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
4715         \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
4716         \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpc_tl
4717       }
4718     }
4719   }

```

```

4720     \tl_set:Nn \l__knot_tmpc_tl {#1}
4721     \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
4722     \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
4723     \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
4724     \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpc_tl
4725   }
4726 }
4727 \c_spath_lineto_tl
4728 {
4729   \tl_set:Nn \l__knot_tmpc_tl {#1}
4730   \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
4731   \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
4732   \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
4733   \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpc_tl
4734 }
4735 }
4736 {
4737   \tl_put_right:Nn \l__knot_tmpa_tl {#1}
4738 }
4739 }

```

(End definition for \knot_split_self_intersects:N.)

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

```

4740 \cs_new_protected_nopar:Npn \knot_intersections:nn #1#2
4741 {
4742   \group_begin:
4743   \tl_set_eq:NN \l__knot_tmpa_tl \l__knot_prefix_tl
4744   \tl_put_right:Nn \l__knot_tmpa_tl {#1}
4745   \tl_set_eq:NN \l__knot_tmpb_tl \l__knot_prefix_tl
4746   \tl_put_right:Nn \l__knot_tmpb_tl {#2}
4747   \tl_set_eq:Nc \l__knot_tmpc_tl {knot \tl_use:N \l__knot_tmpa_tl}
4748   \tl_set_eq:Nc \l__knot_tmpd_tl {knot \tl_use:N \l__knot_tmpb_tl}
4749
4750   \bool_if:nTF {
4751     \l__knot_save_bool
4752     &&
4753     \tl_if_exist_p:c {knot~ intersections~ \tl_use:N \l__knot_name_tl - \tl_use:N \l__knot_t
4754   }
4755   {
4756     \tl_use:c {knot~ intersections~ \tl_use:N \l__knot_name_tl - \tl_use:N \l__knot_tmpa_tl
4757   }
4758   {
4759     \pgfintersectionofpaths{\pgfsetpath\l__knot_tmpc_tl}{\pgfsetpath\l__knot_tmpd_tl}
4760
4761   }
4762
4763   \int_compare:nT {\pgfintersectionsolutions > 0}
4764   {
4765     \int_step_function:nnnN {1} {1} {\pgfintersectionsolutions} \knot_do_intersection:n
4766   }
4767
4768   \knot_save_intersections:VV \l__knot_tmpa_tl \l__knot_tmpb_tl
4769   \group_end:
4770 }

```

(End definition for \knot_intersections:nn.)

\knot_save_intersections:nn

```

4771 \cs_new_protected_nopar:Npn \knot_save_intersections:nn #1#2
4772 {
4773   \bool_if:NT \l__knot_save_bool
4774   {
4775     \tl_clear:N \l__knot_aux_tl
4776     \tl_put_right:Nn \l__knot_aux_tl
4777     {
4778       \def\pgfintersectionsolutions
4779       }
4780     \tl_put_right:Nx \l__knot_aux_tl
4781     {
4782       {\int_eval:n {\pgfintersectionsolutions}}
4783     }
4784     \int_compare:nT {\pgfintersectionsolutions > 0}
4785     {
4786       \int_step_inline:nnnn {1} {1} {\pgfintersectionsolutions}
4787       {
4788         \pgfpointintersectionsolution{##1}
4789         \dim_set:Nn \l__knot_tmpa_dim {\pgf@x}
4790         \dim_set:Nn \l__knot_tmpb_dim {\pgf@y}
4791         \tl_put_right:Nn \l__knot_aux_tl
4792         {
4793           \expandafter\def\csname pgfpoint@intersect@solution@##1\endcsname
4794         }
4795         \tl_put_right:Nx \l__knot_aux_tl
4796         {
4797           {\exp_not:N \pgf@x=\dim_use:N \l__knot_tmpa_dim\exp_not:N\relax\exp_not:N \pgf@y =
4798         }
4799       }
4800     \tl_set:Nn \l__knot_auxa_tl {\expandafter \gdef \csname knot~ intersections~}
4801     \tl_put_right:Nx \l__knot_auxa_tl {\tl_use:N \l__knot_name_tl - #1 - #2}
4802     \tl_put_right:Nn \l__knot_auxa_tl {\endcsname}
4803     \tl_put_right:Nx \l__knot_auxa_tl {{\tl_to_str:N \l__knot_aux_tl}}
4804     \protected@write\@auxout{}\{\tl_to_str:N \l__knot_auxa_tl}
4805   }
4806 }
4807 }
4808 \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.

```

4809 \cs_new_protected_nopar:Npn \knot_do_intersection:n #1
4810 {
4811   \pgfpointintersectionsolution{#1}
4812   \dim_set:Nn \l__knot_tmpa_dim {\pgf@x}
4813   \dim_set:Nn \l__knot_tmpb_dim {\pgf@y}

