The **spath3** package: code

Andrew Stacey
loopspace@mathforge.org

v2.8 from 2024/05/31

1 Introduction

The `spath3` package is intended as a library for manipulating PGF's *soft paths*. In between defining a path and using it, PGF stores a path as a *soft path* where all the defining structure has been resolved into the basic operations but these have not yet been written to the output file. They can therefore still be manipulated by TEX, and as they have a very rigid form (and limited vocabulary), they are relatively easy to modify. This package provides some methods for working with these paths. It was originally not really intended for use by end users but as a foundation on which other packages can be built. However, over the years I've found myself using it at ever higher levels and so a set of interfaces has been designed using TikZ keys.

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

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

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

2 Technical Details

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

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

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

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

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

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

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

3 Implementation

3.1 Initialisation

```
⟨@@=spath⟩
```

Load the LaTeX3 foundation and register us as a LaTeX3 package.

```
\NeedsTeXFormat{LaTeX2e}
\RequirePackage{expl3}
\RequirePackage{pgf}
\ProvidesExplPackage {spath3} {2024/05/31} {2.8} {Functions for
manipulating PGF soft paths}
\RequirePackage{xparse}
```

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

```
\cs_new_protected:Nn \__spath_tl_put_right_braced:Nn
{
  \tl_put_right:Nn #1 { { #2 } }
}
\cs_generate_variant:Nn \__spath_tl_put_right_braced:Nn { NV, cV, cv, Nx, cx }

\cs_new_protected:Nn \__spath_tl_gput_right_braced:Nn
{
  \tl_gput_right:Nn #1 { { #2 } }
}
\cs_generate_variant:Nn \__spath_tl_gput_right_braced:Nn { NV, cV, cv, Nx, cx }
\cs_new_protected:Nn \__spath_tl_put_left_braced:Nn
{
  \tl_put_left:Nn #1 { { #2 } }
}
\cs_generate_variant:Nn \__spath_tl_put_left_braced:Nn { NV, cV, cv, Nx, cx }

\cs_new_protected:Nn \__spath_tl_gput_left_braced:Nn
{
  \tl_gput_left:Nn #1 { { #2 } }
}
\cs_generate_variant:Nn \__spath_tl_gput_left_braced:Nn { NV, cV, cv, Nx, cx }
```

I had to think a bit about how to get TEX to work the way I wanted. I'm really defining *functions* but TEX doesn't really have that concept, even with all the amazing LATEX3 stuff. The main issue I had was with scoping and return values. By default, TEX functions aren't scoped – they work on the same level as the calling functions. To protect the internals from being overwritten, each core function works inside a group. But then I have to work to get the answer out of it. So each of my core functions finishes by storing its return value in an appropriate `output` variable. The core functions are then wrapped in a more user friendly interface that will take that output and assign it to a variable. This also means that I can deal with local and global versions without duplicating code.

```
\tl_new:N \g__spath_output_tl
\int_new:N \g__spath_output_int
\seq_new:N \g__spath_output_seq
\bool_new:N \g__spath_output_bool
```

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

These are general purpose variables.

```
\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_tmpe_tl
\tl_new:N \l__spath_tmpf_tl
\tl_new:N \l__spath_tmpg_tl
\tl_new:N \l__spath_tmph_tl
\tl_new:N \l__spath_tmpi_tl

\seq_new:N \l__spath_tmpa_seq
\seq_new:N \l__spath_tmpb_seq
\seq_new:N \l__spath_tmpc_seq

\dim_new:N \l__spath_tmpa_dim
\dim_new:N \l__spath_tmpb_dim

\fp_new:N \l__spath_tmpa_fp
\fp_new:N \l__spath_tmpb_fp
\fp_new:N \l__spath_tmpc_fp
\fp_new:N \l__spath_tmpd_fp
\fp_new:N \l__spath_tmpe_fp
\fp_new:N \l__spath_tmpf_fp

\int_new:N \l__spath_tmpa_int
\int_new:N \l__spath_tmpb_int

\bool_new:N \l__spath_tmpa_bool
```

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

```
\dim_new:N \l__spath_move_x_dim
\dim_new:N \l__spath_move_y_dim
```

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

```
\bool_new:N \l__spath_closed_bool
```

True rectangles are rare, but need special handling. They are specified by two tokens, the first specifies the lower left corner which can be handled pretty much as other tokens but the second specifies the width and height meaning that it transforms differently. So when encountering on, the coordinates of the lower left corner are useful to remember.

```
\dim_new:N \l__spath_rectx_dim
\dim_new:N \l__spath_recty_dim

\bool_new:N \l__spath_rect_bool
```

When restoring a path we need to know whether to update the stored moveto.

```
\bool_new:N \l_spath_movetorelevant_bool
```

When manipulating soft paths, we might need to separate the shortening due to an arrow from when the path is rendered.

```
\bool_new:N \l_spath_arrow_shortening_bool
\bool_set_true:N \l_spath_arrow_shortening_bool
```

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

```
\tl_new:N \l__spath_intersecta_tl
\tl_new:N \l__spath_intersectb_tl
```

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

```
\tl_const:Nn \c_spath_moveto_tl {\pgfsyssoftpath@movetotoken}
\tl_const:Nn \c_spath_lineto_tl {\pgfsyssoftpath@linetotoken}
\tl_const:Nn \c_spath_curveto_tl {\pgfsyssoftpath@curvetotoken}
\tl_const:Nn \c_spath_curvetoa_tl {\pgfsyssoftpath@curvetosupportatoken}
\tl_const:Nn \c_spath_curvetob_tl {\pgfsyssoftpath@curvetosupportbtoken}
\tl_const:Nn \c_spath_closepath_tl {\pgfsyssoftpath@closepathtoken}
\tl_const:Nn \c_spath_rectcorner_tl {\pgfsyssoftpath@rectcornertoken}
\tl_const:Nn \c_spath_rectsize_tl {\pgfsyssoftpath@rectsizetoken}
```

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

```
\int_new:N \g__spath_anon_int
\int_gzero:N \g__spath_anon_int
```

\spath_anonymous:N
\spath_ganonymous:N

Set the token list to the next anonymous name.

```
\cs_new_protected_nopar:Npn \spath_anonymous:N #1
{
  \tl_set:Nx #1 {anonymous_\int_use:N \g__spath_anon_int}
  \int_gincr:N \g__spath_anon_int
}
\cs_new_protected_nopar:Npn \spath_ganonymous:N #1
{
  \tl_gset:Nx #1 {anonymous_\int_use:N \g__spath_anon_int}
  \int_gincr:N \g__spath_anon_int
}
\cs_generate_variant:Nn \spath_anonymous:N {c}
\cs_generate_variant:Nn \spath_ganonymous:N {c}
```

(*End of definition for* `\spath_anonymous:N` *and* `\spath_ganonymous:N`*.*)

And some error messages

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

3.2 Functional Implementation

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

`\spath_segments_to_seq:Nn`
`\spath_segments_gto_seq:Nn`

Splits a soft path into *segments*, storing the result in a sequence.

```
\cs_new_protected_nopar:Npn \__spath_segments_to_seq:n #1
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \tl_clear:N \l__spath_tmpb_tl
  \seq_clear:N \l__spath_tmpa_seq
  \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}
    \token_case_meaning:NnF \l__spath_tmpc_tl
    {
      \c_spath_moveto_tl
      {
        \tl_set_eq:NN \l__spath_tmpb_tl \c_spath_moveto_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:Nx \l__spath_tmpd_tl {\tl_head:N \l__spath_tmpa_tl}
        \tl_if_eq:NNF \l__spath_tmpd_tl \c_spath_moveto_tl
        {
          \tl_clear:N \l__spath_tmpb_tl
        }

      }

      \c_spath_lineto_tl
      {
        \tl_set_eq:NN \l__spath_tmpb_tl \c_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 \c_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}

      }

      \c_spath_curvetoa_tl
      {
        \tl_set_eq:NN \l__spath_tmpb_tl \c_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 \c_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}

      }

      \c_spath_rectcorner_tl
      {
        \tl_set_eq:NN \l__spath_tmpb_tl \c_spath_rectcorner_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}
        \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}

      }

      \c_spath_closepath_tl
      {
        \tl_set_eq:NN \l__spath_tmpb_tl \c_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 \c_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}

      }

    }
    {

      \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}

    }

    \tl_if_empty:NF \l__spath_tmpb_tl
    {
      \seq_put_right:NV \l__spath_tmpa_seq \l__spath_tmpb_tl
    }
    \tl_clear:N \l__spath_tmpb_tl
  }

  \seq_gclear:N \g__spath_output_seq
  \seq_gset_eq:NN \g__spath_output_seq \l__spath_tmpa_seq
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_segments_to_seq:Nn #1#2
{
  \__spath_segments_to_seq:n {#2}
  \seq_clear_new:N #1
  \seq_set_eq:NN #1 \g__spath_output_seq
  \seq_gclear:N \g__spath_output_seq
}
\cs_generate_variant:Nn \spath_segments_to_seq:Nn {NV, cn, cV, Nv, cv}
\cs_new_protected_nopar:Npn \spath_segments_gto_seq:Nn #1#2
{
  \__spath_segments_to_seq:n {#2}
  \seq_clear_new:N #1
  \seq_gset_eq:NN #1 \g__spath_output_seq
  \seq_gclear:N \g__spath_output_seq
}
\cs_generate_variant:Nn \spath_segments_gto_seq:Nn {NV, cn, cV, Nv, cv}
```

