The spath3 package

Andrew Stacey
stacey@math.ntnu.no

v1.0 from 2013/04/09

1 Introduction

The `spath3` package is intended as a library for manipulating PGF's *soft paths*. In between defining a path and using it, PGF stores a path as a *soft path* were all the defining structure has been resolved into the basic operations but these have not yet been written to the output file. They can therefore still be manipulated by TEX, and as they have a very rigid form (and limited vocabulary), they are relatively easy to modify. This package provides some methods for working with these paths. It is not really intended for use by end users but as a foundation on which other packages can be built. As examples, the `calligraphy` package and the `knot` package are included. The first of these simulates a calligraphic pen stroking a path. The second can be used to draw knot (and similar) diagrams.

The format of a soft path is a sequence of triples of the form `\macro {dimension}{dimension}`. The macro is one of a short list, the dimensions are coordinates in points. There are certain further restrictions, particularly that every path must begin with a `move to`, and Bézier curves consist of three triples.

2 Implementation

2.1 Initialisation

Load the LATEX3 foundation and register us as a LATEX3 package.

```
\NeedsTeXFormat{LaTeX2e}
\RequirePackage{expl3}
\RequirePackage{pgf}
\ProvidesExplPackage {spath3} {2013/04/09} {1.0} {Functions for
manipulating PGF soft paths}
\RequirePackage{xparse}
```

We need a slew of temporary variables.

```
\tl_new:N \l__spath_tmpa_tl
\tl_new:N \l__spath_tmpb_tl
\tl_new:N \l__spath_tmpc_tl
\tl_new:N \l__spath_tmpd_tl
```

```
\tl_new:N \l__spath_smuggle_tl
\dim_new:N \l__spath_tmpa_dim
\dim_new:N \l__spath_tmpb_dim
\int_new:N \l__spath_tmpa_int
```

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

```
\tl_new:N \g__spath_moveto_tl
\tl_new:N \g__spath_lineto_tl
\tl_new:N \g__spath_curveto_tl
\tl_new:N \g__spath_curvetoa_tl
\tl_new:N \g__spath_curvetob_tl
\tl_set:Nn \g__spath_moveto_tl {\pgfsyssoftpath@movetotoken}
\tl_set:Nn \g__spath_lineto_tl {\pgfsyssoftpath@linetotoken}
\tl_set:Nn \g__spath_curveto_tl {\pgfsyssoftpath@curvetotoken}
\tl_set:Nn \g__spath_curvetoa_tl {\pgfsyssoftpath@curvetosupportatoken}
\tl_set:Nn \g__spath_curvetob_tl {\pgfsyssoftpath@curvetosupportbtoken}
```

2.2 Basic Structure and Methods

A soft path is a `prop`. These are lists of the attributes that we define. The first consists of all attributes, the second of those that are "moveable" in the sense that they change if we transform the path, the third are the ones that contain actual paths.

Note that if using these attributes outside an `expl3` context, the spaces should be omitted.

```
\tl_new:N \g__spath_attributes
\tl_new:N \g__spath_moveable_attributes
\tl_new:N \g__spath_path_attributes
\tl_set:Nn \g__spath_attributes {
  {path}
  {reverse path}
  {length}
  {real length}
  {number of components}
  {initial point}
  {final point}
  {initial action}
  {final action}
  {min bb}
  {max bb}
}
\tl_set:Nn \g__spath_moveable_attributes {
  {initial point}
  {final point}
  {min bb}
  {max bb}
}
\tl_set:Nn \g__spath_path_attributes {
  {path}
```

```
  {reverse path}
}
```

An `spath` object is actually a `prop`. The following functions are wrappers around the underlying `prop` functions. We prefix the names to avoid clashing with other `prop`s that might be lying around, this is why all the `spath` methods take argument `:n` and not `:N`. Given that `spath` objects might be created inside a group but used outside it, we work globally throughout.

`\spath_new:n`

```
\cs_new_nopar:Npn \spath_new:n #1
{
  \prop_new:c {l__spath_#1}
}
```

`\spath_clear:n`

```
\cs_new_nopar:Npn \spath_clear:n #1
{
  \prop_gclear:c {l__spath_#1}
}
```

`\spath_clear_new:n`

```
\cs_new_nopar:Npn \spath_clear_new:n #1
{
  \prop_gclear_new:c {l__spath_#1}
}
```

`\spath_show:n`

```
\cs_new_nopar:Npn \spath_show:n #1
{
  \prop_show:c {l__spath_#1}
}
```

`\spath_put:nnn`

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

`\spath_remove:nn`

```
\cs_new_nopar:Npn \spath_remove:nn #1#2
{
  \prop_gremove:cn {l__spath_#1} {#2}
}
```

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

```
\cs_new_nopar:Npn \__spath_get:nn #1#2
{
  \prop_get:cn {l__spath_#1} {#2}
}
```

__spath_get:nnN

```
\cs_new_nopar:Npn \__spath_get:nnN #1#2#3
{
  \prop_get:cnN {l__spath_#1} {#2} #3
}
```

\spath_if_in:nn

```
\prg_new_conditional:Npnn \spath_if_in:nn #1#2 {p, T, F, TF}
{
  \prop_if_in:cnTF {l__spath_#1} {#2}
  { \prg_return_true: }
  { \prg_return_false: }
}
```

_spath_get:nnN

```
\cs_generate_variant:Nn \__prop_split:NnTF {cnTF}
\prg_new_protected_conditional:Npnn \__spath_get:nnN #1#2#3 {T, F, TF}
{
  \__prop_split:cnTF {l__spath_#1} {#2}
  { \__prop_get_true:Nnnn #3 }
  { \prg_return_false: }
}
\cs_generate_variant:Nn \spath_put:nnn {nnV, nnx, nno}
\cs_generate_variant:Nn \__spath_get:nn {Vn}
```

\spath_if_exist:n

```
\prg_new_conditional:Npnn \spath_if_exist:n #1 {p,T,F,TF}
{
  \prop_if_exist:cTF {l__spath_#1}
  {
    \prg_return_true:
  }
  {
    \prg_return_false:
  }
}
```

\spath_clone:nn Clones an spath.

```
\cs_new_nopar:Npn \spath_clone:nn #1 #2
{
  \spath_clear_new:n {#2}
  \tl_map_inline:Nn \g__spath_attributes
  {
    \spath_if_in:nnT {#1} {##1}
    {
      \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
      \spath_put:nnV {#2} {##1} \l__spath_tmpa_tl
    }
  }
}
```

\spath_get_current_path:n

```
\cs_new_protected_nopar:Npn \spath_get_current_path:n #1
{
  \pgfsyssoftpath@getcurrentpath\l__knot_tmpa_tl
  \spath_clear_new:n {#1}
  \spath_put:nnV {#1} {path} \l__knot_tmpa_tl
}
```

\spath_set_current_path:n

```
\cs_new_protected_nopar:Npn \spath_set_current_path:n #1
{
  \spath_get:nnN {#1} {min bb} \l__spath_tmpa_tl
  \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl

  \spath_get:nnN {#1} {max bb} \l__spath_tmpa_tl
  \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl

  \spath_get:nnN {#1} {path} \l__spath_tmpa_tl
  \pgfsyssoftpath@setcurrentpath\l__spath_tmpa_tl
  \pgfsyssoftpath@flushcurrentpath
}
```

\spath_use_path:nn

```
\cs_new_protected_nopar:Npn \spath_use_path:nn #1#2
{
  \spath_set_current_path:n {#1}
  \pgfusepath{#2}
}
```

\spath_protocol_path:n

```
\cs_new_protected_nopar:Npn \spath_protocol_path:n #1
{
  \spath_get:nnN {#1} {min bb} \l__spath_tmpa_tl
  \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl

  \spath_get:nnN {#1} {max bb} \l__spath_tmpa_tl
  \exp_last_unbraced:NV \pgf@protocolsizes\l__spath_tmpa_tl
}
```

\spath_tikz_path:nn

```
\cs_new_protected_nopar:Npn \spath_tikz_path:nn #1#2
{
  \path[#1] \pgfextra{
    \spath_get:nnN {#2} {path} \l__spath_tmpa_tl
    \pgfsyssoftpath@setcurrentpath \l__spath_tmpa_tl
  };
}
\cs_generate_variant:Nn \spath_tikz_path:nn {Vn}
```

2.3 Computing Information

`\spath_get:nn` The information that we store along with a soft path can be computed from it, but computing it every time is wasteful. So this is the real `\spath_get:nn` function which checks to see if we have already computed it and then either retrieves it or computes it.

```
\cs_new_nopar:Npn \spath_get:nn #1#2
{
  \spath_if_in:nnF {#1} {#2}
  {
    \cs_if_exist_use:cT {spath_generate_#2:n} {{#1}}
  }
  \__spath_get:nn {#1} {#2}
}
```

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

```
\cs_new_nopar:Npn \spath_get:nnN #1#2#3
{
  \spath_if_in:nnF {#1} {#2}
  {
    \cs_if_exist_use:cT {spath_generate_#2:n} {{#1}}
  }
  \__spath_get:nnN {#1} {#2} #3
}
\cs_generate_variant:Nn \spath_get:nnN {nnV}
```

The next slew of functions generate data from the original path, storing it in the `prop` for further retrieval.

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

```
\cs_new_nopar:Npn \spath_generate_length:n #1
{
  \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
  \spath_put:nnx {#1} {length} {\int_eval:n {\tl_count:N \l__spath_tmpa_tl /3 }}
}
```

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

```
\cs_new_nopar:Npn \spath_generate_reallength:n #1
{
  \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
  \int_set:Nn \l__spath_tmpa_int {0}
  \tl_map_inline:Nn \l__spath_tmpa_tl {
    \tl_if_eq:nnT {##1} {\pgfsyssoftpath@linetotoken}
    {
      \int_incr:N \l__spath_tmpa_int
    }
    \tl_if_eq:nnT {##1} {\pgfsyssoftpath@curvetotoken}
    {
      \int_incr:N \l__spath_tmpa_int
    }
```

```
  }
  \spath_put:nnx {#1} {real length} {\int_use:N \l__spath_tmpa_int}
}
```

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

```
\cs_new_nopar:Npn \spath_generate_numberofcomponents:n #1
{
  \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
  \int_set:Nn \l__spath_tmpa_int {0}
  \tl_map_inline:Nn \l__spath_tmpa_tl {
    \tl_if_eq:nnT {##1} {\pgfsyssoftpath@movetotoken}
    {
      \int_incr:N \l__spath_tmpa_int
    }
  }
  \spath_put:nnx {#1} {number of components} {\int_use:N \l__spath_tmpa_int}
}
```

\spath_generate_initialpoint:n The starting point of the path.

```
\cs_new_nopar:Npn \spath_generate_initialpoint:n #1
{
  \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
  \tl_clear:N \l__spath_tmpb_tl
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
  \spath_put:nnV {#1} {initial point} \l__spath_tmpb_tl
}
```

\spath_generate_finalpoint:n The final point of the path.

```
\cs_new_nopar:Npn \spath_generate_finalpoint:n #1
{
  \tl_clear:N \l__spath_tmpb_tl
  \spath_if_in:nnTF {#1} {reverse path}
  {
    \__spath_get:nnN {#1} {reverse path} \l__spath_tmpa_tl
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
    \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
    \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
  }
  {
    \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
    \tl_reverse:N \l__spath_tmpa_tl
    \tl_put_left:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
    \tl_put_left:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
```

```
  }
  \spath_put:nnV {#1} {final point} \l__spath_tmpb_tl
}
```

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

```
\cs_new_nopar:Npn \spath_generate_reversepath:n #1
{
  \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
  \tl_clear:N \l__spath_tmpb_tl
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \tl_put_left:Nx \l__spath_tmpb_tl
  {
    {\dim_use:N \l__spath_tmpa_dim}
    {\dim_use:N \l__spath_tmpb_dim}
  }
  \bool_until_do:nn {
    \tl_if_empty_p:N \l__spath_tmpa_tl
  }
  {
    \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
    \tl_set:Nn \l__spath_tmpd_tl {}
    \tl_case:Nnn \l__spath_tmpc_tl
    {
      \g__spath_moveto_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_moveto_tl }
      \g__spath_lineto_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_lineto_tl }
      \g__spath_curveto_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curvetoa_tl }
      \g__spath_curvetoa_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curveto_tl }
      \g__spath_curvetob_tl {\tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curvetob_tl }
    }
    {
      \tl_show:N \l__spath_tmpc_tl
    }
    \tl_put_left:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
    \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
    \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
    \tl_put_left:Nx \l__spath_tmpb_tl
    {
      {\dim_use:N \l__spath_tmpa_dim}
      {\dim_use:N \l__spath_tmpb_dim}
    }
  }
  \tl_put_left:NV \l__spath_tmpb_tl \g__spath_moveto_tl
  \spath_put:nnV {#1} {reverse path} \l__spath_tmpb_tl
```

```
}
```

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

```
\cs_new_nopar:Npn \spath_generate_initialaction:n #1
{
  \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
  \tl_clear:N \l__spath_tmpb_tl
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \tl_if_empty:NF \l__spath_tmpa_tl {
    \tl_set:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
  }
  \spath_put:nnV {#1} {initial action} \l__spath_tmpb_tl
}
```

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

```
\cs_new_nopar:Npn \spath_generate_finalaction:n #1
{
  \tl_clear:N \l__spath_tmpb_tl
  \spath_if_in:nnTF {#1} {reverse path}
  {
    \__spath_get:nnN {#1} {reverse path} \l__spath_tmpa_tl
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  }
  {
    \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
    \tl_reverse:N \l__spath_tmpa_tl
  }
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \tl_if_empty:NF \l__spath_tmpa_tl {
    \tl_set:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
  }
  \tl_if_eq:NNT \l__spath_tmpa_tl \g__spath_curvetoa_tl
  {
    \tl_set_eq:NN \l__spath_tmpa_tl \g__spath_curveto_tl
  }
  \spath_put:nnV {#1} {final action} \l__spath_tmpb_tl
}
```

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

```
\cs_new_nopar:Npn \spath_generate_minbb:n #1
{
  \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
```

```
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \bool_until_do:nn {
    \tl_if_empty_p:N \l__spath_tmpa_tl
  }
  {
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
    \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tmpa
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
    \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tmpb
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  }
  \tl_clear:N \l__spath_tmpb_tl
  \tl_put_right:Nx \l__spath_tmpb_tl
  {
    {\dim_use:N \l__spath_tmpa_dim}
    {\dim_use:N \l__spath_tmpb_dim}
  }
  \spath_put:nnV {#1} {min bb} \l__spath_tmpb_tl
}
```

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

```
\cs_new_nopar:Npn \spath_generate_maxbb:n #1
{
  \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \bool_until_do:nn {
    \tl_if_empty_p:N \l__spath_tmpa_tl
  }
  {
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
    \dim_set:Nn \l__spath_tmpa_dim {\dim_max:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tmpa
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
    \dim_set:Nn \l__spath_tmpb_dim {\dim_max:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tmpb
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  }
  \tl_clear:N \l__spath_tmpb_tl
  \tl_put_right:Nx \l__spath_tmpb_tl
  {
    {\dim_use:N \l__spath_tmpa_dim}
    {\dim_use:N \l__spath_tmpb_dim}
  }
  \spath_put:nnV {#1} {max bb} \l__spath_tmpb_tl
}
```

`\spath_generate_all:n` This function generates all of the data in one fell swoop. By traversing the path just once it is quicker than doing each one individually. However, it does need to store a lot

of data as it goes.