```

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

```
4814 \bool_set_false:N \l__knot_skip_bool
4815 \bool_if:NT \l__knot_self_intersections_bool
4816 {
```

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

```
4817 \tl_set:Nn \l__knot_tmpc_tl {knot previous}
4818 \tl_put_right:NV \l__knot_tmpc_tl \l__knot_tmpa_tl
4819 \tl_set:Nv \l__knot_tmpc_tl \l__knot_tmpc_tl
4820 \tl_if_eq:NNT \l__knot_tmpc_tl \l__knot_tmpb_tl
4821 {
4822 \knot_test_endpoint:VnT \l__knot_tmpb_tl {final point}
4823 {
4824 \bool_set_true:N \l__knot_skip_bool
4825 }
4826 }
4827
4828 \tl_set:Nn \l__knot_tmpc_tl {knot previous}
4829 \tl_put_right:NV \l__knot_tmpc_tl \l__knot_tmpb_tl
4830 \tl_set:Nv \l__knot_tmpc_tl \l__knot_tmpc_tl
4831 \tl_if_eq:NNT \l__knot_tmpc_tl \l__knot_tmpa_tl
4832 {
4833 \knot_test_endpoint:VnT \l__knot_tmpa_tl {final point}
4834 {
4835 \bool_set_true:N \l__knot_skip_bool
4836 }
4837 }
4838 }
```

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

```
4839 \bool_if:NT \l__knot_ignore_ends_bool
4840 {
4841 \knot_test_endpoint:VnT \l__knot_tmpa_tl {initial point}
4842 {
4843 \bool_set_true:N \l__knot_skip_bool
4844 }
4845 \knot_test_endpoint:VnT \l__knot_tmpa_tl {final point}
4846 {
4847 \bool_set_true:N \l__knot_skip_bool
4848 }
4849 \knot_test_endpoint:VnT \l__knot_tmpb_tl {initial point}
4850 {
4851 \bool_set_true:N \l__knot_skip_bool
4852 }
4853 \knot_test_endpoint:VnT \l__knot_tmpb_tl {final point}
4854 {
4855 \bool_set_true:N \l__knot_skip_bool
4856 }
4857 }
```

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

```
4858 \bool_if:NF \l__knot_skip_bool
4859 {
```

```

4860
4861     \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.

```

4862     \bool_if:nF
4863     {
4864         \tl_if_exist_p:c {l__knot_ignore_crossing_ \int_use:N
4865             \g__knot_intersections_int}
4866         &&
4867         ! \tl_if_empty_p:c {l__knot_ignore_crossing_ \int_use:N
4868             \g__knot_intersections_int}
4869     }
4870     {

```

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

```

4871         \bool_if:nTF
4872         {
4873             \tl_if_exist_p:c {l__knot_crossing_ \int_use:N
4874                 \g__knot_intersections_int}
4875             &&
4876             ! \tl_if_empty_p:c {l__knot_crossing_ \int_use:N
4877                 \g__knot_intersections_int}
4878         }
4879         {
4880             \tl_set_eq:NN \l__knot_tmpg_tl \l__knot_tmpb_tl
4881         }
4882         {
4883             \tl_set_eq:NN \l__knot_tmpg_tl \l__knot_tmpa_tl
4884         }

```

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.