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

`\spath_components_to_seq:Nn`
`\spath_components_gto_seq:Nn`
`\spath_components_to_clist:Nn`
`\spath_components_gto_clist:Nn`

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

```
\cs_new_protected_nopar:Npn \__spath_components_to_seq:n #1
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \seq_clear:N \l__spath_tmpa_seq
  \tl_set: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:NV \l__spath_tmpa_tl \c_spath_moveto_tl

  \bool_do_until: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 \c_spath_moveto_tl
    {
      \seq_put_right:NV \l__spath_tmpa_seq \l__spath_tmpb_tl
      \tl_clear:N \l__spath_tmpb_tl
    }
    \tl_if_eq:NNT \l__spath_tmpc_tl \c_spath_rectcorner_tl
    {
      \seq_put_right:NV \l__spath_tmpa_seq \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}
  }

  \seq_gclear:N \g__spath_output_seq
  \seq_gset_eq:NN \g__spath_output_seq \l__spath_tmpa_seq
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_components_to_seq:Nn #1#2
{
  \__spath_components_to_seq:n {#2}
  \seq_clear_new:N #1
  \seq_set_eq:NN #1 \g__spath_output_seq
  \seq_gclear:N \g__spath_output_seq
}
\cs_generate_variant:Nn \spath_components_to_seq:Nn {NV, cn, cV, cv, Nv}
\cs_new_protected_nopar:Npn \spath_components_gto_seq:Nn #1#2
{
  \__spath_components_to_seq:n {#2}
  \seq_clear_new:N #1
  \seq_gset_eq:NN #1 \g__spath_output_seq
  \seq_gclear:N \g__spath_output_seq
}
\cs_generate_variant:Nn \spath_components_gto_seq:Nn {NV, cn, cV, cv, Nv}
\cs_new_protected_nopar:Npn \spath_components_to_clist:Nn #1#2
{
  \__spath_components_to_seq:n {#2}
  \clist_clear_new:N #1
  \clist_set_from_seq:NN #1 \g__spath_output_seq
  \seq_gclear:N \g__spath_output_seq
}
\cs_generate_variant:Nn \spath_components_to_clist:Nn {NV, cn, cV, cv, Nv}
\cs_new_protected_nopar:Npn \spath_components_gto_clist:Nn #1#2
{
  \__spath_components_to_seq:n {#2}
  \clist_clear_new:N #1
  \clist_gset_from_seq:NN #1 \g__spath_output_seq
  \seq_gclear:N \g__spath_output_seq
}
\cs_generate_variant:Nn \spath_components_gto_clist:Nn {NV, cn, cV, cv, Nv}
```

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

`\spath_length:n`

Counts the number of triples in the path.

```
\cs_new_protected_nopar:Npn \spath_length:n #1
{
  \int_eval:n {\tl_count:n {#1} / 3}
}
\cs_generate_variant:Nn \spath_length:n {V}
```

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

`\spath_reallength:Nn`
`\spath_greallength:Nn`

The real length of a path is the number of triples that actually draw something (that is, the number of lines, curves, rectangles, and closepaths).

```
\cs_new_protected_nopar:Npn \__spath_reallength:n #1
{
  \group_begin:
  \int_set:Nn \l__spath_tmpa_int {0}
  \tl_map_inline:nn {#1} {
    \tl_set:Nn \l__spath_tmpa_tl {##1}
    \token_case_meaning:NnT \l__spath_tmpa_tl
    {
      \c_spath_lineto_tl {}
      \c_spath_curveto_tl {}
      \c_spath_closepath_tl {}
      \c_spath_rectsize_tl {}
    }
    {
      \int_incr:N \l__spath_tmpa_int
    }
  }
  \int_gzero:N \g__spath_output_int
  \int_gset_eq:NN \g__spath_output_int \l__spath_tmpa_int
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_reallength:Nn #1#2
{
  \__spath_reallength:n {#2}
  \int_set_eq:NN #1 \g__spath_output_int
  \int_gzero:N \g__spath_output_int
}
\cs_generate_variant:Nn \spath_reallength:Nn {NV, cn, cV, Nv, cv}
\cs_new_protected_nopar:Npn \spath_greallength:Nn #1#2
{
  \__spath_reallength:n {#2}
  \int_gset_eq:NN #1 \g__spath_output_int
  \int_gzero:N \g__spath_output_int
}
\cs_generate_variant:Nn \spath_greallength:Nn {NV, cn, cV}
```

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

`\spath_numberofcomponents:Nn`
`\spath_gnumberofcomponents:Nn`

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

```
\cs_new_protected_nopar:Npn \__spath_numberofcomponents:n #1
{
  \group_begin:
  \int_set:Nn \l__spath_tmpa_int {0}
  \tl_map_inline:nn {#1} {
    \tl_set:Nn \l__spath_tmpa_tl {##1}
    \token_case_meaning:Nn \l__spath_tmpa_tl
    {
      \c_spath_moveto_tl
      {
        \int_incr:N \l__spath_tmpa_int
      }
      \c_spath_rectcorner_tl
      {
        \int_incr:N \l__spath_tmpa_int
      }
    }
  }
  \int_gzero:N \g__spath_output_int
  \int_gset_eq:NN \g__spath_output_int \l__spath_tmpa_int
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_numberofcomponents:Nn #1#2
{
  \__spath_numberofcomponents:n {#2}
  \int_set_eq:NN #1 \g__spath_output_int
  \int_gzero:N \g__spath_output_int
}
\cs_generate_variant:Nn \spath_numberofcomponents:Nn {NV, cn, cV, Nv}
\cs_new_protected_nopar:Npn \spath_gnumberofcomponents:Nn #1#2
{
  \__spath_numberofcomponents:n {#2}
  \int_gset_eq:NN #1 \g__spath_output_int
  \int_gzero:N \g__spath_output_int
}
\cs_generate_variant:Nn \spath_gnumberofcomponents:Nn {NV, cn, cV, Nv}
```

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

`\spath_initialpoint:Nn`
`\spath_ginitialpoint:Nn`

The starting point of the path.

```
\cs_new_protected_nopar:Npn \__spath_initialpoint:n #1
{
  \group_begin:
  \tl_clear:N \l__spath_tmpa_tl
  \tl_set:Nx \l__spath_tmpa_tl
  {
    { \tl_item:nn {#1} {2} }
    { \tl_item:nn {#1} {3} }
  }
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_initialpoint:Nn #1#2
{
  \__spath_initialpoint:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_initialpoint:Nn {NV, cn, cV, Nv}
\cs_new_protected_nopar:Npn \spath_ginitialpoint:Nn #1#2
{
  \__spath_initialpoint:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_ginitialpoint:Nn {NV, cn, cV, Nv}
```