- `\l__spath_rp_tl` will hold the reversed path
- `\l__spath_l_int` will hold the length
- `\l__spath_rl_int` will hold the real length
- `\l__spath_nc_int` will hold the number of components
- `\l__spath_ip_tl` will hold the initial point
- `\l__spath_fp_tl` will hold the final point
- `\l__spath_ia_tl` will hold the initial action
- `\l__spath_fa_tl` will hold the final action
- `\l__spath_minx_dim` will hold the min x bb
- `\l__spath_miny_dim` will hold the min y bb
- `\l__spath_maxx_dim` will hold the max x bb
- `\l__spath_maxy_dim` will hold the max y bb

```
\tl_new:N \l__spath_rp_tl
\int_new:N \l__spath_l_int
\int_new:N \l__spath_rl_int
\int_new:N \l__spath_nc_int
\tl_new:N \l__spath_ip_tl
\tl_new:N \l__spath_fp_tl
\tl_new:N \l__spath_ia_tl
\tl_new:N \l__spath_fa_tl
\dim_new:N \l__spath_minx_dim
\dim_new:N \l__spath_miny_dim
\dim_new:N \l__spath_maxx_dim
\dim_new:N \l__spath_maxy_dim

\cs_new_nopar:Npn \spath_generate_all:n #1
{
  \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl

  \tl_clear:N \l__spath_rp_tl
  \int_set:Nn \l__spath_l_int {1}
  \int_zero:N \l__spath_rl_int
  \int_set:Nn \l__spath_nc_int {1}
  \tl_clear:N \l__spath_ip_tl
  \tl_clear:N \l__spath_fp_tl
  \tl_clear:N \l__spath_ia_tl
  \tl_clear:N \l__spath_fa_tl
  \dim_zero:N \l__spath_minx_dim
  \dim_zero:N \l__spath_miny_dim
```

```
\dim_zero:N \l__spath_maxx_dim
\dim_zero:N \l__spath_maxy_dim

\tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
\dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
\tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
\dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
\tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

\tl_clear:N \l__spath_ip_tl
\tl_put_right:Nx \l__spath_ip_tl
{
  {\dim_use:N \l__spath_tmpa_dim}
  {\dim_use:N \l__spath_tmpb_dim}
}

\dim_set_eq:NN \l__spath_minx_dim \l__spath_tmpa_dim
\dim_set_eq:NN \l__spath_miny_dim \l__spath_tmpb_dim
\dim_set_eq:NN \l__spath_maxx_dim \l__spath_tmpa_dim
\dim_set_eq:NN \l__spath_maxy_dim \l__spath_tmpb_dim

\tl_put_left:Nx \l__spath_rp_tl
{
  {\dim_use:N \l__spath_tmpa_dim}
  {\dim_use:N \l__spath_tmpb_dim}
}

\tl_set:Nx \l__spath_ia_tl {\tl_head:N \l__spath_tmpa_tl}

\bool_until_do:nn {
  \tl_if_empty_p:N \l__spath_tmpa_tl
}
{
  \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
  \tl_set:Nn \l__spath_tmpd_tl {}
  \tl_set_eq:NN \l__spath_fa_tl \l__spath_tmpc_tl
  \int_incr:N \l__spath_l_int

  \tl_case:Nnn \l__spath_tmpc_tl
  {
    \g__spath_moveto_tl {
      \tl_set_eq:NN \l__spath_tmpd_tl \g__spath_moveto_tl
      \int_incr:N \l__spath_nc_int
    }
    \g__spath_lineto_tl {
      \tl_set_eq:NN \l__spath_tmpd_tl \g__spath_lineto_tl
      \int_incr:N \l__spath_rl_int
    }
    \g__spath_curveto_tl {
      \tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curvetoa_tl
```

```
        \int_incr:N \l__spath_rl_int
      }
      \g__spath_curvetoa_tl {
        \tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curveto_tl
      }
      \g__spath_curvetob_tl {
        \tl_set_eq:NN \l__spath_tmpd_tl \g__spath_curvetob_tl
      }
    }
    {
      \tl_show:N \l__spath_tmpc_tl
    }
    \tl_put_left:NV \l__spath_rp_tl \l__spath_tmpd_tl
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

    \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
    \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

    \dim_set:Nn \l__spath_minx_dim {\dim_min:nn { \l__spath_minx_dim} {\l__spath_tmpa_dim}}
    \dim_set:Nn \l__spath_miny_dim {\dim_min:nn { \l__spath_miny_dim} {\l__spath_tmpb_dim}}
    \dim_set:Nn \l__spath_maxx_dim {\dim_max:nn { \l__spath_maxx_dim} {\l__spath_tmpa_dim}}
    \dim_set:Nn \l__spath_maxy_dim {\dim_max:nn { \l__spath_maxy_dim} {\l__spath_tmpb_dim}}

    \tl_put_left:Nx \l__spath_rp_tl
    {
      {\dim_use:N \l__spath_tmpa_dim}
      {\dim_use:N \l__spath_tmpb_dim}
    }

    \tl_set:Nx \l__spath_fp_tl
    {
      {\dim_use:N \l__spath_tmpa_dim}
      {\dim_use:N \l__spath_tmpb_dim}
    }

  }

  \tl_put_left:NV \l__spath_rp_tl \g__spath_moveto_tl

  \spath_put:nnV {#1} {reverse path} \l__spath_rp_tl
  \spath_put:nnV {#1} {length} \l__spath_l_int
  \spath_put:nnV {#1} {real length} \l__spath_rl_int
  \spath_put:nnV {#1} {number of components} \l__spath_nc_int
  \spath_put:nnV {#1} {initial point} \l__spath_ip_tl
  \spath_put:nnV {#1} {final point} \l__spath_fp_tl
  \spath_put:nnV {#1} {initial action} \l__spath_ia_tl
  \spath_put:nnV {#1} {final action} \l__spath_fa_tl

```

```
  \tl_clear:N \l__spath_tmpb_tl
  \tl_put_right:Nx \l__spath_tmpb_tl
  {
    {\dim_use:N \l__spath_minx_dim}
    {\dim_use:N \l__spath_miny_dim}
  }
  \spath_put:nnV {#1} {min bb} \l__spath_tmpb_tl

  \tl_clear:N \l__spath_tmpb_tl
  \tl_put_right:Nx \l__spath_tmpb_tl
  {
    {\dim_use:N \l__spath_maxx_dim}
    {\dim_use:N \l__spath_maxy_dim}
  }
  \spath_put:nnV {#1} {max bb} \l__spath_tmpb_tl

}
```

2.4 Path Manipulation

`\spath_translate:nnn` Translates a path.

```
\cs_new_nopar:Npn \spath_translate:nnn #1#2#3
{
  \tl_map_inline:Nn \g__spath_moveable_attributes
  {
    \spath_if_in:nnT {#1} {##1}
    {
      \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl

      \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl + #2}
      \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
      \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl + #3}
      \tl_clear:N \l__spath_tmpb_tl
      \tl_put_right:Nx \l__spath_tmpb_tl
      {
        {\dim_use:N \l__spath_tmpa_dim}
        {\dim_use:N \l__spath_tmpb_dim}
      }
      \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
    }
  }
  \tl_map_inline:Nn \g__spath_path_attributes
  {
    \spath_if_in:nnT {#1} {##1}
    {
      \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
      \tl_clear:N \l__spath_tmpb_tl
      \bool_until_do:nn {
        \tl_if_empty_p:N \l__spath_tmpa_tl
```

```
      }
      {
        \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
        \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

        \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl + #2}
        \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

        \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl + #3}
        \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

        \tl_put_right:Nx \l__spath_tmpb_tl
        {
          {\dim_use:N \l__spath_tmpa_dim}
          {\dim_use:N \l__spath_tmpb_dim}
        }
      }
      \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
    }
  }
}

\cs_generate_variant:Nn \spath_translate:nnn {nxx}
```

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

```
\cs_new_nopar:Npn \spath_translate:nn #1#2
{
  \spath_translate:nnn {#1} #2
}

\cs_generate_variant:Nn \spath_translate:nn {nV}
```

`\spath_scale:nnn` Scale a path.

```
\cs_new_nopar:Npn \spath_scale:nnn #1#2#3
{
  \tl_map_inline:Nn \g__spath_moveable_attributes
  {
    \spath_if_in:nnT {#1} {##1}
    {
      \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl

      \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl * #2}
      \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
      \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl * #3}
      \tl_clear:N \l__spath_tmpb_tl
      \tl_put_right:Nx \l__spath_tmpb_tl
      {
        {\dim_use:N \l__spath_tmpa_dim}
        {\dim_use:N \l__spath_tmpb_dim}
```

```
      }
      \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
    }
  }
  \tl_map_inline:Nn \g__spath_path_attributes
  {
    \spath_if_in:nnT {#1} {##1}
    {
      \__spath_get:nnN {#1} {##1} \l__spath_tmpa_tl
      \tl_clear:N \l__spath_tmpb_tl
      \bool_until_do:nn {
        \tl_if_empty_p:N \l__spath_tmpa_tl
      }
      {
        \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
        \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

        \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl * #2}
        \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

        \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl * #3}
        \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

        \tl_put_right:Nx \l__spath_tmpb_tl
        {
          {\dim_use:N \l__spath_tmpa_dim}
          {\dim_use:N \l__spath_tmpb_dim}
        }
      }
      \spath_put:nnV {#1} {##1} \l__spath_tmpb_tl
    }
  }
}
```

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

```
\cs_new_nopar:Npn \spath_reverse:n #1
{
  \spath_if_in:nnF {#1} {reverse path} {
    \cs_use:c {spath_generate_reverse path}
  }
  \spath_swap:nnn {#1} {path} {reverse path}
  \spath_swap:nnn {#1} {initial point} {final point}
  \spath_swap:nnn {#1} {initial action} {final action}
}
```

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

```
\cs_new_nopar:Npn \spath_swap:nnn #1#2#3
{
```

```
  \__spath_get:nnNF {#1} {#2} \l__spath_tmpa_tl {\tl_clear:N \l__spath_tmpa_tl}
  \__spath_get:nnNF {#1} {#3} \l__spath_tmpb_tl {\tl_clear:N \l__spath_tmpb_tl}
  \tl_if_empty:NTF \l__spath_tmpb_tl
  {\spath_remove:nnV {#1} {#2}}
  {\spath_put:nnV {#1} {#2} \l__spath_tmpb_tl}
  \tl_if_empty:NTF \l__spath_tmpa_tl
  {\spath_remove:nnV {#1} {#3}}
  {\spath_put:nnV {#1} {#3} \l__spath_tmpa_tl}
}
```

`\spath_weld:nn` This welds one path to another, moving the second so that it's initial point coincides with the first's final point. It is called a *weld* because the initial move of the second path is removed. The first path is updated, the second is not modified.

```
\cs_new_nopar:Npn \spath_weld:nn #1#2
{
  \spath_clone:nn {#2} {tmp_path}
  \spath_get:nnN {#1} {final point} \l__spath_tmpa_tl

  \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}

  \spath_get:nnN {#2} {initial point} \l__spath_tmpa_tl

  \dim_sub:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \dim_sub:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}

  \spath_translate:nxx {tmp_path} {\dim_use:N \l__spath_tmpa_dim} {\dim_use:N \l__spath_tmpb_d

  \__spath_get:nnN {#1} {path} \l__spath_tmpa_tl
  \__spath_get:nnN {tmp_path} {path} \l__spath_tmpb_tl
  \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
  \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
  \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
  \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl

  \spath_put:nnV {#1} {path} \l__spath_tmpa_tl

  \__spath_get:nnNTF {tmp_path} {final point} \l__spath_tmpa_tl
  {
    \spath_put:nnV {#1} {final point} \l__spath_tmpa_tl
  }
  {
    \spath_remove:nn {#1} {final point}
  }

  \__spath_get:nnNTF {tmp_path} {final action} \l__spath_tmpa_tl
  {
```

```
    \spath_put:nnV {#1} {final action} \l__spath_tmpa_tl
  }
  {
    \spath_remove:nn {#1} {final action}
  }

  \__spath_get:nnNT {tmp_path} {min bb} \l__spath_tmpa_tl
  {
    \__spath_get:nnNT {#1} {min bb} \l__spath_tmpb_tl
    {
      \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
      \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
      \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}

      \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\l__spath_tmpa_dim} {\tl_head:N
      \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
      \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\l__spath_tmpb_dim} {\tl_head:N \l__spath_t

      \tl_clear:N \l__spath_tmpb_tl
      \tl_put_right:Nx \l__spath_tmpb_tl
      {
        {\dim_use:N \l__spath_tmpa_dim}
        {\dim_use:N \l__spath_tmpb_dim}
      }
      \spath_put:nnV {#1} {min bb} \l__spath_tmpb_tl
    }
  }

  \__spath_get:nnNT {tmp_path} {max bb} \l__spath_tmpa_tl
  {
    \__spath_get:nnNT {#1} {max bb} \l__spath_tmpb_tl
    {
      \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
      \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
      \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}

      \dim_set:Nn \l__spath_tmpa_dim {\dim_max:nn {\l__spath_tmpa_dim} {\tl_head:N
      \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
      \dim_set:Nn \l__spath_tmpb_dim {\dim_max:nn {\l__spath_tmpb_dim} {\tl_head:N \l__spath_t

      \tl_clear:N \l__spath_tmpb_tl
      \tl_put_right:Nx \l__spath_tmpb_tl
      {
        {\dim_use:N \l__spath_tmpa_dim}
        {\dim_use:N \l__spath_tmpb_dim}
      }
      \spath_put:nnV {#1} {max bb} \l__spath_tmpb_tl
    }
  }