```

4885     \bool_if:NT \l__knot_self_intersections_bool
4886     {
4887         \knot_test_endpoint:VnT \l__knot_tmpg_tl {initial point}
4888         {
4889             \bool_set_true:N \l__knot_prepend_prev_bool
4890         }
4891         {
4892             \bool_set_false:N \l__knot_prepend_prev_bool
4893         }
4894         \knot_test_endpoint:VnT \l__knot_tmpg_tl {final point}
4895         {
4896             \bool_set_true:N \l__knot_append_next_bool
4897         }
4898         {
4899             \bool_set_false:N \l__knot_append_next_bool
4900         }

```

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

```

4901         \bool_if:nT
4902         {

```

```

4903     \l__knot_prepend_prev_bool || \l__knot_append_next_bool
4904 }
4905 {
4906     \tl_clear_new:c {knot \tl_use:N \l__knot_prefix_tl -1}
4907     \tl_set_eq:cc {knot \tl_use:N \l__knot_prefix_tl -1} {knot \tl_use:N \l__knot_tmpg_tl}
4908
4909     \tl_clear_new:c {l__knot_options_ \tl_use:N \l__knot_prefix_tl -1}
4910     \tl_set_eq:cc {l__knot_options_ \tl_use:N \l__knot_prefix_tl -1} {l__knot_options_}
4911
4912     \bool_if:nT
4913     {
4914         \l__knot_prepend_prev_bool
4915         &&
4916         \tl_if_exist_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
4917         &&
4918         !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
4919     }
4920     {
4921         \spath_prepend_no_move:cv {knot \tl_use:N \l__knot_prefix_tl -1} {knot \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.

```

4922     \bool_if:nT
4923     {
4924         \l__knot_splits_bool
4925         &&
4926         \tl_if_exist_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
4927         &&
4928         !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
4929     }
4930     {
4931         \knot_test_endpoint:vnT {knot previous \tl_use:N \l__knot_tmpg_tl} {initial point}
4932         {
4933
4934             \spath_prepend_no_move:cv {knot \tl_use:N \l__knot_prefix_tl -
4935 1} {knot \tl_use:c {knot previous \tl_use:c {knot previous \tl_use:N \l__knot_tmpg_tl}}}
4936             \tl_set_eq:Nc \l__knot_tmpa_tl {knot \tl_use:N \l__knot_prefix_tl -
4937 1}
4938         }

```

Now the same for appending.

```

4939     \bool_if:nT
4940     {
4941         \l__knot_append_next_bool
4942         &&
4943         \tl_if_exist_p:c {knot next \tl_use:N \l__knot_tmpg_tl}
4944         &&
4945         !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
4946     }
4947     {
4948         \spath_append_no_move:cv {knot \tl_use:N \l__knot_prefix_tl -1} {knot \tl_use:c {knot next \tl_use:c {knot next \tl_use:N \l__knot_tmpg_tl}}}
4949         \bool_if:nT
4950         {

```

```

4951     \l__knot_splits_bool
4952     &&
4953     \tl_if_exist_p:c {knot previous \tl_use:N
4954       \l__knot_tmpg_tl}
4955     &&
4956     !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
4957   }
4958   {
4959     \knot_test_endpoint:vnT {knot previous \tl_use:N \l__knot_tmpg_tl} {final point}
4960     {
4961       \spath_append_no_move:cv {knot \tl_use:N \l__knot_prefix_tl -1} {knot \tl_use:N \l__knot_tmpg_tl}
4962     }
4963   }
4964 }
4965 }
4966
4967   \tl_set:Nn \l__knot_tmpg_tl {\tl_use:N \l__knot_prefix_tl -1}
4968 }
4969 }