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

`\spath_finalpoint:Nn`
`\spath_gfinalpoint:Nn`

The final point of the path.

```
\cs_new_protected_nopar:Npn \__spath_finalpoint:n #1
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \tl_reverse:N \l__spath_tmpa_tl
  \tl_clear:N \l__spath_tmpb_tl
  \tl_set:Nx \l__spath_tmpc_tl {\tl_item:Nn \l__spath_tmpa_tl {3}}
  \tl_if_eq:NNTF \l__spath_tmpc_tl \c_spath_rectsize_tl
  {
    \tl_set:Nx \l__spath_tmpb_tl
    {
      {
        \dim_eval:n
        {
          \tl_item:Nn \l__spath_tmpa_tl {2}
          +
          \tl_item:Nn \l__spath_tmpa_tl {5}
        }
      }
      {
        \dim_eval:n
        {
          \tl_item:Nn \l__spath_tmpa_tl {1}
          +
          \tl_item:Nn \l__spath_tmpa_tl {4}
        }
      }
    }
  }
  {
    \tl_set:Nx \l__spath_tmpb_tl
    {
      { \tl_item:Nn \l__spath_tmpa_tl {2} }
      { \tl_item:Nn \l__spath_tmpa_tl {1} }
    }
  }
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_finalpoint:Nn #1#2
{
  \__spath_finalpoint:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_finalpoint:Nn {NV, cn, cV, Nv}
\cs_new_protected_nopar:Npn \spath_gfinalpoint:Nn #1#2
{
  \__spath_finalpoint:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gfinalpoint:Nn {NV, cn, cV, Nv}
```

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

`\spath_finalmovepoint:Nn`
`\spath_gfinalmovepoint:Nn`

Get the last move on the path.

```
\cs_new_protected_nopar:Npn \__spath_finalmovepoint:n #1
```

```
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpc_tl { {0pt} {0pt} }
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \bool_do_until:nn
  {
    \tl_if_empty_p:N \l__spath_tmpa_tl
  }
  {
    \tl_set:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
    \token_case_meaning:Nn \l__spath_tmpb_tl
    {
      \c_spath_moveto_tl
      {
        \tl_set:Nx \l__spath_tmpc_tl
        {
          { \tl_item:Nn \l__spath_tmpa_tl {2} }
          { \tl_item:Nn \l__spath_tmpa_tl {3} }
        }
      }

      \c_spath_rectcorner_tl
      {
        \tl_set:Nx \l__spath_tmpc_tl
        {
          { \tl_item:Nn \l__spath_tmpa_tl {2} }
          { \tl_item:Nn \l__spath_tmpa_tl {3} }
        }
      }

    }
    \prg_replicate:nn {3}
    {
      \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
    }
  }
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpc_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_finalmovepoint:Nn #1#2
{
  \__spath_finalmovepoint:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_finalmovepoint:Nn {NV, cn, cV}
\cs_new_protected_nopar:Npn \spath_gfinalmovepoint:Nn #1#2
{
  \__spath_finalmovepoint:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gfinalmovepoint:Nn {NV, cn, cV}
```

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

\spath_initialtangent:Nn
\spath_ginitialtangent:Nn

The starting tangent of the path.

```
\cs_new_protected_nopar:Npn \__spath_initialtangent:n #1
{
  \group_begin:
  \tl_set:Nx \l__spath_tmpb_tl {\tl_item:nn {#1} {4}}
  \tl_if_eq:NNTF \l__spath_tmpb_tl \c_spath_curvetoa_tl
  {
    \fp_set:Nn \l__spath_tmpa_fp {3}
  }
  {
    \fp_set:Nn \l__spath_tmpa_fp {1}
  }
  \tl_clear:N \l__spath_tmpa_tl
  \tl_set:Nx \l__spath_tmpa_tl
  {
    {
      \fp_to_dim:n {
        \l__spath_tmpa_fp
        *
        (
        \tl_item:nn {#1} {5}
        -
        \tl_item:nn {#1} {2}
        )
      }
    }
    {
      \fp_to_dim:n {
        \l__spath_tmpa_fp
        *
        (
        \tl_item:nn {#1} {6}
        -
        \tl_item:nn {#1} {3}
        )
      }
    }
  }
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_initialtangent:Nn #1#2
{
  \__spath_initialtangent:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_initialtangent:Nn {NV, cn, cV, Nv}
\cs_new_protected_nopar:Npn \spath_ginitialtangent:Nn #1#2
{
  \__spath_initialtangent:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
```

```
\cs_generate_variant:Nn \spath_ginitialtangent:Nn {NV, cn, cV, Nv}
```

(*End of definition for* `\spath_initialtangent:Nn` *and* `\spath_ginitialtangent:Nn`*.*)

`\spath_finaltangent:Nn`
`\spath_gfinaltangent:Nn`

The final tangent of the path.

```
\cs_new_protected_nopar:Npn \__spath_finaltangent:n #1
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \tl_reverse:N \l__spath_tmpa_tl
  \tl_set:Nx \l__spath_tmpb_tl {\tl_item:nn {#1} {6}}
  \tl_if_eq:NNTF \l__spath_tmpb_tl \c_spath_curveto_tl
  {
    \fp_set:Nn \l__spath_tmpa_fp {3}
  }
  {
    \fp_set:Nn \l__spath_tmpa_fp {1}
  }
  \tl_clear:N \l__spath_tmpb_tl
  \tl_set:Nx \l__spath_tmpb_tl
  {
    {
      \fp_to_dim:n {
        \l__spath_tmpa_fp
        *
        (
        \tl_item:Nn \l__spath_tmpa_tl {2}
        -
        \tl_item:Nn \l__spath_tmpa_tl {5}
        )
      }
    }
    {
      \fp_to_dim:n {
        \l__spath_tmpa_fp
        *
        (
        \tl_item:Nn \l__spath_tmpa_tl {1}
        -
        \tl_item:Nn \l__spath_tmpa_tl {4}
        )
      }
    }
  }
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_finaltangent:Nn #1#2
{
  \__spath_finaltangent:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_finaltangent:Nn {NV, cn, cV, Nv}
```

```
\cs_new_protected_nopar:Npn \spath_gfinaltangent:Nn #1#2
{
  \__spath_finaltangent:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gfinaltangent:Nn {NV, cn, cV, Nv}
```

(*End of definition for* `\spath_finaltangent:Nn` *and* `\spath_gfinaltangent:Nn`.)

`\spath_finalmovetangent:Nn`
`\spath_gfinalmovetangent:Nn`

Get the last move on the path.

```
\cs_new_protected_nopar:Npn \__spath_finalmovetangent:n #1
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpc_tl { {0pt} {0pt} }
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \bool_do_until:nn
  {
    \tl_if_empty_p:N \l__spath_tmpa_tl
  }
  {
    \tl_set:Nx \l__spath_tmpb_tl {\tl_head:N \l__spath_tmpa_tl}
    \token_case_meaning:Nn \l__spath_tmpb_tl
    {
      \c_spath_moveto_tl
      {
        \tl_set:Nx \l__spath_tmpd_tl { \tl_item:Nn \l__spath_tmpa_tl {4} }
        \tl_if_eq:NNTF \l__spath_tmpd_tl \c_spath_curveto_tl
        {
          \fp_set:Nn \l__spath_tmpa_fp {3}
        }
        {
          \fp_set:Nn \l__spath_tmpa_fp {1}
        }
        \tl_set:Nx \l__spath_tmpc_tl
        {
          {
            \fp_to_dim:n
            {
              \l__spath_tmpa_fp
              *
              (
              \tl_item:Nn \l__spath_tmpa_tl {5}
              -
              \tl_item:Nn \l__spath_tmpa_tl {2}
              )
            }
          }
          {
            \fp_to_dim:n
            {
              \l__spath_tmpa_fp
              *
              (
              \tl_item:Nn \l__spath_tmpa_tl {6}
              -
              \tl_item:Nn \l__spath_tmpa_tl {3}
              )
            }
          }
        }
      }
    }
    \prg_replicate:nn {3}
    {
      \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
    }
  }
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpc_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_finalmovetangent:Nn #1#2
{
  \__spath_finalmovetangent:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_finalmovetangent:Nn {NV, cn, cV}
\cs_new_protected_nopar:Npn \spath_gfinalmovetangent:Nn #1#2
{
  \__spath_finalmovetangent:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gfinalmovetangent:Nn {NV, cn, cV}
```

(*End of definition for* `\spath_finalmovetangent:Nn` *and* `\spath_gfinalmovetangent:Nn`.)