```

```
  \__spath_get:nnNT {tmp_path} {reverse path} \l__spath_tmpa_tl
  {
    \__spath_get:nnNT {#1} {reverse path} \l__spath_tmpb_tl
    {
      \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
      \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
      \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
      \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl

      \spath_put:nnV {#1} {reverse path} \l__spath_tmpa_tl
    }
  }

  \__spath_get:nnNT {tmp_path} {length} \l__spath_tmpa_tl
  {
    \__spath_get:nnNT {#1} {length} \l__spath_tmpb_tl
    {
      \int_set:Nn \l__spath_tmpa_int {\l__spath_tmpa_tl + \l__spath_tmpb_tl - 1}
      \spath_put:nnV {#1} {length} \l__spath_tmpa_int
    }
  }

  \__spath_get:nnNT {tmp_path} {real length} \l__spath_tmpa_tl
  {
    \__spath_get:nnNT {#1} {real length} \l__spath_tmpb_tl
    {
      \int_set:Nn \l__spath_tmpa_int {\l__spath_tmpa_tl + \l__spath_tmpb_tl}
      \spath_put:nnV {#1} {real length} \l__spath_tmpa_int
    }
  }

  \__spath_get:nnNT {tmp_path} {number of components} \l__spath_tmpa_tl
  {
    \__spath_get:nnNT {#1} {number of components} \l__spath_tmpb_tl
    {
      \int_set:Nn \l__spath_tmpa_int {\l__spath_tmpa_tl + \l__spath_tmpb_tl - 1}
      \spath_put:nnV {#1} {number of components} \l__spath_tmpa_int
    }
  }

}
```

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

```
\cs_new_nopar:Npn \spath_prepend_no_move:nn #1#2
{
  \spath_if_exist:nT {#2}
  {
    \__spath_get:nnN {#2} {path} \l__spath_tmpa_tl
    \__spath_get:nnN {#1} {path} \l__spath_tmpb_tl
    \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
```

```
\tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
\tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
\tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
\spath_put:nnV {#1} {path} \l__spath_tmpa_tl

\spath_if_in:nnTF {#2} {initial point}
{
  \__spath_get:nnN {#2} {initial point} \l__spath_tmpa_tl
  \spath_put:nnV {#1} {initial point} \l__spath_tmpa_tl
}
{
  \spath_remove:nn {#1} {initial point}
}

\spath_if_in:nnTF {#2} {initial action}
{
  \__spath_get:nnN {#2} {initial action} \l__spath_tmpa_tl
  \spath_put:nnV {#1} {initial action} \l__spath_tmpa_tl
}
{
  \spath_remove:nn {#1} {initial action}
}

\bool_if:nTF
{
  \spath_if_in_p:nn {#1} {length}
  &&
  \spath_if_in_p:nn {#2} {length}
}
{
  \__spath_get:nnN {#1} {length} \l__spath_tmpa_tl
  \__spath_get:nnN {#2} {length} \l__spath_tmpb_tl
  \spath_put:nnx {#1} {length} {\int_eval:n {\l__spath_tmpa_tl +
      \l__spath_tmpb_tl - 1}}
}
{
  \spath_remove:nn {#1} {length}
}
\bool_if:nTF
{
  \spath_if_in_p:nn {#1} {real length}
  &&
  \spath_if_in_p:nn {#2} {real length}
}
{
  \__spath_get:nnN {#1} {real length} \l__spath_tmpa_tl
  \__spath_get:nnN {#2} {real length} \l__spath_tmpb_tl
  \spath_put:nnx {#1} {real length} {\int_eval:n {\l__spath_tmpa_tl +
      \l__spath_tmpb_tl }}
}
```

```
    {
      \spath_remove:nn {#1} {real length}
    }
    \bool_if:nTF
    {
      \spath_if_in_p:nn {#1} {number of components}
      &&
      \spath_if_in_p:nn {#2} {number of components}
    }
    {
      \__spath_get:nnN {#1} {number of components} \l__spath_tmpa_tl
      \__spath_get:nnN {#2} {number of components} \l__spath_tmpb_tl
      \spath_put:nnx {#1} {number of components} {\int_eval:n {\l__spath_tmpa_tl +
          \l__spath_tmpb_tl - 1}}
    }
    {
      \spath_remove:nn {#1} {number of components}
    }
    \bool_if:nTF
    {
      \spath_if_in_p:nn {#1} {min bb}
      &&
      \spath_if_in_p:nn {#2} {min bb}
    }
    {
      \__spath_get:nnN {#1} {min bb} \l__spath_tmpa_tl
      \__spath_get:nnN {#2} {min bb} \l__spath_tmpb_tl
      \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_item:Nn
          \l__spath_tmpa_tl {1}} {\tl_item:Nn
          \l__spath_tmpb_tl {1}}}
      \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_item:Nn
          \l__spath_tmpa_tl {2}} {\tl_item:Nn
          \l__spath_tmpb_tl {2}}}
      \spath_put:nnx {#1} {min bb} {
        {\dim_use:N \l__spath_tmpa_dim}
        {\dim_use:N \l__spath_tmpb_dim}
      }
    }
    {
      \spath_remove:nn {#1} {min bb}
    }
    \bool_if:nTF
    {
      \spath_if_in_p:nn {#1} {max bb}
      &&
      \spath_if_in_p:nn {#2} {max bb}
    }
    {
      \__spath_get:nnN {#1} {max bb} \l__spath_tmpa_tl
      \__spath_get:nnN {#2} {max bb} \l__spath_tmpb_tl
```

```
      \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_item:Nn
          \l__spath_tmpa_tl {1}} {\tl_item:Nn
          \l__spath_tmpb_tl {1}}}
      \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_item:Nn
          \l__spath_tmpa_tl {2}} {\tl_item:Nn
          \l__spath_tmpb_tl {2}}}
      \spath_put:nnx {#1} {max bb} {
        {\dim_use:N \l__spath_tmpa_dim}
        {\dim_use:N \l__spath_tmpb_dim}
      }
    }
    {
      \spath_remove:nn {#1} {max bb}
    }
    \bool_if:nTF
    {
      \spath_if_in_p:nn {#1} {reverse path}
      &&
      \spath_if_in_p:nn {#2} {reverse path}
    }
    {
      \__spath_get:nnN {#1} {reverse path} \l__spath_tmpa_tl
      \__spath_get:nnN {#2} {reverse path} \l__spath_tmpb_tl
      \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
      \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
      \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
      \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
      \spath_put:nnV {#1} {reverse path} \l__spath_tmpb_tl
    }
    {
      \spath_remove:nn {#1} {reverse path}
    }

  }
}
```

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

```
\cs_new_nopar:Npn \spath_append_no_move:nn #1#2
{
  \spath_if_exist:nT {#2}
  {
    \spath_get:nnN {#1} {path} \l__spath_tmpa_tl
    \spath_get:nnN {#2} {path} \l__spath_tmpb_tl
    \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
    \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
    \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
    \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
    \spath_put:nnV {#1} {path} \l__spath_tmpa_tl
    \spath_if_in:nnTF {#2} {final point}
    {
```

```
    \__spath_get:nnN {#2} {final point} \l__spath_tmpa_tl
    \spath_put:nnV {#1} {final point} \l__spath_tmpa_tl
  }
  {
    \spath_remove:nn {#1} {final point}
  }
  \spath_if_in:nnTF {#2} {final action}
  {
    \__spath_get:nnN {#2} {final action} \l__spath_tmpa_tl
    \spath_put:nnV {#1} {final action} \l__spath_tmpa_tl
  }
  {
    \spath_remove:nn {#1} {final action}
  }
  \bool_if:nTF
  {
    \spath_if_in_p:nn {#1} {length}
    &&
    \spath_if_in_p:nn {#2} {length}
  }
  {
    \__spath_get:nnN {#1} {length} \l__spath_tmpa_tl
    \__spath_get:nnN {#2} {length} \l__spath_tmpb_tl
    \spath_put:nnx {#1} {length} {\int_eval:n {\l__spath_tmpa_tl +
        \l__spath_tmpb_tl - 1}}
  }
  {
    \spath_remove:nn {#1} {length}
  }
  \bool_if:nTF
  {
    \spath_if_in_p:nn {#1} {real length}
    &&
    \spath_if_in_p:nn {#2} {real length}
  }
  {
    \__spath_get:nnN {#1} {real length} \l__spath_tmpa_tl
    \__spath_get:nnN {#2} {real length} \l__spath_tmpb_tl
    \spath_put:nnx {#1} {real length} {\int_eval:n {\l__spath_tmpa_tl +
        \l__spath_tmpb_tl }}
  }
  {
    \spath_remove:nn {#1} {real length}
  }
  \bool_if:nTF
  {
    \spath_if_in_p:nn {#1} {number of components}
    &&
    \spath_if_in_p:nn {#2} {number of components}
  }
```

```
    {
      \__spath_get:nnN {#1} {number of components} \l__spath_tmpa_tl
      \__spath_get:nnN {#2} {number of components} \l__spath_tmpb_tl
      \spath_put:nnx {#1} {number of components} {\int_eval:n {\l__spath_tmpa_tl +
          \l__spath_tmpb_tl - 1}}
    }
    {
      \spath_remove:nn {#1} {number of components}
    }
    \bool_if:nTF
    {
      \spath_if_in_p:nn {#1} {min bb}
      &&
      \spath_if_in_p:nn {#2} {min bb}
    }
    {
      \__spath_get:nnN {#1} {min bb} \l__spath_tmpa_tl
      \__spath_get:nnN {#2} {min bb} \l__spath_tmpb_tl
      \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_item:Nn
          \l__spath_tmpa_tl {1}} {\tl_item:Nn
          \l__spath_tmpb_tl {1}}}
      \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_item:Nn
          \l__spath_tmpa_tl {2}} {\tl_item:Nn
          \l__spath_tmpb_tl {2}}}
      \spath_put:nnx {#1} {min bb} {
        {\dim_use:N \l__spath_tmpa_dim}
        {\dim_use:N \l__spath_tmpb_dim}
      }
    }
    {
      \spath_remove:nn {#1} {min bb}
    }
    \bool_if:nTF
    {
      \spath_if_in_p:nn {#1} {max bb}
      &&
      \spath_if_in_p:nn {#2} {max bb}
    }
    {
      \__spath_get:nnN {#1} {max bb} \l__spath_tmpa_tl
      \__spath_get:nnN {#2} {max bb} \l__spath_tmpb_tl
      \dim_set:Nn \l__spath_tmpa_dim {\dim_min:nn {\tl_item:Nn
          \l__spath_tmpa_tl {1}} {\tl_item:Nn
          \l__spath_tmpb_tl {1}}}
      \dim_set:Nn \l__spath_tmpb_dim {\dim_min:nn {\tl_item:Nn
          \l__spath_tmpa_tl {2}} {\tl_item:Nn
          \l__spath_tmpb_tl {2}}}
      \spath_put:nnx {#1} {max bb} {
        {\dim_use:N \l__spath_tmpa_dim}
        {\dim_use:N \l__spath_tmpb_dim}
```

```
      }
    }
    {
      \spath_remove:nn {#1} {max bb}
    }
    \bool_if:nTF
    {
      \spath_if_in_p:nn {#1} {reverse path}
      &&
      \spath_if_in_p:nn {#2} {reverse path}
    }
    {
      \__spath_get:nnN {#2} {reverse path} \l__spath_tmpa_tl
      \__spath_get:nnN {#1} {reverse path} \l__spath_tmpb_tl
      \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
      \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
      \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
      \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
      \spath_put:nnV {#1} {reverse path} \l__spath_tmpb_tl
    }
    {
      \spath_remove:nn {#1} {reverse path}
    }
  }
}
```

2.5 Iteration Functions

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

```
\cs_new_nopar:Npn \spath_map_component:Nn #1#2
{
  \int_gincr:N \g__prg_map_int
  \cs_gset:cpn { __prg_map_ \int_use:N \g__prg_map_int :w } ##1 {#2}
  \tl_set:NV \l__spath_tmpa_tl #1
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  \tl_put_right:NV \l__spath_tmpa_tl \g__spath_moveto_tl
  \tl_set_eq:NN \l__spath_tmpb_tl \g__spath_moveto_tl
  \bool_until_do:nn {
    \tl_if_empty_p:N \l__spath_tmpa_tl
  }
  {
    \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
    \tl_if_eq:NNT \l__spath_tmpc_tl \g__spath_moveto_tl
    {
      \exp_args:NnV \use:c { __prg_map_ \int_use:N \g__prg_map_int :w } \l__spath_tmpb_tl
\tl_clear:N \l__spath_tmpb_tl
    }
    \tl_if_single:NTF \l__spath_tmpc_tl
    {
```

```
      \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpc_tl
    }
    {
      \tl_put_right:Nx \l__spath_tmpb_tl {{\l__spath_tmpc_tl}}
    }
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
  }
}
```

\spath_map_segment_inline:Nn

This iterates through the segments of the path, applying the inline function to each.

```
\cs_new_nopar:Npn \spath_map_segment_inline:Nn #1#2
{
  \int_gincr:N \g__prg_map_int
  \cs_gset:cpn { __prg_map_ \int_use:N \g__prg_map_int :w } ##1 ##2 {#2}
  \spath_map_segment_function:Nc #1 { __prg_map_ \int_use:N \g__prg_map_int :w }
}
```

\spath_map_segment_inline:nn

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

```
\cs_new_nopar:Npn \spath_map_segment_inline:nn #1#2
{
  \int_gincr:N \g__prg_map_int
  \cs_gset:cpn { __prg_map_ \int_use:N \g__prg_map_int :w } ##1 ##2 {#2}
  \spath_get:nnN {#1} {path} \l__spath_tmpd_tl
  \spath_map_segment_function:Nc \l__spath_tmpd_tl { __prg_map_ \int_use:N \g__prg_map_int :w
}
```

\spath_map_segment_function:nN

This iterates through the segments of the path of the `spath` object, applying the specified function to each. The specified function should take two `N` type arguments. The first is a token representing the type of path segment, the second is the path segment itself.

```
\cs_new_nopar:Npn \spath_map_segment_function:nN #1#2
{
  \spath_get:nnN {#1} {path} \l__spath_tmpd_tl
  \spath_map_segment_function:NN \l__spath_tmpd_tl #2
}

\cs_new_nopar:Npn \spath_map_segment_function:NN #1#2
{
  \tl_set_eq:NN \l__spath_tmpa_tl #1
  \tl_clear:N \l__spath_tmpb_tl
  \dim_zero:N \l__spath_tmpa_dim
  \dim_zero:N \l__spath_tmpb_dim

  \bool_until_do:nn {
    \tl_if_empty_p:N \l__spath_tmpa_tl
  }
  {
    \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
    \tl_case:Nnn \l__spath_tmpc_tl
```

```
{
  \g__spath_lineto_tl
  {
    \tl_set_eq:NN \l__spath_tmpb_tl \g__spath_moveto_tl
    \tl_put_right:Nx \l__spath_tmpb_tl
    {
      {\dim_use:N \l__spath_tmpa_dim}
      {\dim_use:N \l__spath_tmpb_dim}
    }
    \tl_put_right:NV \l__spath_tmpb_tl \g__spath_lineto_tl

    \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
    \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

    \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
    \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

  }

  \g__spath_curvetoa_tl
  {
    \tl_set_eq:NN \l__spath_tmpb_tl \g__spath_moveto_tl
    \tl_put_right:Nx \l__spath_tmpb_tl
    {
      {\dim_use:N \l__spath_tmpa_dim}
      {\dim_use:N \l__spath_tmpb_dim}
    }
    \tl_put_right:NV \l__spath_tmpb_tl \g__spath_curvetoa_tl

    \prg_replicate:nn {2} {
      \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
      \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
      \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N               \l__spath_tmpa_tl}}
      \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
      \tl_put_right:Nx \l__spath_tmpb_tl {\tl_head:N             \l__spath_tmpa_tl}
      \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
    }

    \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
    \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

    \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
    \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

  }
}
```

```
    {

      \tl_set_eq:NN \l__spath_tmpb_tl \l__spath_tmpc_tl
      \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
      \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpa_tl}
      \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

      \tl_put_right:Nx \l__spath_tmpb_tl {{\tl_head:N \l__spath_tmpa_tl}}
      \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpa_tl}
      \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

    }

    #2 \l__spath_tmpc_tl \l__spath_tmpb_tl
    \tl_clear:N \l__spath_tmpb_tl

  }
}
\cs_generate_variant:Nn \spath_map_segment_function:NN {Nc}
```

2.6 Public Commands

The next functions are more "public" than the previous lot. That said, they aren't intended for direct use in a normal document.