```

Now we render the crossing.

```

4970   \pgfscope
4971   \group_begin:
4972   \tikzset{knot~ diagram/every~ intersection/.try, every~ intersection/.try, knot~ diagram/every~ intersection/.try}
4973   \knot_draw_crossing:VVV \l__knot_tmpg_tl \l__knot_tmpa_dim \l__knot_tmpb_dim
4974   \coordinate (\l__knot_name_tl \c_space_tl \int_use:N \g__knot_intersections_int) at (\l__knot_name_tl \c_space_tl \int_use:N \g__knot_intersections_int)
4975   \group_end:
4976   \endpgfscope

```

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

```

4977   }

```

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

```

4978   \tl_if_empty:NF \l__knot_node_tl
4979   {
4980     \seq_gpush:Nx \g__knot_nodes_seq { \l__knot_node_tl at (\dim_use:N \l__knot_tmpa_dim, \dim_use:N \l__knot_tmpb_dim) }
4981   }
4982 }
4983 }
4984
4985 \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.

```

4986 \prg_new_conditional:Npnn \knot_test_endpoint:N #1 {p,T,F,TF}
4987 {
4988   \dim_compare:nTF
4989   {
4990     \dim_abs:n { \l__knot_tmpa_dim - \tl_item:Nn #1 {1} }
4991     +
4992     \dim_abs:n { \l__knot_tmpb_dim - \tl_item:Nn #1 {2} }
4993     <
4994     \l__knot_tolerance_dim
4995   }

```



```

4996 {
4997   \prg_return_true:
4998 }
4999 {
5000   \prg_return_false:
5001 }
5002 }

```

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

`\knot_test_endpoint:nn` Wrapper around the above.

```

5003 \prg_new_protected_conditional:Npnn \knot_test_endpoint:nn #1#2 {T,F,TF}
5004 {
5005   \use:c {spath_#2:Nv} \l__knot_tmpd_tl {knot #1}
5006   \knot_test_endpoint:NTF \l__knot_tmpd_tl
5007   {
5008     \prg_return_true:
5009   }
5010   {
5011     \prg_return_false:
5012   }
5013 }
5014
5015 \cs_generate_variant:Nn \knot_test_endpoint:nnT {VnT,vnT}
5016 \cs_generate_variant:Nn \knot_test_endpoint:nnF {VnF,vnF}
5017 \cs_generate_variant:Nn \knot_test_endpoint:nnTF {VnTF,vnTF}

```

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

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

```

5018 \cs_new_protected_nopar:Npn \knot_draw_crossing:nnn #1#2#3
5019 {
5020   \group_begin:
5021   \pgfscope
5022   \path[knot~ diagram/background~ clip] (#2, #3) circle[radius=\l__knot_clip_bg_radius_dim];
5023
5024   \tl_set:Nn \l__knot_tmpa_tl {knot~ diagram/every~ strand/.try,}
5025   \tl_if_exist:cT {l__knot_options_ #1}
5026   {
5027     \tl_put_right:Nv \l__knot_tmpa_tl {l__knot_options_ #1}
5028   }
5029   \tl_put_right:Nn \l__knot_tmpa_tl {,knotbg,line~ width= \tl_use:N \l__knot_clip_width_tl *
5030   \spath_tikz_path:Vv \l__knot_tmpa_tl {knot #1}
5031
5032   \endpgfscope
5033
5034   \pgfscope
5035   \path[knot~ diagram/clip] (#2, #3) circle[radius=\l__knot_clip_draw_radius_dim];
5036
5037   \tl_set:Nn \l__knot_tmpa_tl {knot~ diagram/every~ strand/.try,}
5038   \tl_if_exist:cT {l__knot_options_ #1}
5039   {
5040     \tl_put_right:Nv \l__knot_tmpa_tl {l__knot_options_ #1}
5041   }
5042   \tl_put_right:Nn \l__knot_tmpa_tl {,knot~ diagram/only~ when~ rendering/.try,only~ when~ r

```