`\spath_reverse:Nn`
`\spath_greverse:Nn`

This computes the reverse of the path.

```
\cs_new_protected_nopar:Npn \__spath_reverse:n #1
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}

  \tl_clear:N \l__spath_tmpb_tl
  \tl_clear:N \l__spath_tmpd_tl

  \bool_set_false:N \l__spath_rect_bool

  \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
  \bool_while_do:nn {
    \tl_if_eq_p:NN \l__spath_tmpc_tl \c_spath_rectcorner_tl
  }
  {
    \tl_put_left:Nx \l__spath_tmpd_tl
    {
      \tl_item:Nn \l__spath_tmpa_tl {1}
      {\tl_item:Nn \l__spath_tmpa_tl {2}}
      {\tl_item:Nn \l__spath_tmpa_tl {3}}
      \tl_item:Nn \l__spath_tmpa_tl {4}
      {\tl_item:Nn \l__spath_tmpa_tl {5}}
      {\tl_item:Nn \l__spath_tmpa_tl {6}}
    }
    \prg_replicate:nn {6}
    {
      \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
    }
    \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
    \bool_set_true:N \l__spath_rect_bool
  }

  \tl_if_empty:NF \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}

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

    \bool_set_false:N \l__spath_closed_bool

    \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}
      \bool_set_false:N \l__spath_rect_bool

      \token_case_meaning:NnTF \l__spath_tmpc_tl
      {
        \c_spath_moveto_tl {

          \bool_if:NT \l__spath_closed_bool
          {
            \tl_put_right:NV \l__spath_tmpd_tl \c_spath_closepath_tl
            \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpd_tl}
            \tl_put_right:Nx \l__spath_tmpd_tl
            {
              { \tl_head:N \l__spath_tmpd_tl }
              { \tl_head:N \l__spath_tmpe_tl }
            }
          }
          \bool_set_false:N \l__spath_closed_bool
          \tl_put_left:NV \l__spath_tmpd_tl \c_spath_moveto_tl
          \tl_put_left:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
          \tl_clear:N \l__spath_tmpd_tl
        }
        \c_spath_rectcorner_tl {

          \bool_if:NT \l__spath_closed_bool
          {
            \tl_put_right:NV \l__spath_tmpd_tl \c_spath_closepath_tl
            \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpd_tl}
            \tl_put_right:Nx \l__spath_tmpd_tl
            {
              { \tl_head:N \l__spath_tmpd_tl }
              { \tl_head:N \l__spath_tmpe_tl }
            }
          }
          \bool_set_false:N \l__spath_closed_bool
          \tl_put_left:NV \l__spath_tmpd_tl \c_spath_moveto_tl
          \tl_put_left:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
          \tl_clear:N \l__spath_tmpd_tl

          \bool_while_do:nn {
            \tl_if_eq_p:NN \l__spath_tmpc_tl \c_spath_rectcorner_tl
          }
          {
            \tl_put_left:Nx \l__spath_tmpb_tl
            {
              \tl_item:Nn \l__spath_tmpa_tl {1}
              {\tl_item:Nn \l__spath_tmpa_tl {2}}
              {\tl_item:Nn \l__spath_tmpa_tl {3}}
              \tl_item:Nn \l__spath_tmpa_tl {4}
              {\tl_item:Nn \l__spath_tmpa_tl {5}}
              {\tl_item:Nn \l__spath_tmpa_tl {6}}
            }
            \prg_replicate:nn {6}
            {
              \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
            }
            \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
          }
          \bool_set_true:N \l__spath_rect_bool

        }
        \c_spath_lineto_tl {
          \tl_put_left:NV \l__spath_tmpd_tl \c_spath_lineto_tl
        }
        \c_spath_curveto_tl {
          \tl_put_left:NV \l__spath_tmpd_tl \c_spath_curvetoa_tl
        }
        \c_spath_curvetoa_tl {
          \tl_put_left:NV \l__spath_tmpd_tl \c_spath_curveto_tl
        }
        \c_spath_curvetob_tl {
          \tl_put_left:NV \l__spath_tmpd_tl \c_spath_curvetob_tl
        }
      }
      {
        \tl_if_empty:NF \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}

        \tl_put_left:Nx \l__spath_tmpd_tl
        {
          {\dim_use:N \l__spath_tmpa_dim}
          {\dim_use:N \l__spath_tmpb_dim}
        }
        }
      }
      {
        \tl_if_eq:NNTF \l__spath_tmpc_tl \c_spath_closepath_tl
        {
          \bool_set_true:N \l__spath_closed_bool
        }
        {
          \msg_warning:nnx
          { spath3 }
          { unknown path construction }
          { \l__spath_tmpc_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}

      }
    }

    \bool_if:NT \l__spath_closed_bool
    {
      \tl_put_right:NV \l__spath_tmpd_tl \c_spath_closepath_tl
      \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpd_tl}
      \tl_put_right:Nx \l__spath_tmpd_tl
      {
        { \tl_head:N \l__spath_tmpd_tl }
        { \tl_head:N \l__spath_tmpe_tl }
      }
    }

    \bool_set_false:N \l__spath_closed_bool
    \bool_if:NF \l__spath_rect_bool
    {
      \tl_put_left:NV \l__spath_tmpd_tl \c_spath_moveto_tl
    }
  }

  \tl_put_left:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
```

```
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_reverse:Nn #1#2
{
  \__spath_reverse:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_reverse:Nn {NV, cn, cV, Nv}
\cs_new_protected_nopar:Npn \spath_reverse:N #1
{
  \spath_reverse:NV #1#1
}
\cs_generate_variant:Nn \spath_reverse:N {c}
\cs_new_protected_nopar:Npn \spath_greverse:Nn #1#2
{
  \__spath_reverse:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_greverse:Nn {NV, cn, cV, Nv}
\cs_new_protected_nopar:Npn \spath_greverse:N #1
{
  \spath_greverse:NV #1#1
}
\cs_generate_variant:Nn \spath_greverse:N {c}
```

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

`\spath_initialaction:Nn`
`\spath_ginitialaction:Nn`

This is the first thing that the path does (after the initial move).

```
\cs_new_protected_nopar:Npn \__spath_initialaction:n #1
{
  \group_begin:
  \tl_clear:N \l__spath_tmpa_tl
  \tl_set:Nx \l__spath_tmpb_tl {\tl_head:n {#1}}
  \tl_if_eq:NNTF \l__spath_tmpb_tl \c_spath_rectcorner_tl
  {
    \tl_set_eq:NN \l__spath_tmpa_tl \c_spath_rectcorner_tl
  }
  {
    \int_compare:nT
    {
      \tl_count:n {#1} > 3
    }
    {
      \tl_set:Nx \l__spath_tmpa_tl
      {
        \tl_item:Nn {#1} {4}
      }
    }
  }
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_initialaction:Nn #1#2
{
  \__spath_initialaction:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_initialaction:Nn {NV}
\cs_new_protected_nopar:Npn \spath_ginitialaction:Nn #1#2
{
  \__spath_initialaction:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_ginitialaction:Nn {NV}
```

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

`\spath_finalaction:Nn`
`\spath_gfinalaction:Nn`

This is the last thing that the path does.

```
\cs_new_protected_nopar:Npn \__spath_finalaction:n #1
{
  \group_begin:
  \tl_clear:N \l__spath_tmpb_tl
  \int_compare:nT
  {
    \tl_count:n {#1} > 3
  }
  {
    \tl_set:Nn \l__spath_tmpa_tl {#1}
    \tl_reverse:N \l__spath_tmpa_tl
    \tl_set:Nx \l__spath_tmpb_tl
    {
      \tl_item:Nn \l__spath_tmpa_tl {3}
    }
    \tl_if_eq:NNT \l__spath_tmpb_tl \c_spath_curvetoa_tl
    {
      \tl_set_eq:NN \l__spath_tmpb_tl \c_spath_curveto_tl
    }
    \tl_if_eq:NNT \l__spath_tmpb_tl \c_spath_rectsize_tl
    {
      \tl_set_eq:NN \l__spath_tmpb_tl \c_spath_rectcorner_tl
    }
  }
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_finalaction:Nn #1#2
{
  \__spath_finalaction:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_finalaction:Nn {NV}
\cs_new_protected_nopar:Npn \spath_gfinalaction:Nn #1#2
```

```
{
  \__spath_finalaction:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gfinalaction:Nn {NV}
```