Most are just wrappers around internal functions.

`\MakeSPath` Constructs an `spath` object out of the given name and path.

```
\NewDocumentCommand \MakeSPath { m m }
{
  \spath_clear_new:n {#1}
  \spath_put:nno {#1} {path} {#2}
}
```

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

```
\NewDocumentCommand \MakeSPathList { m m }
{
  \tl_gclear_new:c {l__spath_list_#1}
  \int_zero:N \l__spath_tmpa_int
  \spath_map_component:Nn #2 {
    \spath_clear_new:n {#1 _ \int_use:N \l__spath_tmpa_int}
    \spath_put:nnn  {#1 _ \int_use:N \l__spath_tmpa_int} {path} {##1}
    \tl_gput_right:cx {l__spath_list_#1} {{#1 _ \int_use:N \l__spath_tmpa_int}}
    \int_incr:N \l__spath_tmpa_int
  }
}
```

`\CloneSPath`

```
\NewDocumentCommand \CloneSPath { m m }
{
```

```
  \spath_clone:nn {#1} {#2}
}
```

\SPathInfo

```
\NewDocumentCommand \SPathInfo { m m }
{
  \spath_get:nn {#1} {#2}
}
```

\SPathPrepare

```
\NewDocumentCommand \SPathPrepare { m }
{
  \spath_generate_all:n {#1}
}
```

\SPathListPrepare

```
\NewDocumentCommand \SPathListPrepare { m }
{
  \tl_map_inline:cn {l__spath_list_#1}
  {
    \spath_generate_all:n {##1}
  }
}
```

\SPathInfoInto

```
\NewDocumentCommand \SPathInfoInto { m m m }
{
  \tl_set:Nn \l_tmpa_tl #3
  \spath_get:nnV {#1} {#2} \l_tmpa_tl
}
```

\SPathShow

```
\NewDocumentCommand \SPathShow { m m }
{
  \spath_show:n {#1}
}
```

\SPathTranslate

```
\NewDocumentCommand \SPathTranslate { m m m }
{
  \spath_translate:nnn {#1} {#2} {#3}
}
```

\SPathTranslateInto Clones the path before translating it.

```
\NewDocumentCommand \SPathTranslateInto { m m m m }
{
  \spath_clone:nn {#1} {#2}
  \spath_translate:nnn {#2} {#3} {#4}
}
```

\SPathScale

```
\NewDocumentCommand \SPathScale { m m m }
{
  \spath_translate:nnn {#1} {#2} {#3}
}
```

\SPathScaleInto Clones the path first.

```
\NewDocumentCommand \SPathScaleInto { m m m m }
{
  \spath_clone:nn {#1} {#2}
  \spath_translate:nnn {#2} {#3} {#4}
}
```

\SPathWeld

```
\NewDocumentCommand \SPathWeld { m m }
{
  \spath_weld:nn {#1} {#2}
}
```

\SPathWeldInto

```
\NewDocumentCommand \SPathWeldInto { m m m }
{
  \spath_clone:nn {#1} {#2}
  \spath_weld:nn {#2} {#3}
}
```

Interfaces via TikZ keys.

```
\tikzset{
  save~spath/.code={
    \tikz@addmode{
      \spath_get_current_path:n {#1}
    }
  },
  restore~spath/.code={
    \spath_set_current_path:n {#1}
  },
}
```

2.7 Miscellaneous Commands

\spath_split_curve:nnNN Splits a Bezier cubic into pieces.

```
\cs_new_nopar:Npn \spath_split_curve:nnNN #1#2#3#4
{
  \group_begin:
  \tl_gclear:N \l__spath_smuggle_tl
  \tl_set_eq:NN \l__spath_tmpa_tl \g__spath_moveto_tl
  \tl_put_right:Nx \l__spath_tmpa_tl {
    {\tl_item:nn {#2} {3}}
    {\tl_item:nn {#2} {4}}
```

```
}
\tl_put_right:NV \l__spath_tmpa_tl \g__spath_curvetoa_tl
\tl_put_right:Nx \l__spath_tmpa_tl
{
  {\fp_to_dim:n
  {
    (1 - #1) * \tl_item:nn {#2} {2} + (#1) * \tl_item:nn {#2} {5}
  }}
  {\fp_to_dim:n
  {
    (1 - #1) * \tl_item:nn {#2} {3} + (#1) * \tl_item:nn {#2} {6}
  }}
}
\tl_put_right:NV \l__spath_tmpa_tl \g__spath_curvetob_tl
\tl_put_right:Nx \l__spath_tmpa_tl
{
  {\fp_to_dim:n
  {
    (1 - #1)^2 * \tl_item:nn {#2} {2} + 2 * (1 - #1) * (#1) * \tl_item:nn {#2} {5} + (#1)^2
  }}
  {\fp_to_dim:n
  {
    (1 - #1)^2 * \tl_item:nn {#2} {3} + 2 * (1 - #1) * (#1) * \tl_item:nn {#2} {6} + (#1)^2
  }}
}
\tl_put_right:NV \l__spath_tmpa_tl \g__spath_curveto_tl
\tl_put_right:Nx \l__spath_tmpa_tl
{
  {\fp_to_dim:n
    {
    (1 - #1)^3 * \tl_item:nn {#2} {2} + 3 * (1 - #1)^2 * (#1) * \tl_item:nn {#2} {5} + 3 * (
  }}
  {\fp_to_dim:n
  {
    (1 - #1)^3 * \tl_item:nn {#2} {3} + 3 * (1 - #1)^2 * (#1) * \tl_item:nn {#2} {6} + 3 * (
  }}
}
\tl_gset_eq:NN \l__spath_smuggle_tl \l__spath_tmpa_tl
\group_end:
\tl_set_eq:NN #3 \l__spath_smuggle_tl
\group_begin:
\tl_set_eq:NN \l__spath_tmpa_tl \g__spath_moveto_tl
\tl_put_right:Nx \l__spath_tmpa_tl
{
  {\fp_to_dim:n
    {
    (1 - #1)^3 * \tl_item:nn {#2} {2} + 3 * (1 - #1)^2 * (#1) * \tl_item:nn {#2} {5} + 3 * (
  }}
  {\fp_to_dim:n
  {
```

```
      (1 - #1)^3 * \tl_item:nn {#2} {3} + 3 * (1 - #1)^2 * (#1) * \tl_item:nn {#2} {6} + 3 * (
    }}
  }
  \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curvetoa_tl
  \tl_put_right:Nx \l__spath_tmpa_tl
  {
    {\fp_to_dim:n
    {
      (1 - #1)^2 * \tl_item:nn {#2} {5} + 2 * (1 - #1) * (#1) * \tl_item:nn {#2} {8} + (#1)^2
    }}
    {\fp_to_dim:n
    {
      (1 - #1)^2 * \tl_item:nn {#2} {6} + 2 * (1 - #1) * (#1) * \tl_item:nn {#2} {9} + (#1)^2
    }}
  }
  \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curvetob_tl
  \tl_put_right:Nx \l__spath_tmpa_tl
  {
    {\fp_to_dim:n
    {
      (1 - #1) * \tl_item:nn {#2} {8} + (#1) * \tl_item:nn {#2} {11}
    }}
    {\fp_to_dim:n
    {
      (1 - #1) * \tl_item:nn {#2} {9} + (#1) * \tl_item:nn {#2} {12}
    }}
  }
  \tl_put_right:NV \l__spath_tmpa_tl \g__spath_curveto_tl
  \tl_put_right:Nx \l__spath_tmpa_tl {
    {\tl_item:nn {#2} {11}}
    {\tl_item:nn {#2} {12}}
  }
  \tl_gset_eq:NN \l__spath_smuggle_tl \l__spath_tmpa_tl
  \group_end:
  \tl_set_eq:NN #4 \l__spath_smuggle_tl
}

\cs_generate_variant:Nn \spath_split_curve:nnNN {nVNN, VVNN}
```

3 The Calligraphy Package

3.1 Initialisation

```
\RequirePackage{spath3}
\ExplSyntaxOn

\tl_new:N \l__cal_tmpa_tl
\tl_new:N \l__cal_tmpb_tl
\int_new:N \l__cal_tmpa_int
```

```
\int_new:N \l__cal_tmpb_int
\int_new:N \l__cal_path_component_int
\dim_new:N \l__cal_tmpa_dim
\dim_new:N \l__cal_tmpb_dim
\dim_new:N \l__cal_tmpc_dim
\dim_new:N \l__cal_tmpd_dim
\dim_new:N \l__cal_tmpe_dim
\dim_new:N \l__cal_tmpf_dim
\dim_new:N \l__cal_tmpg_dim
\dim_new:N \l__cal_tmph_dim
\bool_new:N \l__cal_taper_start_bool
\bool_new:N \l__cal_taper_end_bool
\bool_new:N \l__cal_taperable_bool
\dim_new:N \l__cal_taper_width_dim
\dim_new:N \l__cal_line_width_dim

\bool_set_true:N \l__cal_taper_start_bool
\bool_set_true:N \l__cal_taper_end_bool
```

3.2 TikZ Keys

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

```
\tikzset{
  define~pen/.code={
    \tikzset{pen~name=#1}
    \pgf@relevantforpicturesizefalse
    \tikz@addmode{
      \pgfsyssoftpath@getcurrentpath\l__cal_tmpa_tl
      \MakeSPathList{calligraphy pen \pgfkeysvalueof{/tikz/pen~name}}{\l__cal_tmpa_tl}
      \SPathListPrepare{calligraphy pen \pgfkeysvalueof{/tikz/pen~name}}
      \pgfusepath{discard}%
    }
  },
  define~pen/.default={default},
  use~pen/.code={
    \tikzset{pen~name=#1}
    \int_gzero:N \l__cal_path_component_int
    \cs_set_eq:NN \pgfpathmoveto \cal_moveto:n
    \tikz@addmode{
      \pgfsyssoftpath@getcurrentpath\l__cal_tmpa_tl
      \MakeSPathList{calligraphy path}{\l__cal_tmpa_tl}
      \SPathListPrepare{calligraphy path}
      \CalligraphyPathCreate{calligraphy path}{\pgfkeysvalueof{/tikz/pen~name}}
    }
  },
  use~pen/.default={default},
  pen~name/.initial={default},
  copperplate/.style={pen~name=copperplate},
  pen~colour/.initial={black},
  weight/.is~choice,
```

```
  weight/heavy/.style={
    line~width=\pgfkeysvalueof{/tikz/heavy~line~width},
    taper~width=\pgfkeysvalueof{/tikz/light~line~width},
  },
  weight/light/.style={
    line~width=\pgfkeysvalueof{/tikz/light~line~width},
    taper~width=0pt,
  },
  heavy/.style={
    weight=heavy
  },
  light/.style={
    weight=light
  },
  heavy~line~width/.initial=2pt,
  light~line~width/.initial=1pt,
  taper/.is~choice,
  taper/.default=both,
  taper/none/.style={
    taper~start=false,
    taper~end=false,
  },
  taper/both/.style={
    taper~start=true,
    taper~end=true,
  },
  taper/start/.style={
    taper~start=true,
    taper~end=false,
  },
  taper/end/.style={
    taper~start=false,
    taper~end=true,
  },
  taper~start/.code={
    \tl_if_eq:nnTF {#1} {true}
    {
      \bool_set_true:N \l__cal_taper_start_bool
    }
    {
      \bool_set_false:N \l__cal_taper_start_bool
    }
  },
  taper~start/.default={true},
  taper~end/.code={
    \tl_if_eq:nnTF {#1} {true}
    {
      \bool_set_true:N \l__cal_taper_end_bool
    }
    {
```

```
      \bool_set_false:N \l__cal_taper_end_bool
    }
  },
  taper~end/.default={true},
  taper~width/.code={\dim_set:Nn \l__cal_taper_width_dim {#1}},
  nib~style/.code~2~args={
    \tl_clear_new:c {l__cal_nib_style_#1}
    \tl_set:cn {l__cal_nib_style_#1} {#2}
  },
  stroke~style/.code~2~args={
    \tl_clear_new:c {l__cal_stroke_style_#1}
    \tl_set:cn {l__cal_stroke_style_#1} {#2}
  },
  this~stroke~style/.code={
    \tl_clear_new:c {l__cal_stroke_inline_style_ \int_use:N \l__cal_path_component_int}
    \tl_set:cn {l__cal_stroke_inline_style_ \int_use:N \l__cal_path_component_int} {#1}
  },
}
```

Some wrappers around the TikZ keys.

```
\NewDocumentCommand \pen { O{} }
{
  \path[define~ pen,#1]
}

\NewDocumentCommand \definepen { O{} }
{
  \tikz \path[define~ pen,#1]
}

\NewDocumentCommand \calligraphy { O{} }
{
  \path[use~ pen,#1]
}
```