```

5043 \spath_tikz_path:Vv \l__knot_tmpa_tl {knot #1}
5044
5045 \endpgfscope
5046 \group_end:
5047 }
5048
5049 \cs_generate_variant:Nn \knot_draw_crossing:nnn {nVV, VVV}
5050
5051 \cs_new_protected_nopar:Npn \knot_draw_crossing:nn #1#2
5052 {
5053 \tikz@scan@one@point\pgfutil@firstofone #2 \relax
5054 \knot_draw_crossing:nVV {#1} \pgf@x \pgf@y
5055 }

```

(End definition for \knot_draw_crossing:nnn.)

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

```

5056 \cs_new_protected_nopar:Npn \knot_split_strands:
5057 {
5058 \int_gzero:N \g__knot_filaments_int
5059 \int_step_function:nnnN {1} {1} {\l__knot_strands_int} \knot_split_strand:n
5060 \int_step_function:nnnN {1} {1} {\g__knot_filaments_int} \knot_compute_nexts:n
5061 }

```

(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.

```

5062 \cs_new_protected_nopar:Npn \knot_compute_nexts:n #1
5063 {
5064 \tl_clear_new:c {knot next \tl_use:c {knot previous filament #1}}
5065 \tl_set:cn {knot next \tl_use:c {knot previous filament #1}} {filament #1}
5066 }

```

(End definition for \knot_compute_nexts:n.)

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

```

5067 \cs_new_protected_nopar:Npn \knot_split_strand:n #1
5068 {
5069 \int_set_eq:NN \l__knot_component_start_int \g__knot_filaments_int
5070 \int_incr:N \l__knot_component_start_int
5071 \tl_set_eq:Nc \l__knot_tmpa_tl {l__knot_options_strand #1}
5072 \spath_segments_to_seq:Nv \l__knot_segments_seq {knot strand #1}
5073 \seq_map_function:NN \l__knot_segments_seq \knot_save_filament:N
5074 }

```

(End definition for \knot_split_strand:n.)

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

```

5075 \cs_new_protected_nopar:Npn \knot_save_filament:N #1
5076 {
5077 \tl_set:Nx \l__knot_tmpb_tl {\tl_item:nn {#1} {4}}
5078 \tl_case:NnF \l__knot_tmpb_tl
5079 {
5080 \c_spath_moveto_tl

```

```

5081 {
5082   \int_compare:nT {\l__knot_component_start_int < \g__knot_filaments_int}
5083   {
5084     \int_set_eq:NN \l__knot_component_start_int \g__knot_filaments_int
5085   }
5086 }
5087 \c_spath_lineto_tl
5088 {
5089   \int_gincr:N \g__knot_filaments_int
5090   \tl_clear_new:c {knot filament \int_use:N \g__knot_filaments_int}
5091   \tl_set:cn {knot filament \int_use:N \g__knot_filaments_int} {#1}
5092
5093   \tl_clear_new:c {l__knot_options_filament \int_use:N \g__knot_filaments_int}
5094   \tl_set_eq:cN {l__knot_options_filament \int_use:N \g__knot_filaments_int} \l__knot_tm
5095
5096   \tl_clear_new:c {knot previous filament \int_use:N \g__knot_filaments_int}
5097   \int_compare:nF {\l__knot_component_start_int == \g__knot_filaments_int}
5098   {
5099     \tl_set:cx {knot previous filament \int_use:N \g__knot_filaments_int} {filament \int
5100   }
5101 }
5102 \c_spath_curvetoa_tl
5103 {
5104   \int_gincr:N \g__knot_filaments_int
5105   \tl_clear_new:c {knot filament \int_use:N \g__knot_filaments_int}
5106   \tl_set:cn {knot filament \int_use:N \g__knot_filaments_int} {#1}
5107   \tl_clear_new:c {l__knot_options_filament \int_use:N \g__knot_filaments_int}
5108   \tl_set_eq:cN {l__knot_options_filament \int_use:N \g__knot_filaments_int} \l__knot_tm
5109
5110   \tl_clear_new:c {knot previous filament \int_use:N \g__knot_filaments_int}
5111   \int_compare:nF {\l__knot_component_start_int == \g__knot_filaments_int}
5112   {
5113     \tl_set:cx {knot previous filament \int_use:N \g__knot_filaments_int} {filament \int
5114   }
5115 }
5116 \c_spath_closepath_tl
5117 {
5118   \int_gincr:N \g__knot_filaments_int
5119   \tl_clear_new:c {knot filament \int_use:N \g__knot_filaments_int}
5120   \tl_clear:N \l__knot_tmpa_tl
5121   \tl_put_right:Nx {\tl_item:nn {#1} {1}}{\tl_item:nn {#1} {2}}{\tl_item:nn {#1} {3}}
5122   \tl_put_right:NV \l__knot_tmpa_tl \c_spath_lineto_tl
5123   \tl_put_right:Nx {\tl_item:nn {#1} {5}}{\tl_item:nn {#1} {6}}
5124
5125   \tl_set:cV {knot filament \int_use:N \g__knot_filaments_int} \l__knot_tmpa_tl
5126   \tl_set_eq:cN {l__knot_options_filament \int_use:N \g__knot_filaments_int} \l__knot_tm
5127   \tl_clear_new:c {knot previous filament \int_use:N \g__knot_filaments_int}
5128   \int_compare:nF {\l__knot_component_start_int == \g__knot_filaments_int}
5129   {
5130     \tl_set:cx {knot previous filament \int_use:N \g__knot_filaments_int} {filament \int
5131   }
5132   \tl_set:cx {knot previous filament \int_use:N \l__knot_component_start_int} {filament
5133 }
5134 }

```