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

`\spath_minbb:Nn`
`\spath_gminbb:Nn`

This computes the minimum (bottom left) of the bounding box of the path.

```
\cs_new_protected_nopar:Npn \__spath_minbb:n #1
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}

  \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}

  \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_if_eq:NNT \l__spath_tmpc_tl \c_spath_rectcorner_tl
  {
    \dim_set_eq:NN \l__spath_rectx_dim \l__spath_tmpa_dim
    \dim_set_eq:NN \l__spath_recty_dim \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_if_eq:NNTF \l__spath_tmpc_tl \c_spath_rectsize_tl
    {
      \dim_set:Nn \l__spath_tmpa_dim
      {\dim_min:nn
        {\tl_head:N \l__spath_tmpa_tl + \l__spath_rectx_dim} {\l__spath_tmpa_dim}
      }
      \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_recty_dim} {\l__spath_tmpb_dim}
      }
      \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_dim}}
      \tl_if_eq:NNT \l__spath_tmpc_tl \c_spath_rectcorner_tl
      {
        \dim_set:Nn \l__spath_rectx_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
      {\dim_min:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tmpb_dim}}
      \tl_if_eq:NNT \l__spath_tmpc_tl \c_spath_rectcorner_tl
      {
        \dim_set:Nn \l__spath_recty_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_tmpb_tl
  \tl_put_right:Nx \l__spath_tmpb_tl
  {
    {\dim_use:N \l__spath_tmpa_dim}
    {\dim_use:N \l__spath_tmpb_dim}
  }
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_minbb:Nn #1#2
{
  \__spath_minbb:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_minbb:Nn {NV, cn, cV}
\cs_new_protected_nopar:Npn \spath_gminbb:Nn #1#2
{
  \__spath_minbb:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gminbb:Nn {NV, cn, cV}
```

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

`\spath_maxbb:Nn`
`\spath_gmaxbb:Nn`

This computes the maximum (top right) of the bounding box of the path.

```
\cs_new_protected_nopar:Npn \__spath_maxbb:n #1
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}

  \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}

  \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_if_eq:NNT \l__spath_tmpc_tl \c_spath_rectcorner_tl
{
  \dim_set_eq:NN \l__spath_rectx_dim \l__spath_tmpa_dim
  \dim_set_eq:NN \l__spath_recty_dim \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_if_eq:NNTF \l__spath_tmpc_tl \c_spath_rectsize_tl
  {
    \dim_set:Nn \l__spath_tmpa_dim
    {\dim_max:nn
      {\tl_head:N \l__spath_tmpa_tl + \l__spath_rectx_dim} {\l__spath_tmpa_dim}
    }
    \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_recty_dim} {\l__spath_tmpb_dim}
    }
    \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_dim}}
    \tl_if_eq:NNT \l__spath_tmpc_tl \c_spath_rectcorner_tl
    {
      \dim_set:Nn \l__spath_rectx_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
    {\dim_max:nn {\tl_head:N \l__spath_tmpa_tl} {\l__spath_tmpb_dim}}
    \tl_if_eq:NNT \l__spath_tmpc_tl \c_spath_rectcorner_tl
    {
      \dim_set:Nn \l__spath_recty_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_tmpb_tl
\tl_set:Nx \l__spath_tmpb_tl
{
  {\dim_use:N \l__spath_tmpa_dim}
  {\dim_use:N \l__spath_tmpb_dim}
}
\tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
```

```
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_maxbb:Nn #1#2
{
  \__spath_maxbb:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_maxbb:Nn {NV, cn, cV}
\cs_new_protected_nopar:Npn \spath_gmaxbb:Nn #1#2
{
  \__spath_maxbb:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gmaxbb:Nn {NV, cn, cV}
```

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

`\spath_save_to_aux:Nn`
`\spath_save_to_aux:N`

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

```
\int_set:Nn \l__spath_tmpa_int {\char_value_catcode:n {`@}}
\char_set_catcode_letter:N @
\cs_new_protected_nopar:Npn \spath_save_to_aux:Nn #1#2 {
  \tl_if_empty:nF {#2}
  {
    \tl_clear:N \l__spath_tmpa_tl
    \tl_put_right:Nn \l__spath_tmpa_tl {
      \ExplSyntaxOn
      \tl_gclear_new:N #1
      \tl_gset:Nn #1 {#2}
      \ExplSyntaxOff
    }
    \protected@write\@auxout{}{
      \tl_to_str:N \l__spath_tmpa_tl
    }
  }
}
\char_set_catcode:nn {`@} {\l__spath_tmpa_int}
\cs_generate_variant:Nn \spath_save_to_aux:Nn {cn, cV, NV}
\cs_new_protected_nopar:Npn \spath_save_to_aux:N #1
{
  \tl_if_exist:NT #1
  {
    \spath_save_to_aux:NV #1#1
  }
}
\cs_generate_variant:Nn \spath_save_to_aux:N {c}
```

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

3.3 Path Manipulation

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

`\spath_translate:Nnnn`
`\spath_translate:Nnn`
`\spath_gtranslate:Nnnn`
`\spath_gtranslate:Nnn`

Translates a path by an amount.

```
\cs_new_protected_nopar:Npn \__spath_translate:nnn #1#2#3
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \tl_clear:N \l__spath_tmpb_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_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_if_eq:NNTF \l__spath_tmpc_tl \c_spath_rectsize_tl
    {
      \dim_set:Nn \l__spath_tmpa_dim {\tl_head: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}

    \tl_if_eq:NNTF \l__spath_tmpc_tl \c_spath_rectsize_tl
    {
      \dim_set:Nn \l__spath_tmpb_dim {\tl_head: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}
    }
  }
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
  \group_end:
}
\cs_generate_variant:Nn \__spath_translate:nnn {nVV}
\cs_new_protected_nopar:Npn \spath_translate:Nnnn #1#2#3#4
{
  \__spath_translate:nnn {#2}{#3}{#4}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_translate:Nnnn {NVxx, NVVV, NVnn}
```

```
\cs_new_protected_nopar:Npn \spath_translate:Nnn #1#2#3
{
  \spath_translate:NVnn #1#1{#2}{#3}
}
\cs_generate_variant:Nn \spath_translate:Nnn {NVV, cnn, cVV}
\cs_new_protected_nopar:Npn \spath_gtranslate:Nnnn #1#2#3#4
{
  \__spath_translate:nnn {#2}{#3}{#4}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gtranslate:Nnnn {NVxx, NVVV, NVnn}
\cs_new_protected_nopar:Npn \spath_gtranslate:Nnn #1#2#3
{
  \spath_gtranslate:NVnn #1#1{#2}{#3}
}
\cs_generate_variant:Nn \spath_gtranslate:Nnn {NVV, cnn, cVV}
```

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

```
\cs_new_protected_nopar:Npn \spath_translate:Nn #1#2
{
  \spath_translate:Nnn #1 #2
}
\cs_generate_variant:Nn \spath_translate:Nn {NV}
\cs_new_protected_nopar:Npn \spath_gtranslate:Nn #1#2
{
  \spath_gtranslate:Nnn #1 #2
}
\cs_generate_variant:Nn \spath_gtranslate:Nn {NV}
```

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

\spath_translate_to:Nnnn
\spath_translate_to:Nnn
\spath_gtranslate_to:Nnnn
\spath_gtranslate_to:Nnn

Translates a path so that it starts at a point.

```
\cs_new_protected_nopar:Npn \__spath_translate_to:nnn #1#2#3
{
  \group_begin:
  \spath_initialpoint:Nn \l__spath_tmpa_tl {#1}

  \dim_set:Nn \l__spath_tmpa_dim
  {
    #2
    -
    \tl_item:Nn \l__spath_tmpa_tl {1}
  }
  \dim_set:Nn \l__spath_tmpb_dim
  {
    #3
    -
    \tl_item:Nn \l__spath_tmpa_tl {2}
  }

  \__spath_translate:nVV {#1} \l__spath_tmpa_dim \l__spath_tmpb_dim
  \group_end:
}
```

```
\cs_new_protected_nopar:Npn \spath_translate_to:Nnnn #1#2#3#4
{
  \__spath_translate_to:nnn {#2}{#3}{#4}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_translate_to:Nnnn {NVxx, NVVV, NVnn}
\cs_new_protected_nopar:Npn \spath_translate_to:Nnn #1#2#3
{
  \spath_translate_to:NVnn #1#1{#2}{#3}
}
\cs_generate_variant:Nn \spath_translate_to:Nnn {NVV, cnn, cVV}
\cs_new_protected_nopar:Npn \spath_gtranslate_to:Nnnn #1#2#3#4
{
  \__spath_translate_to:nnn {#2}{#3}{#4}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gtranslate_to:Nnnn {NVxx, NVVV, NVnn}
\cs_new_protected_nopar:Npn \spath_gtranslate_to:Nnn #1#2#3
{
  \spath_gtranslate_to:NVnn #1#1{#2}{#3}
}
\cs_generate_variant:Nn \spath_gtranslate_to:Nnn {NVV, cnn, cVV}
```