3.3 The Path Creation

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

```
\NewDocumentCommand \CalligraphyPathCreate { m m }
{
  \int_zero:N \l__cal_tmpa_int
  \tl_map_inline:cn {l__spath_list_#1}
  {
    \int_incr:N \l__cal_tmpa_int
    \int_zero:N \l__cal_tmpb_int
    \tl_map_inline:cn {l__spath_list_calligraphy pen #2}
    {
      \int_incr:N \l__cal_tmpb_int
      \group_begin:
      \cal_apply_style:c {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_int}
```

```
      \cal_apply_style:c {l__cal_stroke_inline_style_ \int_use:N \l__cal_tmpa_int}
      \cal_apply_style:c {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}

      \spath_clone:nn {##1} {calligraphy temp path}

      \__spath_get:nnN {####1} {initial point} \l__spath_tmpa_tl
      \spath_translate:nV {calligraphy temp path} \l__spath_tmpa_tl

      \__spath_get:nnN {####1} {length} \l__spath_tmpa_tl

      \int_compare:nTF {\l__spath_tmpa_tl = 1}
      {
        \cal_at_least_three:n {calligraphy temp path}

        \spath_protocol_path:n {calligraphy temp path}

        \__spath_get:nnN {calligraphy temp path} {path} \l__spath_tmpa_tl

        \tikz@options
        \dim_set:Nn \l__cal_line_width_dim {\pgflinewidth}
        \cal_maybe_taper:N \l__spath_tmpa_tl
      }
      {

        \spath_weld:nn {calligraphy temp path} {####1}
        \spath_reverse:n {##1}
        \spath_reverse:n {####1}
        \spath_weld:nn {calligraphy temp path} {##1}
        \spath_weld:nn {calligraphy temp path} {####1}
        \spath_reverse:n {##1}
        \spath_reverse:n {####1}

        \spath_set_current_path:n {calligraphy temp path}

        \tikz@options
        \pgfsetfillcolor{\pgfkeysvalueof{/tikz/pen~colour}}
        \pgfusepath{fill}
      }
      \group_end:
    }
  }
}
```

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

```
\cs_new_eq:NN \cal_orig_moveto:n \pgfpathmoveto
\cs_new_nopar:Npn \cal_moveto:n #1
{
  \int_gincr:N \l__cal_path_component_int
  \cal_orig_moveto:n {#1}
```

```
}
```

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

```
\cs_new_nopar:Npn \cal_apply_style:N #1
{
  \tl_if_exist:NT #1 {
    \exp_args:NV \tikzset #1
  }
}
\cs_generate_variant:Nn \cal_apply_style:N {c}
```

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

```
\cs_new_nopar:Npn \cal_at_least_three:n #1
{
  \spath_get:nnN {#1} {real length} \l__cal_tmpa_tl
  \tl_clear:N \l__cal_tmpb_tl
  \int_compare:nTF {\l__cal_tmpa_tl = 1}
  {
    \spath_get:nnN {#1} {path} \l__cal_tmpa_tl
    \spath_map_segment_inline:Nn \l__cal_tmpa_tl
    {
      \tl_case:Nnn ##1 {
        \g__spath_lineto_tl {
          \cal_split_line_in_three:NN \l__cal_tmpb_tl ##2
        }
        \g__spath_curvetoa_tl {
          \cal_split_curve_in_three:NN \l__cal_tmpb_tl ##2
        }
      }
      {
        \tl_put_right:NV \l__cal_tmpb_tl ##2
      }
    }
    \spath_put:nnV {#1} {path} \l__cal_tmpb_tl
  }
  {
    \int_compare:nT {\l__cal_tmpa_tl = 2}
    {
      \spath_get:nnN {#1} {path} \l__cal_tmpa_tl
      \spath_map_segment_inline:Nn \l__cal_tmpa_tl
      {
        \tl_case:Nnn ##1 {
          \g__spath_lineto_tl {
            \cal_split_line_in_two:NN \l__cal_tmpb_tl ##2
          }
          \g__spath_curvetoa_tl {
            \cal_split_curve_in_two:NN \l__cal_tmpb_tl ##2
          }
```

```
        }
        {
          \tl_put_right:NV \l__cal_tmpb_tl ##2
        }
      }
      \spath_put:nnV {#1} {path} \l__cal_tmpb_tl
    }
  }
}
```

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

```
\cs_new_nopar:Npn \cal_split_line_in_two:NN #1#2
{
  \tl_set_eq:NN \l__cal_tmpc_tl #2

  \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}

  \dim_set:Nn \l__cal_tmpa_dim {\tl_head:N \l__cal_tmpc_tl}
  \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}

  \dim_set:Nn \l__cal_tmpb_dim {\tl_head:N \l__cal_tmpc_tl}
  \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}

  \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}

  \dim_set:Nn \l__cal_tmpc_dim {\tl_head:N \l__cal_tmpc_tl}
  \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
  \dim_set:Nn \l__cal_tmpd_dim {\tl_head:N \l__cal_tmpc_tl}
  \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}

  \tl_put_right:NV #1 \g__spath_lineto_tl

  \tl_put_right:Nx #1 {
    {\dim_eval:n {(\l__cal_tmpa_dim + \l__cal_tmpc_dim)/2}}
    {\dim_eval:n {(\l__cal_tmpb_dim + \l__cal_tmpd_dim)/2}}
  }

  \tl_put_right:NV #1 \g__spath_lineto_tl
  \tl_put_right:Nx #1 {
    {\dim_use:N \l__cal_tmpc_dim}
    {\dim_use:N \l__cal_tmpd_dim}
  }
}
```

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

```
\cs_new_nopar:Npn \cal_split_line_in_three:NN #1#2
{
  \tl_set_eq:NN \l__cal_tmpc_tl #2

```

```
  \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}

  \dim_set:Nn \l__cal_tmpa_dim {\tl_head:N \l__cal_tmpc_tl}
  \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}

  \dim_set:Nn \l__cal_tmpb_dim {\tl_head:N \l__cal_tmpc_tl}
  \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}

  \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}

  \dim_set:Nn \l__cal_tmpc_dim {\tl_head:N \l__cal_tmpc_tl}
  \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}
  \dim_set:Nn \l__cal_tmpd_dim {\tl_head:N \l__cal_tmpc_tl}
  \tl_set:Nx \l__cal_tmpc_tl {\tl_tail:N \l__cal_tmpc_tl}

  \tl_put_right:NV #1 \g__spath_lineto_tl

  \tl_put_right:Nx #1 {
    {\dim_eval:n {(2\l__cal_tmpa_dim + \l__cal_tmpc_dim)/3}}
    {\dim_eval:n {(2\l__cal_tmpb_dim + \l__cal_tmpd_dim)/3}}
  }

  \tl_put_right:NV #1 \g__spath_lineto_tl

  \tl_put_right:Nx #1 {
    {\dim_eval:n {(\l__cal_tmpa_dim + 2\l__cal_tmpc_dim)/3}}
    {\dim_eval:n {(\l__cal_tmpb_dim + 2\l__cal_tmpd_dim)/3}}
  }

  \tl_put_right:NV #1 \g__spath_lineto_tl
  \tl_put_right:Nx #1 {
    {\dim_use:N \l__cal_tmpc_dim}
    {\dim_use:N \l__cal_tmpd_dim}
  }
}
```

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

```
\cs_new_nopar:Npn \cal_split_curve_in_two:NN #1#2
{
  \spath_split_curve:nVNN {.5} #2 \l_tmpa_tl \l_tmpb_tl
  \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
  \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
  \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
  \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
  \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
  \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
  \tl_put_right:NV #1 \l_tmpa_tl
  \tl_put_right:NV #1 \l_tmpb_tl
}
```

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

```
\cs_new_nopar:Npn \cal_split_curve_in_three:NN #1#2
{
  \spath_split_curve:nVNN {1/3} #2 \l_tmpa_tl \l_tmpb_tl

  \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
  \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
  \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
  \tl_put_right:NV #1 \l_tmpa_tl

  \spath_split_curve:nVNN {.5} \l_tmpb_tl \l_tmpa_tl \l_tmpb_tl
  \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
  \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
  \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
  \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
  \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
  \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
  \tl_put_right:NV #1 \l_tmpa_tl
  \tl_put_right:NV #1 \l_tmpb_tl
}
```

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

```
\cs_new_nopar:Npn \cal_maybe_taper:N #1
{
  \tl_set_eq:NN \l__cal_tmpa_tl #1

  \bool_if:NT \l__cal_taper_start_bool
  {

    \dim_set:Nn \l__cal_tmpa_dim {\tl_item:Nn \l__cal_tmpa_tl {2}}
    \dim_set:Nn \l__cal_tmpb_dim {\tl_item:Nn \l__cal_tmpa_tl {3}}
    \tl_set:Nx \l__cal_tmpb_tl {\tl_item:Nn \l__cal_tmpa_tl {4}}

    \tl_case:Nnn \l__cal_tmpb_tl
    {
      \g__spath_lineto_tl
      {

        \bool_set_true:N \l__cal_taperable_bool
        \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {5}}
        \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {6}}
        \dim_set:Nn \l__cal_tmpc_dim {(2\l__cal_tmpa_dim + \l__cal_tmpg_dim)/3}
        \dim_set:Nn \l__cal_tmpd_dim {(2\l__cal_tmpb_dim + \l__cal_tmph_dim)/3}
        \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmpa_dim + 2\l__cal_tmpg_dim)/3}
        \dim_set:Nn \l__cal_tmpf_dim {(\l__cal_tmpb_dim + 2\l__cal_tmph_dim)/3}
        \prg_replicate:nn {4}
        {
          \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
        }
```

```
      \tl_put_left:NV \l__cal_tmpa_tl \g__spath_moveto_tl
    }
    \g__spath_curvetoa_tl
    {
      \bool_set_true:N \l__cal_taperable_bool
      \dim_set:Nn \l__cal_tmpc_dim {\tl_item:Nn \l__cal_tmpa_tl {5}}
      \dim_set:Nn \l__cal_tmpd_dim {\tl_item:Nn \l__cal_tmpa_tl {6}}
      \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {8}}
      \dim_set:Nn \l__cal_tmpf_dim {\tl_item:Nn \l__cal_tmpa_tl {9}}
      \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {11}}
      \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {12}}
      \prg_replicate:nn {10}
      {
        \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
      }
      \tl_put_left:NV \l__cal_tmpa_tl \g__spath_moveto_tl
    }
  }
  {
    \bool_set_false:N \l__cal_taperable_bool
  }

  \bool_if:NT \l__cal_taperable_bool
  {
    \__cal_taper_aux:
  }

}

\bool_if:NT \l__cal_taper_end_bool
{

  \dim_set:Nn \l__cal_tmpa_dim {\tl_item:Nn \l__cal_tmpa_tl {-2}}
  \dim_set:Nn \l__cal_tmpb_dim {\tl_item:Nn \l__cal_tmpa_tl {-1}}
  \tl_set:Nx \l__cal_tmpb_tl {\tl_item:Nn \l__cal_tmpa_tl {-3}}

  \tl_case:Nnn \l__cal_tmpb_tl
  {
    \g__spath_lineto_tl
    {

      \bool_set_true:N \l__cal_taperable_bool
      \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {-5}}
      \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {-4}}
      \dim_set:Nn \l__cal_tmpc_dim {(2\l__cal_tmpa_dim + \l__cal_tmpg_dim)/3}
      \dim_set:Nn \l__cal_tmpd_dim {(2\l__cal_tmpb_dim + \l__cal_tmph_dim)/3}
      \dim_set:Nn \l__cal_tmpe_dim {(\l__cal_tmpa_dim + 2\l__cal_tmpg_dim)/3}
      \dim_set:Nn \l__cal_tmpf_dim {(\l__cal_tmpb_dim + 2\l__cal_tmph_dim)/3}
      \tl_reverse:N \l__cal_tmpa_tl
      \prg_replicate:nn {3}
```

```
        {
          \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
        }
        \tl_reverse:N \l__cal_tmpa_tl
      }
      \g__spath_curveto_tl
      {
        \bool_set_true:N \l__cal_taperable_bool
        \dim_set:Nn \l__cal_tmpc_dim {\tl_item:Nn \l__cal_tmpa_tl {-5}}
        \dim_set:Nn \l__cal_tmpd_dim {\tl_item:Nn \l__cal_tmpa_tl {-4}}
        \dim_set:Nn \l__cal_tmpe_dim {\tl_item:Nn \l__cal_tmpa_tl {-8}}
        \dim_set:Nn \l__cal_tmpf_dim {\tl_item:Nn \l__cal_tmpa_tl {-7}}
        \dim_set:Nn \l__cal_tmpg_dim {\tl_item:Nn \l__cal_tmpa_tl {-11}}
        \dim_set:Nn \l__cal_tmph_dim {\tl_item:Nn \l__cal_tmpa_tl {-10}}
        \tl_reverse:N \l__cal_tmpa_tl
        \prg_replicate:nn {9}
        {
          \tl_set:Nx \l__cal_tmpa_tl {\tl_tail:N \l__cal_tmpa_tl}
        }
        \tl_reverse:N \l__cal_tmpa_tl
      }
    }
    {
      \bool_set_false:N \l__cal_taperable_bool
    }

    \bool_if:NT \l__cal_taperable_bool
    {
      \__cal_taper_aux:
    }

  }

  \pgfsyssoftpath@setcurrentpath\l__cal_tmpa_tl
  \pgfsetstrokecolor{\pgfkeysvalueof{/tikz/pen~colour}}
  \pgfusepath{stroke}

}
```