```

5135 {
5136 }
5137 }

```

(End definition for \knot_save_filament:N.)

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

```

5138 \NewDocumentCommand \redraw { m m }
5139 {
5140 % \tikz@scan@one@point\pgfutil@firstofone #2 \relax
5141 \tl_put_right:Nn \l__knot_redraws_tl {\knot_draw_crossing:nm}
5142 \tl_put_right:Nx \l__knot_redraws_tl {
5143 {strand #1} {#2}% {\dim_use:N \pgf@x} {\dim_use:N \pgf@y}
5144 }
5145 }

```

(End definition for \redraw.)

```

5146 \ExplSyntaxOff

```

<@@=>

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

```

5147 \def\pgf@sh__knotknotanchor#1#2{%
5148 \anchor{#2 north west}{%
5149 \csname pgf@anchor@knot #1@north west\endcsname%
5150 \pgf@x=#2\pgf@x%
5151 \pgf@y=#2\pgf@y%
5152 }%
5153 \anchor{#2 north east}{%
5154 \csname pgf@anchor@knot #1@north east\endcsname%
5155 \pgf@x=#2\pgf@x%
5156 \pgf@y=#2\pgf@y%
5157 }%
5158 \anchor{#2 south west}{%
5159 \csname pgf@anchor@knot #1@south west\endcsname%
5160 \pgf@x=#2\pgf@x%
5161 \pgf@y=#2\pgf@y%
5162 }%
5163 \anchor{#2 south east}{%
5164 \csname pgf@anchor@knot #1@south east\endcsname%
5165 \pgf@x=#2\pgf@x%
5166 \pgf@y=#2\pgf@y%
5167 }%
5168 \anchor{#2 north}{%
5169 \csname pgf@anchor@knot #1@north\endcsname%
5170 \pgf@x=#2\pgf@x%
5171 \pgf@y=#2\pgf@y%
5172 }%
5173 \anchor{#2 east}{%
5174 \csname pgf@anchor@knot #1@east\endcsname%
5175 \pgf@x=#2\pgf@x%
5176 \pgf@y=#2\pgf@y%
5177 }%
5178 \anchor{#2 west}{%
5179 \csname pgf@anchor@knot #1@west\endcsname%

```

```

5180     \pgf@x=#2\pgf@x%
5181     \pgf@y=#2\pgf@y%
5182   }%
5183   \anchor{#2 south}{%
5184     \csname pgf@anchor@knot #1@south\endcsname%
5185     \pgf@x=#2\pgf@x%
5186     \pgf@y=#2\pgf@y%
5187   }%
5188 }

```

(End definition for \pgf@sh__knotknotanchor.)

knot_crossing