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

```
\cs_new_protected_nopar:Npn \spath_translate_to:Nn #1#2
{
  \spath_translate_to:Nnn #1 #2
}
\cs_generate_variant:Nn \spath_translate_to:Nn {NV}
\cs_new_protected_nopar:Npn \spath_gtranslate_to:Nn #1#2
{
  \spath_gtranslate_to:Nnn #1 #2
}
\cs_generate_variant:Nn \spath_gtranslate_to:Nn {NV}
```

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

`\spath_scale:Nnnn`
`\spath_scale:Nnn`
`\spath_gscale:Nnnn`
`\spath_gscale:Nnn`

Scale a path.

```
\cs_new_protected_nopar:Npn \__spath_scale:nnn #1#2#3
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \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}

    \fp_set:Nn \l__spath_tmpa_fp {\tl_head:N \l__spath_tmpa_tl * #2}
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

    \fp_set:Nn \l__spath_tmpb_fp {\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
    {
      {\fp_to_dim:N \l__spath_tmpa_fp}
      {\fp_to_dim:N \l__spath_tmpb_fp}
    }
  }
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_scale:Nnnn #1#2#3#4
{
  \__spath_scale:nnn {#2}{#3}{#4}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_scale:Nnnn {NVnn, Nnxx}
\cs_new_protected_nopar:Npn \spath_scale:Nnn #1#2#3
{
  \spath_scale:NVnn #1#1{#2}{#3}
}
\cs_generate_variant:Nn \spath_scale:Nnn {cnn, cVV, NVV}
\cs_new_protected_nopar:Npn \spath_gscale:Nnnn #1#2#3#4
{
  \__spath_scale:nnn {#2}{#3}{#4}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gscale:Nnnn {NVnn, Nnxx}
\cs_new_protected_nopar:Npn \spath_gscale:Nnn #1#2#3
{
  \spath_gscale:NVnn #1#1{#2}{#3}
}
\cs_generate_variant:Nn \spath_gscale:Nnn {cnn, cVV, NVV}
```

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

```
\cs_new_protected_nopar:Npn \spath_scale:Nn #1#2
{
  \spath_scale:Nnn #1 #2
}

\cs_generate_variant:Nn \spath_scale:Nn {NV}
\cs_new_protected_nopar:Npn \spath_gscale:Nn #1#2
{
  \spath_gscale:Nnn #1 #2
}

\cs_generate_variant:Nn \spath_gscale:Nn {NV}
```

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

`\spath_transform:Nnnnnnnn`
`\spath_transform:Nnnnnnn`
`\spath_gtransform:Nnnnnnnn`
`\spath_gtransform:Nnnnnnn`

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

```
\cs_new_protected_nopar:Npn \__spath_transform:nnnnnnn #1#2#3#4#5#6#7
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \tl_clear:N \l__spath_tmpb_tl
  \bool_until_do:nn {
    \tl_if_empty_p:N \l__spath_tmpa_tl
  }
  {
    \tl_set:Nx \l__spath_tmpe_tl {\tl_head: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_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_set:Nx \l__spath_tmpd_tl {\tl_head:N \l__spath_tmpa_tl}
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

    \tl_if_eq:NNTF \l__spath_tmpe_tl \c_spath_rectsize_tl
    {
      \fp_set:Nn \l__spath_tmpa_fp
      {\l__spath_tmpc_tl * #2 + \l__spath_tmpd_tl * #4}
      \fp_set:Nn \l__spath_tmpb_fp
      {\l__spath_tmpc_tl * #3 + \l__spath_tmpd_tl * #5}
    }
    {
      \fp_set:Nn \l__spath_tmpa_fp
      {\l__spath_tmpc_tl * #2 + \l__spath_tmpd_tl * #4 + #6}
      \fp_set:Nn \l__spath_tmpb_fp
      {\l__spath_tmpc_tl * #3 + \l__spath_tmpd_tl * #5 + #7}
    }

    \tl_put_right:Nx \l__spath_tmpb_tl
    {
      {\fp_to_dim:N \l__spath_tmpa_fp}
      {\fp_to_dim:N \l__spath_tmpb_fp}
    }
  }

  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_transform:Nnnnnnnn #1#2#3#4#5#6#7#8
{
  \__spath_transform:nnnnnnn {#2}{#3}{#4}{#5}{#6}{#7}{#8}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_transform:Nnnnnnnn
{NVnnnnnn, Nnxxxxxx, cnnnnnnn}
\cs_new_protected_nopar:Npn \spath_transform:Nnnnnnn #1#2#3#4#5#6#7
{
  \spath_transform:NVnnnnnn #1#1{#2}{#3}{#4}{#5}{#6}{#7}
}
\cs_generate_variant:Nn \spath_transform:Nnnnnnn {cnnnnnn}
\cs_new_protected_nopar:Npn \spath_transform:Nnn #1#2#3
{
  \spath_transform:Nnnnnnnn #1{#2}#3
}
\cs_generate_variant:Nn \spath_transform:Nnn {cnn, cVn, NVn, NnV}
\cs_new_protected_nopar:Npn \spath_transform:Nn #1#2
{
  \spath_transform:NVnnnnnn #1#1#2
}
\cs_generate_variant:Nn \spath_transform:Nn {cn, cV, NV}

\cs_new_protected_nopar:Npn \spath_gtransform:Nnnnnnnn #1#2#3#4#5#6#7#8
{
  \__spath_transform:nnnnnnn {#2}{#3}{#4}{#5}{#6}{#7}{#8}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gtransform:Nnnnnnnn {NVnnnnnn, Nnxxxxxx, cnnnnnnn}
\cs_new_protected_nopar:Npn \spath_gtransform:Nnnnnnn #1#2#3#4#5#6#7
{
  \spath_gtransform:NVnnnnnn #1#1{#2}{#3}{#4}{#5}{#6}{#7}
}
\cs_generate_variant:Nn \spath_gtransform:Nnnnnnn {cnnnnnn}
\cs_new_protected_nopar:Npn \spath_gtransform:Nnn #1#2#3
{
  \spath_gtransform:Nnnnnnnn #1{#2}#3
}
\cs_generate_variant:Nn \spath_gtransform:Nnn {cnn, cVn, NVn, NnV}
\cs_new_protected_nopar:Npn \spath_gtransform:Nn #1#2
{
  \spath_gtransform:NVnnnnnn #1#1#2
}
\cs_generate_variant:Nn \spath_gtransform:Nn {cn, cV, NV}
```

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

\spath_span:Nnnn
\spath_span:Nnn
\spath_gspan:Nnnn
\spath_gspan:Nnn
\spath_normalise:Nn
\spath_normalise:N
\spath_gnormalise:Nn
\spath_gnormalise:N

The `span` functions transform a path to start and end at specified points. The `normalise` functions transform it to start at the origin and end at (1pt, 0pt).