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

```
\cs_new_nopar:Npn \__cal_taper_aux:
{
  \tl_clear:N \l__cal_tmpb_tl
  \tl_put_right:NV \l__cal_tmpb_tl \g__spath_moveto_tl

  \fp_set:Nn \l__cal_tmpa_fp
  {
    \l__cal_tmpd_dim - \l__cal_tmpb_dim
  }
  \fp_set:Nn \l__cal_tmpb_fp
```

```
{
  \l__cal_tmpa_dim - \l__cal_tmpc_dim
}
\fp_set:Nn \l__cal_tmpe_fp
{
  (\l__cal_tmpa_fp^2 + \l__cal_tmpb_fp^2)^.5
}

\fp_set:Nn \l__cal_tmpa_fp {.5*\l__cal_taper_width_dim *     \l__cal_tmpa_fp / \l__cal_tmpe
\fp_set:Nn \l__cal_tmpb_fp {.5*\l__cal_taper_width_dim *     \l__cal_tmpb_fp / \l__cal_tmpe

\fp_set:Nn \l__cal_tmpc_fp
{
  \l__cal_tmph_dim - \l__cal_tmpf_dim
}
\fp_set:Nn \l__cal_tmpd_fp
{
  \l__cal_tmpe_dim - \l__cal_tmpg_dim
}
\fp_set:Nn \l__cal_tmpe_fp
{
  (\l__cal_tmpc_fp^2 + \l__cal_tmpd_fp^2)^.5
}

\fp_set:Nn \l__cal_tmpc_fp {.5*\l__cal_line_width_dim * \l__cal_tmpc_fp / \l__cal_tmpe_fp}
\fp_set:Nn \l__cal_tmpd_fp {.5*\l__cal_line_width_dim * \l__cal_tmpd_fp / \l__cal_tmpe_fp}

\tl_put_right:Nx \l__cal_tmpb_tl
{
  {\dim_eval:n { \fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim}}
  {\dim_eval:n { \fp_to_dim:N \l__cal_tmpb_fp +             \l__cal_tmpb_dim}}
}

\tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetoa_tl

\tl_put_right:Nx \l__cal_tmpb_tl
{
  {\dim_eval:n { \fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpc_dim}}
  {\dim_eval:n { \fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpd_dim}}
}

\tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetob_tl

\tl_put_right:Nx \l__cal_tmpb_tl
{
  {\dim_eval:n { \fp_to_dim:N \l__cal_tmpc_fp + \l__cal_tmpe_dim}}
  {\dim_eval:n { \fp_to_dim:N \l__cal_tmpd_fp + \l__cal_tmpf_dim}}
}

\tl_put_right:NV \l__cal_tmpb_tl \g__spath_curveto_tl
```

```
\tl_put_right:Nx \l__cal_tmpb_tl
{
  {\dim_eval:n { \fp_to_dim:N \l__cal_tmpc_fp + \l__cal_tmpg_dim}}
  {\dim_eval:n { \fp_to_dim:N \l__cal_tmpd_fp + \l__cal_tmph_dim}}
}

\tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetoa_tl

\tl_put_right:Nx \l__cal_tmpb_tl
{
  {\dim_eval:n { \fp_to_dim:N \l__cal_tmpc_fp + \l__cal_tmpg_dim - \fp_to_dim:n{ 1.32 * \l__
  {\dim_eval:n { \fp_to_dim:N \l__cal_tmpd_fp + \l__cal_tmph_dim + \fp_to_dim:n {1.32* \l__c
}

\tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetob_tl

\tl_put_right:Nx \l__cal_tmpb_tl
{
  {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpc_fp + \l__cal_tmpg_dim - \fp_to_dim:n {1.32 * \l_
  {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpd_fp + \l__cal_tmph_dim + \fp_to_dim:n {1.32 * \l_
}

\tl_put_right:NV \l__cal_tmpb_tl \g__spath_curveto_tl

\tl_put_right:Nx \l__cal_tmpb_tl
{
  {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpc_fp + \l__cal_tmpg_dim}}
  {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpd_fp + \l__cal_tmph_dim}}
}

\tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetoa_tl

\tl_put_right:Nx \l__cal_tmpb_tl
{
  {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpc_fp + \l__cal_tmpe_dim}}
  {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpd_fp + \l__cal_tmpf_dim}}
}

\tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetob_tl

\tl_put_right:Nx \l__cal_tmpb_tl
{
  {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpc_dim}}
  {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpd_dim}}
}

\tl_put_right:NV \l__cal_tmpb_tl \g__spath_curveto_tl

\tl_put_right:Nx \l__cal_tmpb_tl
```

```
  {
    {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim}}
    {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim}}
  }

  \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetoa_tl

  \tl_put_right:Nx \l__cal_tmpb_tl
  {
    {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim + \fp_to_dim:n{ 1.32 * \l
    {\dim_eval:n { -\fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim - \fp_to_dim:n {1.32* \l__
  }

  \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curvetob_tl

  \tl_put_right:Nx \l__cal_tmpb_tl
  {
    {\dim_eval:n { \fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim + \fp_to_dim:n {1.32 * \l__
    {\dim_eval:n { \fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim - \fp_to_dim:n {1.32 * \l__
  }

  \tl_put_right:NV \l__cal_tmpb_tl \g__spath_curveto_tl

  \tl_put_right:Nx \l__cal_tmpb_tl
  {
    {\dim_eval:n { \fp_to_dim:N \l__cal_tmpa_fp + \l__cal_tmpa_dim}}
    {\dim_eval:n { \fp_to_dim:N \l__cal_tmpb_fp +             \l__cal_tmpb_dim}}
  }

  \pgfsyssoftpath@setcurrentpath\l__cal_tmpb_tl
  \pgfsetfillcolor{\pgfkeysvalueof{/tikz/pen~colour}}
  \pgfusepath{fill}
}
```

Defines a copperplate pen.

```
\tl_set:Nn \l__cal_tmpa_tl {\pgfsyssoftpath@movetotoken{0pt}{0pt}}
\MakeSPathList{calligraphy pen copperplate}{\l__cal_tmpa_tl}
\SPathListPrepare{calligraphy pen copperplate}

\ExplSyntaxOff
```

3.4 Decorations

If a decoration library is loaded we define some decorations that use the calligraphy library, specifically the copperplate pen with its tapering.

First, a brace decoration.

```
\expandafter\ifx\csname pgfdeclaredecoration\endcsname\relax
\else
\pgfdeclaredecoration{calligraphic brace}{brace}
{
```

```
  \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]
  {
    \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}
    \pgfpathmoveto{\pgfpointorigin}
    \pgfpathcurveto
    {\pgfqpoint{.15\pgfdecorationsegmentamplitude}{.3\pgfdecorationsegmentamplitude}}
    {\pgfqpoint{.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
    {\pgfqpoint{\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
    {
      \pgftransformxshift{+\pgfdecorationsegmentaspect\pgfdecoratedremainingdistance}
      \pgfpathlineto{\pgfqpoint{-\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplit
      \pgfpathcurveto
      {\pgfqpoint{-.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
      {\pgfqpoint{-.15\pgfdecorationsegmentamplitude}{.7\pgfdecorationsegmentamplitude}}
      {\pgfqpoint{0\pgfdecorationsegmentamplitude}{1\pgfdecorationsegmentamplitude}}
      \pgfpathmoveto{\pgfqpoint{0\pgfdecorationsegmentamplitude}{1\pgfdecorationsegmentamplitu
      \pgfpathcurveto
      {\pgfqpoint{.15\pgfdecorationsegmentamplitude}{.7\pgfdecorationsegmentamplitude}}
      {\pgfqpoint{.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
      {\pgfqpoint{\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
    }
    {
      \pgftransformxshift{+\pgfdecoratedremainingdistance}
      \pgfpathlineto{\pgfqpoint{-\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplit
      \pgfpathcurveto
      {\pgfqpoint{-.5\pgfdecorationsegmentamplitude}{.5\pgfdecorationsegmentamplitude}}
      {\pgfqpoint{-.15\pgfdecorationsegmentamplitude}{.3\pgfdecorationsegmentamplitude}}
      {\pgfqpoint{0pt}{0pt}}
    }
    \tikzset{
      taper width=.5\pgflinewidth,
      taper
    }%
    \pgfsyssoftpath@getcurrentpath\cal@tmp@path
    \MakeSPathList{calligraphy path}{\cal@tmp@path}%
    \SPathListPrepare{calligraphy path}%
    \CalligraphyPathCreate{calligraphy path}{copperplate}%
  }
  \state{final}{}
}
```

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

```
\pgfdeclaredecoration{calligraphic straight parenthesis}{brace}
{
  \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]
  {
    \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}
    \pgfpathmoveto{\pgfpointorigin}
    \pgfpathcurveto
```

```
    {\pgfqpoint{.76604\pgfdecorationsegmentamplitude}{.64279\pgfdecorationsegmentamplitude}}
    {\pgfqpoint{2.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
    {\pgfqpoint{3.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
    {
      \pgftransformxshift{+\pgfdecoratedremainingdistance}
      \pgfpathlineto{\pgfqpoint{-3.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentar
      \pgfpathcurveto
      {\pgfqpoint{-2.3333\pgfdecorationsegmentamplitude}{\pgfdecorationsegmentamplitude}}
      {\pgfqpoint{-.76604\pgfdecorationsegmentamplitude}{.64279\pgfdecorationsegmentamplitude}
      {\pgfqpoint{0pt}{0pt}}
    }
    \tikzset{
      taper width=.5\pgflinewidth,
      taper
    }%
    \pgfsyssoftpath@getcurrentpath\cal@tmp@path
    \MakeSPathList{calligraphy path}{\cal@tmp@path}%
    \SPathListPrepare{calligraphy path}%
    \CalligraphyPathCreate{calligraphy path}{copperplate}%
  }
  \state{final}{}%
}
```

The third is a curved parenthesis.

```
\pgfdeclaredecoration{calligraphic curved parenthesis}{brace}
{
  \state{brace}[width=+\pgfdecoratedremainingdistance,next state=final]
  {
    \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}
    \pgfpathmoveto{\pgfpointorigin}
    \pgf@xa=\pgfdecoratedremainingdistance\relax
    \advance\pgf@xa by -1.5890\pgfdecorationsegmentamplitude\relax
    \edef\cgrphy@xa{\the\pgf@xa}
    \pgfpathcurveto
    {\pgfqpoint{1.5890\pgfdecorationsegmentamplitude}{1.3333\pgfdecorationsegmentamplitude}}
    {\pgfqpoint{\cgrphy@xa}{1.3333\pgfdecorationsegmentamplitude}}
    {\pgfqpoint{\pgfdecoratedremainingdistance}{0pt}}
    \tikzset{
      taper width=.5\pgflinewidth,
      taper
    }%
    \pgfsyssoftpath@getcurrentpath\cal@tmp@path
    \MakeSPathList{calligraphy path}{\cal@tmp@path}%
    \SPathListPrepare{calligraphy path}%
    \CalligraphyPathCreate{calligraphy path}{copperplate}%
  }
  \state{final}{}%
}
```

End the conditional for if pgfdecoration module is loaded

```
\fi
```

4 Drawing Knots

4.1 Initialisation

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

```
\RequirePackage{spath3}
\usetikzlibrary{intersections}

\ExplSyntaxOn

\tl_new:N \l__knot_tmpa_tl
\tl_new:N \l__knot_tmpb_tl
\tl_new:N \l__knot_tmpc_tl
\tl_new:N \l__knot_tmpd_tl
\tl_new:N \l__knot_tmpe_tl
\tl_new:N \l__knot_tmpf_tl
\tl_new:N \l__knot_tmpg_tl
\tl_new:N \l__knot_redraws_tl
\tl_new:N \l__knot_clip_width_tl
\tl_new:N \l__knot_name_tl
\tl_new:N \l__knot_node_tl

\int_new:N \l__knot_tmpa_int
\int_new:N \l__knot_strands_int
\int_new:N \l__knot_intersections_int
\int_new:N \l__knot_filaments_int
\int_new:N \l__knot_component_start_int

\dim_new:N \l__knot_tmpa_dim
\dim_new:N \l__knot_tmpb_dim
\dim_new:N \l__knot_tmpc_dim
\dim_new:N \l__knot_tolerance_dim
\dim_new:N \l__knot_clip_radius_dim

\bool_new:N \l__knot_draft_bool
\bool_new:N \l__knot_ignore_ends_bool
\bool_new:N \l__knot_self_intersections_bool
\bool_new:N \l__knot_splits_bool
\bool_new:N \l__knot_super_draft_bool

\bool_new:N \l__knot_prepend_prev_bool
\bool_new:N \l__knot_append_next_bool
\bool_new:N \l__knot_skip_bool

\bool_set_true:N \l__knot_ignore_ends_bool
```

Configuration is via TikZ keys and styles.

```
\tikzset{
  knot/.code={
    \tl_if_eq:nnTF {#1} {none}
```

```
    {
      \tikz@addmode{\tikz@mode@doublefalse}
    }
    {
      \tikz@addmode{\tikz@mode@doubletrue}
      \tl_if_empty:nTF {#1}
      {
        \pgfsetinnerstrokecolor{\tikz@strokecolor}
      }
      {
        \pgfsetinnerstrokecolor{#1}
      }
      \tikz@addoption{\pgfsetstrokecolor{knotbg}}
      \tl_set:Nn \tikz@double@setup{
        \pgfsetinnerlinewidth{\pgflinewidth}
        \pgfsetlinewidth{\dim_eval:n {\tl_use:N \l__knot_gap_tl \pgflinewidth}}
      }
    }
  },
  knot~ gap/.store~ in=\l__knot_gap_tl,
  knot~ gap=3,
  knot~ diagram/.is~family,
  knot~ diagram/.unknown/.code={
    \tl_set_eq:NN \l__knot_tmpa_tl \pgfkeyscurrentname
    \pgfkeysalso{
      /tikz/\l__knot_tmpa_tl=#1
    }
  },
  background~ colour/.code={%
    \colorlet{knotbg}{#1}%
  },
  background~ color/.code={%
    \colorlet{knotbg}{#1}%
  },
  background~ colour=white,
  knot~ diagram,
  name/.store~in=\l__knot_name_tl,
  name={knot},
  every~ strand/.style={draw},
  ignore~ endpoint~ intersections/.code={
    \tl_if_eq:nnTF {#1} {true}
    {
      \bool_set_true:N \l__knot_ignore_ends_bool
    }
    {
      \bool_set_false:N \l__knot_ignore_ends_bool
    }
  },
  ignore~ endpoint~ intersections/.default=true,
  consider~ self~ intersections/.is~choice,
```

```
  consider~ self~ intersections/true/.code={
    \bool_set_true:N \l__knot_self_intersections_bool
    \bool_set_true:N \l__knot_splits_bool
  },
  consider~ self~ intersections/false/.code={
    \bool_set_false:N \l__knot_self_intersections_bool
    \bool_set_false:N \l__knot_splits_bool
  },
  consider~ self~ intersections/no~ splits/.code={
    \bool_set_true:N \l__knot_self_intersections_bool
    \bool_set_false:N \l__knot_splits_bool
  },
  consider~ self~ intersections/.default={true},
  clip~ radius/.code={
    \dim_set:Nn \l__knot_clip_radius_dim {#1}
  },
  clip~ radius=10pt,
  end~ tolerance/.code={
    \dim_set:Nn \l__knot_tolerance_dim {#1}
  },
  end~ tolerance=14pt,
  clip~ width/.code={
    \tl_set:Nn \l__knot_clip_width_tl {#1}
  },
  clip~ width=3,
  flip~ crossing/.code={%
    \tl_clear_new:c {l__knot_crossing_#1}
    \tl_set:cn {l__knot_crossing_#1} {x}
  },
  draft~ mode/.is~ choice,
  draft~ mode/off/.code={%
    \bool_set_false:N \l__knot_draft_bool
    \bool_set_false:N \l__knot_super_draft_bool
  },
  draft~ mode/crossings/.code={%
    \bool_set_true:N \l__knot_draft_bool
    \bool_set_false:N \l__knot_super_draft_bool
  },
  draft~ mode/strands/.code={%
    \bool_set_true:N \l__knot_draft_bool
    \bool_set_true:N \l__knot_super_draft_bool
  },
  draft/.is~ family,
  draft,
  crossing~ label/.style={
    overlay,
    fill=white,
    fill~ opacity=.5,
    text~ opacity=1,
    text=blue,
```

```
    pin~ edge={blue,<-}
  },
  strand~ label/.style={
    overlay,
    circle,
    draw=purple,
    fill=white,
    fill~ opacity=.5,
    text~ opacity=1,
    text=purple,
    inner~ sep=0pt
  },
}
```

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

```
\cs_new_nopar:Npn \knot_apply_style:N #1
{
  \tl_if_exist:NT #1 {
    \exp_args:NV \tikzset #1
  }
}
\cs_generate_variant:Nn \knot_apply_style:N {c}
```

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

```
\NewDocumentCommand \flipcrossings {m}
{
  \tikzset{knot~ diagram/flip~ crossing/.list={#1}}%
}
```

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

```
\NewDocumentCommand \strand { O{} }
{
  \int_incr:N \l__knot_strands_int
  \tl_clear_new:c {l__knot_options_strand \int_use:N \l__knot_strands_int}
  \tl_set:cn {l__knot_options_strand \int_use:N \l__knot_strands_int} {#1}
  \path[#1,save~ spath=knot strand \int_use:N  \l__knot_strands_int]
}
```

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

```
\NewDocumentEnvironment{knot} { O{} }
{
  \knot_initialise:n {#1}
}
{
  \knot_render:
}
```

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

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

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

```
\cs_new_protected_nopar:Npn \knot_render:
{
```

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

```
  \pgfscope
  \pgftransformreset
```

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

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

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

```
  \bool_if:NF \l__knot_super_draft_bool
  {
```

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.

```
    \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.