```

5189 \pgfdeclareshape{knot crossing}
5190 {
5191   \inheritsavedanchors[from=circle] % this is nearly a circle
5192   \inheritanchorborder[from=circle]
5193   \inheritanchor[from=circle]{north}
5194   \inheritanchor[from=circle]{north west}
5195   \inheritanchor[from=circle]{north east}
5196   \inheritanchor[from=circle]{center}
5197   \inheritanchor[from=circle]{west}
5198   \inheritanchor[from=circle]{east}
5199   \inheritanchor[from=circle]{mid}
5200   \inheritanchor[from=circle]{mid west}
5201   \inheritanchor[from=circle]{mid east}
5202   \inheritanchor[from=circle]{base}
5203   \inheritanchor[from=circle]{base west}
5204   \inheritanchor[from=circle]{base east}
5205   \inheritanchor[from=circle]{south}
5206   \inheritanchor[from=circle]{south west}
5207   \inheritanchor[from=circle]{south east}
5208   \inheritanchorborder[from=circle]
5209   \pgf@sh__knotknotanchor{crossing}{2}
5210   \pgf@sh__knotknotanchor{crossing}{3}
5211   \pgf@sh__knotknotanchor{crossing}{4}
5212   \pgf@sh__knotknotanchor{crossing}{8}
5213   \pgf@sh__knotknotanchor{crossing}{16}
5214   \pgf@sh__knotknotanchor{crossing}{32}
5215   \backgroundpath{
5216     \pgfutil@tempdima=\radius%
5217     \pgfmathsetlength{\pgf@xb}{\pgfkeysvalueof{/pgf/outer xsep}}%
5218     \pgfmathsetlength{\pgf@yb}{\pgfkeysvalueof{/pgf/outer ysep}}%
5219     \ifdim\pgf@xb<\pgf@yb%
5220       \advance\pgfutil@tempdima by-\pgf@yb%
5221     \else%
5222       \advance\pgfutil@tempdima by-\pgf@xb%
5223     \fi%
5224   }
5225 }

```

(End definition for knot crossing.)

knot_over_cross

```

5226 \pgfdeclareshape{knot over cross}
5227 {
5228   \inheritsavedanchors[from=rectangle] % this is nearly a circle
5229   \inheritanchorborder[from=rectangle]
5230   \inheritanchor[from=rectangle]{north}
5231   \inheritanchor[from=rectangle]{north west}
5232   \inheritanchor[from=rectangle]{north east}
5233   \inheritanchor[from=rectangle]{center}
5234   \inheritanchor[from=rectangle]{west}
5235   \inheritanchor[from=rectangle]{east}
5236   \inheritanchor[from=rectangle]{mid}
5237   \inheritanchor[from=rectangle]{mid west}
5238   \inheritanchor[from=rectangle]{mid east}
5239   \inheritanchor[from=rectangle]{base}
5240   \inheritanchor[from=rectangle]{base west}
5241   \inheritanchor[from=rectangle]{base east}
5242   \inheritanchor[from=rectangle]{south}
5243   \inheritanchor[from=rectangle]{south west}
5244   \inheritanchor[from=rectangle]{south east}
5245   \inheritanchorborder[from=rectangle]
5246   \backgroundpath{
5247     \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
5248     \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
5249     \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
5250     \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
5251   }
5252   \foregroundpath{
5253     % store lower right in xa/ya and upper right in xb/yb
5254     \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
5255     \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
5256     \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
5257     \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
5258   }
5259 }

```

(End definition for knot over cross.)

knot_□under_□cross