If the path starts and ends at the same point then it is translated to the specified point (or origin) but not otherwise changed.

```
\cs_new_protected_nopar:Npn \__spath_span:nnn #1#2#3
{
  \group_begin:
  \spath_initialpoint:Nn \l__spath_tmpa_tl {#1}
  \spath_finalpoint:Nn \l__spath_tmpb_tl {#1}

  \fp_set:Nn \l__spath_tmpa_fp
  {
    (\tl_item:Nn \l__spath_tmpb_tl {1}) -
    (\tl_item:Nn \l__spath_tmpa_tl {1})
  }
```

```
\fp_set:Nn \l__spath_tmpb_fp
{
  (\tl_item:Nn \l__spath_tmpb_tl {2}) -
  (\tl_item:Nn \l__spath_tmpa_tl {2})
}
\fp_set:Nn \l__spath_tmpc_fp
{
  (\l__spath_tmpa_fp) * (\l__spath_tmpa_fp)
  +
  (\l__spath_tmpb_fp * \l__spath_tmpb_fp)
}

\fp_compare:nTF
{
  \l__spath_tmpc_fp < 0.001
}
{
  \spath_translate_to:Nnnn \l__spath_tmpd_tl {#1} #2
}
{
  \fp_set:Nn \l__spath_tmpa_fp
  {
    (
    ((\tl_item:nn {#3} {1})
    -
    (\tl_item:nn {#2} {1}))
    *
    ((\tl_item:Nn \l__spath_tmpb_tl {1})
    -
    (\tl_item:Nn \l__spath_tmpa_tl {1}))
    +
    ((\tl_item:nn {#3} {2})
    -
    (\tl_item:nn {#2} {2}))
    *
    ((\tl_item:Nn \l__spath_tmpb_tl {2})
    -
    (\tl_item:Nn \l__spath_tmpa_tl {2}))
    )
    /
    \l__spath_tmpc_fp
  }
  \fp_set:Nn \l__spath_tmpb_fp
  {
    (
    ((\tl_item:nn {#3} {2})
    -
    (\tl_item:nn {#2} {2}))
    *
    ((\tl_item:Nn \l__spath_tmpb_tl {1})
    -
    (\tl_item:Nn \l__spath_tmpa_tl {1}))
    -
    ((\tl_item:nn {#3} {1})
      -
      (\tl_item:nn {#2} {1}))
      *
      ((\tl_item:Nn \l__spath_tmpb_tl {2})
      -
      (\tl_item:Nn \l__spath_tmpa_tl {2}))
      )
      /
      \l__spath_tmpc_fp
    }

    \tl_set:Nx \l__spath_tmpc_tl
    {
      {
        \fp_to_decimal:N \l__spath_tmpa_fp
      }
      {
        \fp_to_decimal:N \l__spath_tmpb_fp
      }
      {
        \fp_eval:n { - \l__spath_tmpb_fp }
      }
      {
        \fp_to_decimal:N \l__spath_tmpa_fp
      }
      {
        \fp_to_dim:n
        {
          \tl_item:nn {#2} {1}
          -
          \l__spath_tmpa_fp * (\tl_item:Nn \l__spath_tmpa_tl {1})
          +
          \l__spath_tmpb_fp * (\tl_item:Nn \l__spath_tmpa_tl {2})
        }
      }
      {
        \fp_to_dim:n
        {
          \tl_item:nn {#2} {2}
          -
          \l__spath_tmpb_fp * (\tl_item:Nn \l__spath_tmpa_tl {1})
          -
          \l__spath_tmpa_fp * (\tl_item:Nn \l__spath_tmpa_tl {2})
        }
      }
    }
    \spath_transform:NnV \l__spath_tmpd_tl {#1} \l__spath_tmpc_tl
  }
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpd_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_span:Nnnn #1#2#3#4
{
  \__spath_span:nnn {#2}{#3}{#4}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_span:Nnnn {NVnn, NVVV, NnVV}
\cs_new_protected_nopar:Npn \spath_span:Nnn #1#2#3
{
  \spath_span:NVnn #1#1{#2}{#3}
}
\cs_generate_variant:Nn \spath_span:Nnn {NVV, cnn, cvv, cVV}
\cs_new_protected_nopar:Npn \spath_gspan:Nnnn #1#2#3#4
{
  \__spath_span:nnn {#2}{#3}{#4}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gspan:Nnnn {NVnn, NVVV}
\cs_new_protected_nopar:Npn \spath_gspan:Nnn #1#2#3
{
  \spath_gspan:NVnn #1#1{#2}{#3}
}
\cs_generate_variant:Nn \spath_gspan:Nnn {NVV, cnn, cvv, cVV}
\cs_new_protected_nopar:Npn \__spath_normalise:n #1
{
  \__spath_span:nnn {#1}{{0pt}{0pt}}{{1pt}{0pt}}
}
\cs_new_protected_nopar:Npn \spath_normalise:Nn #1#2
{
  \__spath_normalise:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_normalise:Nn {cn,NV, cV, cv}
\cs_new_protected_nopar:Npn \spath_normalise:N #1
{
  \spath_normalise:NV #1#1
}
\cs_generate_variant:Nn \spath_normalise:N {c}
\cs_new_protected_nopar:Npn \spath_gnormalise:Nn #1#2
{
  \__spath_normalise:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gnormalise:Nn {cn,NV, cV, cv}
\cs_new_protected_nopar:Npn \spath_gnormalise:N #1
{
  \spath_gnormalise:NV #1#1
}
\cs_generate_variant:Nn \spath_gnormalise:N {c}
```

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

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

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

```
\cs_new_protected_nopar:Npn \__spath_splice_between:nnn #1#2#3
{
  \group_begin:
  \spath_finalpoint:Nn \l__spath_tmpd_tl {#1}
  \spath_initialpoint:Nn \l__spath_tmpe_tl {#3}
  \spath_span:NnVV \l__spath_tmpb_tl {#2} \l__spath_tmpd_tl \l__spath_tmpe_tl
  \spath_append_no_move:NnV \l__spath_tmpa_tl {#1} \l__spath_tmpb_tl
  \spath_append_no_move:Nn \l__spath_tmpa_tl {#3}
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_splice_between:Nnnn #1#2#3#4
{
  \__spath_splice_between:nnn {#2}{#3}{#4}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_splice_between:Nnnn {NVnn, NVVV}
\cs_new_protected_nopar:Npn \spath_splice_between:Nnn #1#2#3
{
  \spath_splice_between:NVnn #1#1{#2}{#3}
}
\cs_generate_variant:Nn \spath_splice_between:Nnn {NVV, cnn, cvv, Nvn, NVn}
\cs_new_protected_nopar:Npn \spath_gsplice_between:Nnnn #1#2#3#4
{
  \__spath_splice_between:nnn {#2}{#3}{#4}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gsplice_between:Nnnn {NVnn, NVVV}
\cs_new_protected_nopar:Npn \spath_gsplice_between:Nnn #1#2#3
{
  \spath_gsplice_between:NVnn #1#1{#2}{#3}
}
\cs_generate_variant:Nn \spath_gsplice_between:Nnn {NVV, cnn, cvv, Nvn, NVn}
```

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

`\spath_hobby_curve:Nnnnn` Construct the curve from Hobby's algorithm given the start, end, and tangent directions.

```
\cs_new_protected_nopar:Npn \__spath_hobby_curve:nnnn #1#2#3#4
{
  \group_begin:
```

First tangent vector projected onto vector between endpoints

Something a bit weird here as the components are opposite to how I thought they should be ...

```
  \fp_set:Nn \l__spath_tmpa_fp
  {
    (
    (\tl_item:nn {#2} {1})
    *
    (\tl_item:nn {#4} {1} - \tl_item:nn {#1} {1})
    +
    (\tl_item:nn {#2} {2})
    *
    (\tl_item:nn {#4} {2} - \tl_item:nn {#1} {2})
    )
    /
    sqrt
    (
    (
    (\tl_item:nn {#2} {1})^2
    +
    (\tl_item:nn {#2} {2})^2
    )
    *
    (
    (\tl_item:nn {#4} {1} - \tl_item:nn {#1} {1})^2
    +
    (\tl_item:nn {#4} {2} - \tl_item:nn {#1} {2})^2
    )
    )
  }
  \fp_set:Nn \l__spath_tmpb_fp
  {
    (
    -
    (\tl_item:nn {#2} {1})
    *
    (\tl_item:nn {#4} {2} - \tl_item:nn {#1} {2})
    +
    (\tl_item:nn {#2} {2})
    *
    (\tl_item:nn {#4} {1} - \tl_item:nn {#1} {1})
    )
    /
    sqrt
    (
    (
    (\tl_item:nn {#2} {1})^2
    +
    (\tl_item:nn {#2} {2})^2
    )
    *
    (
    (\tl_item:nn {#4} {1} - \tl_item:nn {#1} {1})^2
    +
    (\tl_item:nn {#4} {2} - \tl_item:nn {#1} {2})^2
    )
    )
  }
```