```
    \bool_if:NTF \l__knot_self_intersections_bool
    {
      \knot_split_strands:
      \int_set_eq:NN \l__knot_tmpa_int \l__knot_filaments_int
      \tl_set:Nn \l__knot_prefix_tl {filament}
    }
    {
      \int_set_eq:NN \l__knot_tmpa_int \l__knot_strands_int
      \tl_set:Nn \l__knot_prefix_tl {strand}
    }
```

Initialise the intersection count.

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

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

```
    \bool_if:NTF \l__knot_draft_bool
    {
      \tl_set:Nn \l__knot_node_tl
      {\node[coordinate,pin={[knot~ diagram/draft/crossing~ label]{\int_use:N \l__knot_interse
    }
    {
      \tl_set:Nn \l__knot_node_tl {\node[coordinate]}
    }
```

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

```
    \int_step_variable:nnnNn {1} {1} {\l__knot_tmpa_int - 1} \l__knot_tmpa_tl
    {
      \int_step_variable:nnnNn {\tl_use:N \l__knot_tmpa_tl + 1} {1}     {\l__knot_tmpa_int} \l
      {
        \knot_intersections:VV \l__knot_tmpa_tl \l__knot_tmpb_tl
      }
    }
```

If any redraws were requested, do them here.

```
    \tl_use:N \l__knot_redraws_tl
  }
```

Close the scope

```
  \endpgfscope
}
```

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

```
\cs_new_protected_nopar:Npn \knot_draw_strand:n #1
{
  \pgfscope
  \group_begin:
  \tl_set:Nn \l_tmpa_tl {knot~ diagram/every~ strand/.try,}
  \tl_put_right:Nv \l_tmpa_tl {l__knot_options_strand #1}
  \tl_put_right:Nn \l_tmpa_tl {,knot~ diagram/only~ when~ rendering/.try,only~ when~ rendering
  \spath_tikz_path:Vn \l_tmpa_tl {knot strand #1}
  \group_end:
  \endpgfscope
}
\cs_generate_variant:Nn \tl_put_right:Nn {Nv}
```

`\knot_draw_labels:n` Draw a label at each end of each strand, if in draft mode. Also, if requested, split potentially self intersecting Bezier curves.

```
\cs_new_protected_nopar:Npn \knot_draw_labels:n #1
{
  \bool_if:NT \l__knot_draft_bool
  {
    \spath_get:nnN {knot strand #1} {final point} \l__knot_tmpb_tl
    \dim_set:Nn \l__knot_tmpa_dim {\tl_item:Nn \l__knot_tmpb_tl {1}}
    \dim_set:Nn \l__knot_tmpb_dim {\tl_item:Nn \l__knot_tmpb_tl {2}}
    \node[knot~ diagram/draft/strand~label] at (\l__knot_tmpa_dim,\l__knot_tmpb_dim) {#1};
    \spath_get:nnN {knot strand #1} {initial point} \l__knot_tmpb_tl
    \dim_set:Nn \l__knot_tmpa_dim {\tl_item:Nn \l__knot_tmpb_tl {1}}
    \dim_set:Nn \l__knot_tmpb_dim {\tl_item:Nn \l__knot_tmpb_tl {2}}
    \node[knot~ diagram/draft/strand~label] at (\l__knot_tmpa_dim,\l__knot_tmpb_dim) {#1};
  }
  \bool_if:nT {
    \l__knot_self_intersections_bool
    &&
```

```
    \l__knot_splits_bool
  }
  {
    \tl_clear:N \l__knot_tmpa_tl
    \spath_map_segment_function:nN {knot strand #1} \knot_split_self_intersects:NN
    \spath_put:nnV {knot strand #1} {path} \l__knot_tmpa_tl
  }
}
```

\knot_split_self_intersects:NN

This is the macro that does the split. Figuring out whether a Bezier cubic self intersects is apparently a difficult problem so we don't bother. We compute a point such that if there is an intersection then it lies on either side of the point. I don't recall where the formula came from!

```
\cs_new_protected_nopar:Npn \knot_split_self_intersects:NN #1#2
{
  \tl_case:Nnn #1
  {
    \g__spath_curvetoa_tl
    {
      \fp_set:Nn \l_tmpa_fp
      {
        (\tl_item:Nn #2 {3} - 3 * \tl_item:Nn #2 {6} + 3 * \tl_item:Nn #2 {9} - \tl_item:Nn #2
        *
        (3 * \tl_item:Nn #2 {8} - 3 * \tl_item:Nn #2 {11})
        -
        (\tl_item:Nn #2 {2} - 3 * \tl_item:Nn #2 {5} + 3 * \tl_item:Nn #2 {8} - \tl_item:Nn #2
        *
        (3 * \tl_item:Nn #2 {9} - 3 * \tl_item:Nn #2 {12})
      }
      \fp_set:Nn \l_tmpb_fp
      {
        (\tl_item:Nn #2 {2} - 3 * \tl_item:Nn #2 {5} + 3 * \tl_item:Nn #2 {8} - \tl_item:Nn #2
        *
        (3 * \tl_item:Nn #2 {6} - 6 * \tl_item:Nn #2 {9} + 3 * \tl_item:Nn #2 {12})
        -
        (\tl_item:Nn #2 {3} - 3 * \tl_item:Nn #2 {6} + 3 * \tl_item:Nn #2 {9} - \tl_item:Nn #2
        *
        (3 * \tl_item:Nn #2 {5} - 6 * \tl_item:Nn #2 {8} + 3 * \tl_item:Nn #2 {11})
      }
      \fp_compare:nTF
      {
        \l_tmpb_fp != 0
      }
      {
        \fp_set:Nn \l_tmpa_fp {.5 * \l_tmpa_fp / \l_tmpb_fp}
        \fp_compare:nTF
        {
          0 < \l_tmpa_fp && \l_tmpa_fp < 1
        }
```

```
        {
          \spath_split_curve:VVNN \l_tmpa_fp #2 \l_tmpa_tl \l_tmpb_tl
          \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
          \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
          \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
          \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
          \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
          \tl_set:Nx \l_tmpb_tl {\tl_tail:N \l_tmpb_tl}
          \tl_put_right:NV \l__knot_tmpa_tl \l_tmpa_tl
          \tl_put_right:NV \l__knot_tmpa_tl \l_tmpb_tl
        }
        {
          \tl_set_eq:NN \l_tmpa_tl #2
          \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
          \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
          \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
          \tl_put_right:NV \l__knot_tmpa_tl \l_tmpa_tl
        }
      }
      {
        \tl_set_eq:NN \l_tmpa_tl #2
        \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
        \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
        \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
        \tl_put_right:NV \l__knot_tmpa_tl \l_tmpa_tl
      }
    }
    \g__spath_lineto_tl
    {
      \tl_set_eq:NN \l_tmpa_tl #2
      \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
      \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
      \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
      \tl_put_right:NV \l__knot_tmpa_tl \l_tmpa_tl
    }
  }
  {
    \tl_put_right:NV \l__knot_tmpa_tl #2
  }
}
```

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

```
\cs_new_protected_nopar:Npn \knot_intersections:nn #1#2
{
  \group_begin:
  \tl_set_eq:NN \l__knot_tmpa_tl \l__knot_prefix_tl
  \tl_put_right:Nn \l__knot_tmpa_tl {#1}
  \tl_set_eq:NN \l__knot_tmpb_tl \l__knot_prefix_tl
  \tl_put_right:Nn \l__knot_tmpb_tl {#2}
  \spath_get:nnN {knot \tl_use:N \l__knot_tmpa_tl} {path} \l__knot_tmpc_tl
```

```
  \spath_get:nnN {knot \tl_use:N \l__knot_tmpb_tl} {path} \l__knot_tmpd_tl

  \pgfintersectionofpaths{\pgfsetpath\l__knot_tmpc_tl}{\pgfsetpath\l__knot_tmpd_tl}

  \int_compare:nT {\pgfintersectionsolutions > 0}
  {
    \int_step_function:nnnN {1} {1} {\pgfintersectionsolutions} \knot_do_intersection:n
  }
  \group_end:
}
```

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

```
\cs_new_protected_nopar:Npn \knot_do_intersection:n #1
{
```

Get the intersection coordinates.

```
  \pgfpointintersectionsolution{#1}
  \dim_set:Nn \l__knot_tmpa_dim {\pgf@x}
  \dim_set:Nn \l__knot_tmpb_dim {\pgf@y}
```

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

```
  \bool_set_false:N \l__knot_skip_bool
  \bool_if:NT \l__knot_self_intersections_bool
  {
```

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

```
    \tl_set:Nn \l_tmpa_tl {knot previous}
    \tl_put_right:NV \l_tmpa_tl \l__knot_tmpa_tl
    \tl_set:Nv \l_tmpa_tl \l_tmpa_tl
    \tl_if_eq:NNT \l_tmpa_tl \l__knot_tmpb_tl
    {
      \knot_test_endpoint:VnT \l__knot_tmpb_tl {final point}
      {
        \bool_set_true:N \l__knot_skip_bool
      }
    }

    \tl_set:Nn \l_tmpa_tl {knot previous}
    \tl_put_right:NV \l_tmpa_tl \l__knot_tmpb_tl
    \tl_set:Nv \l_tmpa_tl \l_tmpa_tl
    \tl_if_eq:NNT \l_tmpa_tl \l__knot_tmpa_tl
    {
      \knot_test_endpoint:VnT \l__knot_tmpa_tl {final point}
      {
        \bool_set_true:N \l__knot_skip_bool
      }
    }
  }
```

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

```
\bool_if:NT \l__knot_ignore_ends_bool
{
  \knot_test_endpoint:VnT \l__knot_tmpa_tl {initial point}
  {
    \bool_set_true:N \l__knot_skip_bool
  }
  \knot_test_endpoint:VnT \l__knot_tmpa_tl {final point}
  {
    \bool_set_true:N \l__knot_skip_bool
  }
  \knot_test_endpoint:VnT \l__knot_tmpb_tl {initial point}
  {
    \bool_set_true:N \l__knot_skip_bool
  }
  \knot_test_endpoint:VnT \l__knot_tmpb_tl {final point}
  {
    \bool_set_true:N \l__knot_skip_bool
  }
}
```

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

```
\bool_if:NF \l__knot_skip_bool
{

  \int_gincr:N \l__knot_intersections_int
```

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

```
  \bool_if:nTF
  {
    \tl_if_exist_p:c {l__knot_crossing_ \int_use:N
      \l__knot_intersections_int}
    &&
    ! \tl_if_empty_p:c {l__knot_crossing_ \int_use:N
      \l__knot_intersections_int}
  }
  {
    \tl_set_eq:NN \l__knot_tmpg_tl \l__knot_tmpb_tl
  }
  {
    \tl_set_eq:NN \l__knot_tmpg_tl \l__knot_tmpa_tl
  }
```

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.

```
  \bool_if:NT \l__knot_self_intersections_bool
  {
```

```
      \knot_test_endpoint:VnT \l__knot_tmpg_tl {initial point}
      {
        \bool_set_true:N \l__knot_prepend_prev_bool
      }
      {
        \bool_set_false:N \l__knot_prepend_prev_bool
      }

      \knot_test_endpoint:VnT \l__knot_tmpg_tl {final point}
      {
        \bool_set_true:N \l__knot_append_next_bool
      }
      {
        \bool_set_false:N \l__knot_append_next_bool
      }
```

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

```
      \bool_if:nT
      {
        \l__knot_prepend_prev_bool || \l__knot_append_next_bool
      }
      {
        \spath_clone:nn {knot \tl_use:N \l__knot_tmpg_tl}
        {knot \tl_use:N \l__knot_prefix_tl -1}

        \tl_set_eq:cc {l__knot_options_ \tl_use:N \l__knot_prefix_tl -1} {l__knot_options_ \tl

        \bool_if:NT \l__knot_prepend_prev_bool
        {
          \spath_prepend_no_move:nn {knot \tl_use:N \l__knot_prefix_tl -1} {knot \tl_use:c {kn
```

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

```
          \bool_if:NT \l__knot_splits_bool
          {
            \knot_test_endpoint:vnT {knot previous \tl_use:N \l__knot_tmpg_tl} {initial point}
            {
              \spath_get:nnN {knot \tl_use:N \l__knot_prefix_tl -1} {path} \l_tmpa_tl
              \spath_prepend_no_move:nn {knot \tl_use:N \l__knot_prefix_tl -1} {knot \tl_use:c
              \spath_get:nnN {knot \tl_use:N \l__knot_prefix_tl -1} {path} \l_tmpa_tl

            }
          }
        }
```

Now the same for appending.

```
        \bool_if:NT \l__knot_append_next_bool
        {
          \spath_append_no_move:nn {knot \tl_use:N \l__knot_prefix_tl -1} {knot \tl_use:c {kno
          \bool_if:NT \l__knot_splits_bool
          {
```

```
            \knot_test_endpoint:vnT {knot previous \tl_use:N \l__knot_tmpg_tl} {final point}
            {
              \spath_append_no_move:nn {knot \tl_use:N \l__knot_prefix_tl -1} {knot \tl_use:c

            }
          }
        }

        \tl_set:Nn \l__knot_tmpg_tl {\tl_use:N \l__knot_prefix_tl -1}
      }
    }
```

Now we render the crossing.

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

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

```
    \tl_use:N \l__knot_node_tl (\l__knot_name_tl \c_space_tl \int_use:N \l__knot_intersections
  }
}

\cs_generate_variant:Nn \knot_intersections:nn {VV}
```

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

```
\prg_new_conditional:Npnn \knot_test_endpoint:N #1 {p,T,F,TF}
{
  \dim_compare:nTF
  {
    \dim_abs:n {\pgf@x - \tl_item:Nn #1 {1}}
    +
    \dim_abs:n {\pgf@y - \tl_item:Nn #1 {2}}
    <
    \l__knot_tolerance_dim
  }
  {
    \prg_return_true:
  }
  {
    \prg_return_false:
  }
}
```

`\knot_test_endpoint:nn` Wrapper around the above.