```

5260 \pgfdeclareshape{knot under cross}
5261 {
5262   \inheritsavedanchors[from=rectangle] % this is nearly a circle
5263   \inheritanchorborder[from=rectangle]
5264   \inheritanchor[from=rectangle]{north}
5265   \inheritanchor[from=rectangle]{north west}
5266   \inheritanchor[from=rectangle]{north east}
5267   \inheritanchor[from=rectangle]{center}
5268   \inheritanchor[from=rectangle]{west}
5269   \inheritanchor[from=rectangle]{east}
5270   \inheritanchor[from=rectangle]{mid}
5271   \inheritanchor[from=rectangle]{mid west}
5272   \inheritanchor[from=rectangle]{mid east}
5273   \inheritanchor[from=rectangle]{base}
5274   \inheritanchor[from=rectangle]{base west}
5275   \inheritanchor[from=rectangle]{base east}

```

```

5276 \inheritanchor[from=rectangle]{south}
5277 \inheritanchor[from=rectangle]{south west}
5278 \inheritanchor[from=rectangle]{south east}
5279 \inheritanchorborder[from=rectangle]
5280 \backgroundpath{
5281   \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
5282   \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
5283   \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
5284   \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
5285 }
5286 \foregroundpath{
5287 % store lower right in xa/ya and upper right in xb/yb
5288   \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
5289   \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
5290   \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
5291   \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
5292 }
5293 }

```

(End definition for knot under cross.)

knot_vert

```

5294 \pgfdeclareshape{knot vert}
5295 {
5296   \inheritssavedanchors[from=rectangle] % this is nearly a circle
5297   \inheritanchorborder[from=rectangle]
5298   \inheritanchor[from=rectangle]{north}
5299   \inheritanchor[from=rectangle]{north west}
5300   \inheritanchor[from=rectangle]{north east}
5301   \inheritanchor[from=rectangle]{center}
5302   \inheritanchor[from=rectangle]{west}
5303   \inheritanchor[from=rectangle]{east}
5304   \inheritanchor[from=rectangle]{mid}
5305   \inheritanchor[from=rectangle]{mid west}
5306   \inheritanchor[from=rectangle]{mid east}
5307   \inheritanchor[from=rectangle]{base}
5308   \inheritanchor[from=rectangle]{base west}
5309   \inheritanchor[from=rectangle]{base east}
5310   \inheritanchor[from=rectangle]{south}
5311   \inheritanchor[from=rectangle]{south west}
5312   \inheritanchor[from=rectangle]{south east}
5313   \inheritanchorborder[from=rectangle]
5314   \backgroundpath{
5315 % store lower right in xa/ya and upper right in xb/yb
5316   \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
5317   \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
5318   \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
5319   \pgfpathlineto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
5320   \pgfpathmoveto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
5321   \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
5322 }
5323 }

```

(End definition for knot vert.)

knot_□horiz

```
5324 \pgfdeclareshape{knot horiz}
5325 {
5326   \inheritsavedanchors[from=rectangle] % this is nearly a circle
5327   \inheritanchorborder[from=rectangle]
5328   \inheritanchor[from=rectangle]{north}
5329   \inheritanchor[from=rectangle]{north west}
5330   \inheritanchor[from=rectangle]{north east}
5331   \inheritanchor[from=rectangle]{center}
5332   \inheritanchor[from=rectangle]{west}
5333   \inheritanchor[from=rectangle]{east}
5334   \inheritanchor[from=rectangle]{mid}
5335   \inheritanchor[from=rectangle]{mid west}
5336   \inheritanchor[from=rectangle]{mid east}
5337   \inheritanchor[from=rectangle]{base}
5338   \inheritanchor[from=rectangle]{base west}
5339   \inheritanchor[from=rectangle]{base east}
5340   \inheritanchor[from=rectangle]{south}
5341   \inheritanchor[from=rectangle]{south west}
5342   \inheritanchor[from=rectangle]{south east}
5343   \inheritanchorborder[from=rectangle]
5344   \foregroundpath{
5345     % store lower right in xa/ya and upper right in xb/yb
5346     \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
5347     \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
5348     \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
5349     \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
5350     \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
5351     \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
5352   }
5353 }
```

(End definition for knot horiz.)