Second tangent vector projected onto vector between endpoints

```
\fp_set:Nn \l__spath_tmpc_fp
{
  (
  (\tl_item:nn {#3} {1})
  *
  (\tl_item:nn {#4} {1} - \tl_item:nn {#1} {1})
  +
  (\tl_item:nn {#3} {2})
  *
  (\tl_item:nn {#4} {2} - \tl_item:nn {#1} {2})
  )
  /
  sqrt
  (
  (
  (\tl_item:nn {#3} {1})^2
  +
  (\tl_item:nn {#3} {2})^2
  )
  *
  (
  (\tl_item:nn {#4} {1} - \tl_item:nn {#1} {1})^2
  +
  (\tl_item:nn {#4} {2} - \tl_item:nn {#1} {2})^2
  )
  )
}
\fp_set:Nn \l__spath_tmpd_fp
{
  (
  (\tl_item:nn {#3} {1})
  *
  (\tl_item:nn {#4} {2} - \tl_item:nn {#1} {2})
  -
  (\tl_item:nn {#3} {2})
  *
  (\tl_item:nn {#4} {1} - \tl_item:nn {#1} {1})
  )
  /
  sqrt
  (
  (
  (\tl_item:nn {#3} {1})^2
  +
  (\tl_item:nn {#3} {2})^2
  )
  *
  (
  (\tl_item:nn {#4} {1} - \tl_item:nn {#1} {1})^2
  +
  (\tl_item:nn {#4} {2} - \tl_item:nn {#1} {2})^2
  )
  )
}

\fp_set:Nn \l__spath_tmpe_fp
{
  (
  2 + sqrt(2) *
  (\l__spath_tmpb_fp - 1/16 * \l__spath_tmpd_fp)
  *
  (\l__spath_tmpd_fp - 1/16 * \l__spath_tmpb_fp)
  *
  (\l__spath_tmpa_fp - \l__spath_tmpc_fp)
  )
  /
  (
  1
  +
  (1 - (3 - sqrt(5))/2)
  *
  \l__spath_tmpa_fp
  +
  (3 - sqrt(5))/2
  *
  \l__spath_tmpc_fp
  )
  *
  sqrt
  (
  (
  (\tl_item:nn {#4} {1} - \tl_item:nn {#1} {1})^2
  +
  (\tl_item:nn {#4} {2} - \tl_item:nn {#1} {2})^2
  )
  /
  (
  (\tl_item:nn {#2} {1})^2
  +
  (\tl_item:nn {#2} {2})^2
  )
  )
  /3
}
\fp_set:Nn \l__spath_tmpf_fp
{
  (
  2 - sqrt(2) *
  (\l__spath_tmpb_fp - 1/16 * \l__spath_tmpd_fp)
  *
  (\l__spath_tmpd_fp - 1/16 * \l__spath_tmpb_fp)
  *
  (\l__spath_tmpa_fp - \l__spath_tmpc_fp)
  )
  /
  (
  1
  +
  (1 - (3 - sqrt(5))/2)
  *
  \l__spath_tmpc_fp
  +
  (3 - sqrt(5))/2
  *
  \l__spath_tmpa_fp
  )
  *
  sqrt
  (
  (
  (\tl_item:nn {#4} {1} - \tl_item:nn {#1} {1})^2
  +
  (\tl_item:nn {#4} {2} - \tl_item:nn {#1} {2})^2
  )
  /
  (
  (\tl_item:nn {#3} {1})^2
  +
  (\tl_item:nn {#3} {2})^2
  )
  )
  /3
}

\tl_set:Nx \l__spath_tmpa_tl
{
  {
    \fp_to_dim:n
    {
      \tl_item:nn {#1} {1}
      +
      \l__spath_tmpe_fp
      *
      (\tl_item:nn {#2} {1})
    }
  }
  {
    \fp_to_dim:n
    {
      \tl_item:nn {#1} {2}
      +
      \l__spath_tmpe_fp
      *
      (\tl_item:nn {#2} {2})
    }
  }
}
\tl_set:Nx \l__spath_tmpb_tl
{
  {
    \fp_to_dim:n
    {
      \tl_item:nn {#4} {1}
      -
      \l__spath_tmpf_fp
      *
      (\tl_item:nn {#3} {1})
    }
  }
```

```
    {
      \fp_to_dim:n
      {
        \tl_item:nn {#4} {2}
        -
        \l__spath_tmpf_fp
        *
        (\tl_item:nn {#3} {2})
      }
    }
  }

  \tl_clear:N \l__spath_tmpc_tl
  \tl_set:NV \l__spath_tmpc_tl \c_spath_moveto_tl
  \tl_put_right:Nn \l__spath_tmpc_tl {#1}
  \tl_put_right:NV \l__spath_tmpc_tl \c_spath_curvetoa_tl
  \tl_put_right:NV \l__spath_tmpc_tl \l__spath_tmpa_tl
  \tl_put_right:NV \l__spath_tmpc_tl \c_spath_curvetob_tl
  \tl_put_right:NV \l__spath_tmpc_tl \l__spath_tmpb_tl
  \tl_put_right:NV \l__spath_tmpc_tl \c_spath_curveto_tl
  \tl_put_right:Nn \l__spath_tmpc_tl {#4}
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpc_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_hobby_curve:Nnnnn #1#2#3#4#5
{
  \__spath_hobby_curve:nnnn {#2}{#3}{#4}{#5}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_hobby_curve:Nnnnn {NVVVV}
\cs_new_protected_nopar:Npn \spath_ghobby_curve:Nnnnn #1#2#3#4#5
{
  \__spath_hobby_curve:nnnn {#2}{#3}{#4}{#5}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_ghobby_curve:Nnnnn {NVVVV}
```

(*End of definition for* `\spath_hobby_curve:Nnnnn`*.*)

`\spath_curve_between:Nnn`
`\spath_curve_between:Nn`
`\spath_gcurve_between:Nnn`
`\spath_gcurve_between:Nn`

This takes two paths and returns a single path formed by joining the two paths by a curve.

```
\cs_new_protected_nopar:Npn \__spath_curve_between:nn #1#2
{
  \group_begin:
  \spath_finalpoint:Nn \l__spath_tmpa_tl {#1}
  \spath_finaltangent:Nn \l__spath_tmpb_tl {#1}
  \spath_initialpoint:Nn \l__spath_tmpc_tl {#2}
  \spath_initialtangent:Nn \l__spath_tmpd_tl {#2}

  \spath_hobby_curve:NVVVV \l__spath_tmpe_tl
  \l__spath_tmpa_tl \l__spath_tmpb_tl \l__spath_tmpd_tl \l__spath_tmpc_tl

  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \spath_append_no_move:NV \l__spath_tmpa_tl \l__spath_tmpe_tl
  \spath_append_no_move:Nn \l__spath_tmpa_tl {#2}
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_curve_between:Nnn #1#2#3
{
  \__spath_curve_between:nn {#2}{#3}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_curve_between:Nnn {NVn, NVV}
\cs_new_protected_nopar:Npn \spath_curve_between:Nn #1#2
{
  \spath_curve_between:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_curve_between:Nn {NV, cn, cv}
\cs_new_protected_nopar:Npn \spath_gcurve_between:Nnn #1#2#3
{
  \__spath_curve_between:nn {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gcurve_between:Nnn {NVn, NVV}
\cs_new_protected_nopar:Npn \spath_gcurve_between:Nn #1#2
{
  \spath_gcurve_between:NVnn #1#1{#2}
}
\cs_generate_variant:Nn \spath_gcurve_between:Nn {NV, cn, cv}
```

(*End of definition for* `\spath_curve_between:Nnn` *and others.*)

\spath_close_with:Nn
\spath_gclose_with:Nn

Closes the first path by splicing in the second.

```
\cs_new_protected_nopar:Npn \__spath_close_with:nn #1#2
{
  \group_begin:
  \spath_finalmovepoint:Nn \l__spath_tmpa_tl {#1}
  \spath_finalpoint:Nn \l__spath_tmpb_tl {#1}
  \dim_compare:nTF
  {
    \dim_abs:n
    {
      \tl_item:Nn \l__spath_tmpa_tl {1}
      -
      \tl_item:Nn \l__spath_tmpb_tl {1}
    }
    +
    \dim_abs:n
    {
      \tl_item:Nn \l__spath_tmpa_tl {2}
      -
      \tl_item:Nn \l__spath_tmpb_tl {2}
    }
    < 0.01pt
  }
  {
    \__spath_close:n {#1}
  }
  {
    \spath_span:NnVV \l__spath_tmpc_tl {#2} \l__spath_tmpb_tl \l__spath_tmpa_tl
    \spath_append_no_move:NnV \l__spath_tmpd_tl {#1} \l__spath_tmpc_tl
    \__spath_close:V \l__spath_tmpd_tl
  }
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_close_with:Nnn #1#2#3
{
  \__spath_close_with:nn {#2}{#3}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_close_with:Nnn {cnn, cVV, cvv, NVn}
\cs_new_protected_nopar:Npn \spath_close_with:Nn #1#2
{
  \spath_close_with:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_close_with:Nn {cn, cV, cv, NV}
\cs_new_protected_nopar:