```
\prg_new_protected_conditional:Npnn \knot_test_endpoint:nn #1#2 {T,F,TF}
{
  \spath_get:nnN {knot #1} {#2} \l__knot_tmpd_tl
  \knot_test_endpoint:NTF \l__knot_tmpd_tl
  {
```

```
    \prg_return_true:
  }
  {
    \prg_return_false:
  }
}

\cs_generate_variant:Nn \knot_test_endpoint:nnT {VnT,vnT}
\cs_generate_variant:Nn \knot_test_endpoint:nnF {VnF,vnF}
\cs_generate_variant:Nn \knot_test_endpoint:nnTF {VnTF,vnTF}
```

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

```
\cs_new_nopar:Npn \knot_draw_crossing:nnn #1#2#3
{
  \group_begin:
  \pgfscope
  \clip (#2, #3) circle[radius=\l__knot_clip_radius_dim];

  \tl_set:Nn \l_tmpa_tl {knot~ diagram/every~ strand/.try,}
  \tl_if_exist:cT {l__knot_options_ #1}
  {
  \tl_put_right:Nv \l_tmpa_tl {l__knot_options_ #1}
  }
  \tl_put_right:Nn \l_tmpa_tl {,knotbg,line~ width= \tl_use:N \l__knot_clip_width_tl * \pgflin
  \spath_tikz_path:Vn \l_tmpa_tl {knot #1}

  \endpgfscope

  \pgfscope
  \clip (#2, #3) circle[radius=1.1\l__knot_clip_radius_dim];

  \tl_set:Nn \l_tmpa_tl {knot~ diagram/every~ strand/.try,}
  \tl_if_exist:cT {l__knot_options_ #1}
  {
  \tl_put_right:Nv \l_tmpa_tl {l__knot_options_ #1}
  }
  \tl_put_right:Nn \l_tmpa_tl {,knot~ diagram/only~ when~ rendering/.try,only~ when~ renderin
  \spath_tikz_path:Vn \l_tmpa_tl {knot #1}

  \endpgfscope
  \group_end:
}

\cs_generate_variant:Nn \knot_draw_crossing:nnn {nVV}
```

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

```
\cs_new_protected_nopar:Npn \knot_split_strands:
{
  \int_gzero:N \l__knot_filaments_int
```

```
  \int_step_function:nnnN {1} {1} {\l__knot_strands_int} \knot_split_strand:n
  \int_step_function:nnnN {1} {1} {\l__knot_filaments_int} \knot_compute_nexts:n
}
```

`\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.

```
\cs_new_protected_nopar:Npn \knot_compute_nexts:n #1
{
  \tl_clear_new:c {knot next \tl_use:c {knot previous filament #1}}
  \tl_set:cn {knot next \tl_use:c {knot previous filament #1}} {filament #1}
}
```

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

```
\cs_new_protected_nopar:Npn \knot_split_strand:n #1
{
  \int_set_eq:NN \l__knot_component_start_int \l__knot_filaments_int
  \int_incr:N \l__knot_component_start_int
  \spath_map_segment_function:nN {knot strand #1} \knot_save_filament:NN
}
```

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

```
\cs_new_protected_nopar:Npn \knot_save_filament:NN #1#2
{
  \tl_case:Nnn #1
  {
    \g__spath_moveto_tl
    {
      \int_compare:nT {\l__knot_component_start_int < \l__knot_filaments_int}
      {
        \int_set_eq:NN \l__knot_component_start_int \l__knot_filaments_int
      }
    }
    \g__spath_lineto_tl
    {
      \int_gincr:N \l__knot_filaments_int
      \spath_clear_new:n {knot filament \int_use:N \l__knot_filaments_int}
      \spath_put:nnV {knot filament \int_use:N \l__knot_filaments_int} {path} #2
      \tl_clear_new:c {knot previous filament \int_use:N \l__knot_filaments_int}
      \int_compare:nF {\l__knot_component_start_int == \l__knot_filaments_int}
      {
        \tl_set:cx {knot previous filament \int_use:N \l__knot_filaments_int} {filament \int_e
      }
    }
    \g__spath_curvetoa_tl
    {
      \int_gincr:N \l__knot_filaments_int
      \spath_clear_new:n {knot filament \int_use:N \l__knot_filaments_int}
      \spath_put:nnV {knot filament \int_use:N \l__knot_filaments_int} {path} #2
      \tl_clear_new:c {knot previous filament \int_use:N \l__knot_filaments_int}
```

```
      \int_compare:nF {\l__knot_component_start_int == \l__knot_filaments_int}
      {
        \tl_set:cx {knot previous filament \int_use:N \l__knot_filaments_int} {filament \int_e
      }
    }
    \g__spath_close_tl
    {
      \int_gincr:N \l__knot_filaments_int
      \spath_clear_new:n {knot filament \int_use:N \l__knot_filaments_int}
      \tl_set_eq:NN \l_tmpa_tl #2
      \tl_set:Nx \l_tmpa_tl {\tl_item:Nn #2 {1}\tl_item:Nn #2 {2}\tl_item:Nn #2 {3}}
      \tl_put_right:NV \l_tmpa_tl \g__spath_lineto_tl
      \tl_set:Nx \l_tmpa_tl {\tl_item:Nn #2 {5}\tl_item:Nn #2 {6}}
      \spath_put:nnV {knot filament \int_use:N \l__knot_filaments_int} {path} \l_tmpa_tl
      \tl_clear_new:c {knot previous filament \int_use:N \l__knot_filaments_int}
      \int_compare:nF {\l__knot_component_start_int == \l__knot_filaments_int}
      {
        \tl_set:cx {knot previous filament \int_use:N \l__knot_filaments_int} {filament \int_e
      }
      \tl_set:cx {knot previous filament \int_use:N \l__knot_component_start_int} {filament \i
    }
  }
  {
  }
}
```

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

```
\NewDocumentCommand \redraw { m m }
{
  \tikz@scan@one@point\pgfutil@firstofone #2 \relax
  \tl_put_right:Nn \l__knot_redraws_tl {\knot_draw_crossing:nnn}
  \tl_put_right:Nx \l__knot_redraws_tl {
    {strand #1} {\dim_use:N \pgf@x} {\dim_use:N \pgf@y}
  }
}
```

```
\ExplSyntaxOff
```

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

```
\def\pgf@sh@@knotanchor#1#2{%
  \anchor{#2 north west}{%
    \csname pgf@anchor@knot #1@north west\endcsname%
    \pgf@x=#2\pgf@x%
    \pgf@y=#2\pgf@y%
  }%
  \anchor{#2 north east}{%
    \csname pgf@anchor@knot #1@north east\endcsname%
    \pgf@x=#2\pgf@x%
    \pgf@y=#2\pgf@y%
  }%
```

```
  \anchor{#2 south west}{%
    \csname pgf@anchor@knot #1@south west\endcsname%
    \pgf@x=#2\pgf@x%
    \pgf@y=#2\pgf@y%
  }%
  \anchor{#2 south east}{%
    \csname pgf@anchor@knot #1@south east\endcsname%
    \pgf@x=#2\pgf@x%
    \pgf@y=#2\pgf@y%
  }%
  \anchor{#2 north}{%
    \csname pgf@anchor@knot #1@north\endcsname%
    \pgf@x=#2\pgf@x%
    \pgf@y=#2\pgf@y%
  }%
  \anchor{#2 east}{%
    \csname pgf@anchor@knot #1@east\endcsname%
    \pgf@x=#2\pgf@x%
    \pgf@y=#2\pgf@y%
  }%
  \anchor{#2 west}{%
    \csname pgf@anchor@knot #1@west\endcsname%
    \pgf@x=#2\pgf@x%
    \pgf@y=#2\pgf@y%
  }%
  \anchor{#2 south}{%
    \csname pgf@anchor@knot #1@south\endcsname%
    \pgf@x=#2\pgf@x%
    \pgf@y=#2\pgf@y%
  }%
}
```

knotcrossing

```
\pgfdeclareshape{knot crossing}
{
  \inheritsavedanchors[from=circle] % this is nearly a circle
  \inheritanchorborder[from=circle]
  \inheritanchor[from=circle]{north}
  \inheritanchor[from=circle]{north west}
  \inheritanchor[from=circle]{north east}
  \inheritanchor[from=circle]{center}
  \inheritanchor[from=circle]{west}
  \inheritanchor[from=circle]{east}
  \inheritanchor[from=circle]{mid}
  \inheritanchor[from=circle]{mid west}
  \inheritanchor[from=circle]{mid east}
  \inheritanchor[from=circle]{base}
  \inheritanchor[from=circle]{base west}
  \inheritanchor[from=circle]{base east}
  \inheritanchor[from=circle]{south}
```

```
  \inheritanchor[from=circle]{south west}
  \inheritanchor[from=circle]{south east}
  \inheritanchorborder[from=circle]
  \pgf@sh@@knotanchor{crossing}{2}
  \pgf@sh@@knotanchor{crossing}{3}
  \pgf@sh@@knotanchor{crossing}{4}
  \pgf@sh@@knotanchor{crossing}{8}
  \pgf@sh@@knotanchor{crossing}{16}
  \pgf@sh@@knotanchor{crossing}{32}
  \backgroundpath{
    \pgfutil@tempdima=\radius%
    \pgfmathsetlength{\pgf@xb}{\pgfkeysvalueof{/pgf/outer xsep}}%
    \pgfmathsetlength{\pgf@yb}{\pgfkeysvalueof{/pgf/outer ysep}}%
    \ifdim\pgf@xb<\pgf@yb%
      \advance\pgfutil@tempdima by-\pgf@yb%
    \else%
      \advance\pgfutil@tempdima by-\pgf@xb%
    \fi%
  }
}
```

knotovercross

```
\pgfdeclareshape{knot over cross}
{
  \inheritsavedanchors[from=rectangle] % this is nearly a circle
  \inheritanchorborder[from=rectangle]
  \inheritanchor[from=rectangle]{north}
  \inheritanchor[from=rectangle]{north west}
  \inheritanchor[from=rectangle]{north east}
  \inheritanchor[from=rectangle]{center}
  \inheritanchor[from=rectangle]{west}
  \inheritanchor[from=rectangle]{east}
  \inheritanchor[from=rectangle]{mid}
  \inheritanchor[from=rectangle]{mid west}
  \inheritanchor[from=rectangle]{mid east}
  \inheritanchor[from=rectangle]{base}
  \inheritanchor[from=rectangle]{base west}
  \inheritanchor[from=rectangle]{base east}
  \inheritanchor[from=rectangle]{south}
  \inheritanchor[from=rectangle]{south west}
  \inheritanchor[from=rectangle]{south east}
  \inheritanchorborder[from=rectangle]
  \backgroundpath{
    \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
    \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
    \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
    \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
  }
  \foregroundpath{
% store lower right in xa/ya and upper right in xb/yb
```

```
    \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
    \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
    \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
    \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
 }
}
```

knotundercross

```
\pgfdeclareshape{knot under cross}
{
  \inheritsavedanchors[from=rectangle] % this is nearly a circle
  \inheritanchorborder[from=rectangle]
  \inheritanchor[from=rectangle]{north}
  \inheritanchor[from=rectangle]{north west}
  \inheritanchor[from=rectangle]{north east}
  \inheritanchor[from=rectangle]{center}
  \inheritanchor[from=rectangle]{west}
  \inheritanchor[from=rectangle]{east}
  \inheritanchor[from=rectangle]{mid}
  \inheritanchor[from=rectangle]{mid west}
  \inheritanchor[from=rectangle]{mid east}
  \inheritanchor[from=rectangle]{base}
  \inheritanchor[from=rectangle]{base west}
  \inheritanchor[from=rectangle]{base east}
  \inheritanchor[from=rectangle]{south}
  \inheritanchor[from=rectangle]{south west}
  \inheritanchor[from=rectangle]{south east}
  \inheritanchorborder[from=rectangle]
  \backgroundpath{
    \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
    \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
    \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
    \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
  }
  \foregroundpath{
% store lower right in xa/ya and upper right in xb/yb
    \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
    \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
    \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
    \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
 }
}
```

knotvert

```
\pgfdeclareshape{knot vert}
{
  \inheritsavedanchors[from=rectangle] % this is nearly a circle
  \inheritanchorborder[from=rectangle]
  \inheritanchor[from=rectangle]{north}
```

```
  \inheritanchor[from=rectangle]{north west}
  \inheritanchor[from=rectangle]{north east}
  \inheritanchor[from=rectangle]{center}
  \inheritanchor[from=rectangle]{west}
  \inheritanchor[from=rectangle]{east}
  \inheritanchor[from=rectangle]{mid}
  \inheritanchor[from=rectangle]{mid west}
  \inheritanchor[from=rectangle]{mid east}
  \inheritanchor[from=rectangle]{base}
  \inheritanchor[from=rectangle]{base west}
  \inheritanchor[from=rectangle]{base east}
  \inheritanchor[from=rectangle]{south}
  \inheritanchor[from=rectangle]{south west}
  \inheritanchor[from=rectangle]{south east}
  \inheritanchorborder[from=rectangle]
  \backgroundpath{
% store lower right in xa/ya and upper right in xb/yb
    \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
    \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
    \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
    \pgfpathlineto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
    \pgfpathmoveto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
    \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
 }
}
```

knothoriz

```
\pgfdeclareshape{knot horiz}
{
  \inheritsavedanchors[from=rectangle] % this is nearly a circle
  \inheritanchorborder[from=rectangle]
  \inheritanchor[from=rectangle]{north}
  \inheritanchor[from=rectangle]{north west}
  \inheritanchor[from=rectangle]{north east}
  \inheritanchor[from=rectangle]{center}
  \inheritanchor[from=rectangle]{west}
  \inheritanchor[from=rectangle]{east}
  \inheritanchor[from=rectangle]{mid}
  \inheritanchor[from=rectangle]{mid west}
  \inheritanchor[from=rectangle]{mid east}
  \inheritanchor[from=rectangle]{base}
  \inheritanchor[from=rectangle]{base west}
  \inheritanchor[from=rectangle]{base east}
  \inheritanchor[from=rectangle]{south}
  \inheritanchor[from=rectangle]{south west}
  \inheritanchor[from=rectangle]{south east}
  \inheritanchorborder[from=rectangle]
  \foregroundpath{
% store lower right in xa/ya and upper right in xb/yb
    \southwest \pgf@xa=\pgf@x \pgf@ya=\pgf@y
```

```
    \northeast \pgf@xb=\pgf@x \pgf@yb=\pgf@y
    \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@ya}}
    \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@ya}}
    \pgfpathmoveto{\pgfqpoint{\pgf@xa}{\pgf@yb}}
    \pgfpathlineto{\pgfqpoint{\pgf@xb}{\pgf@yb}}
  }
}
```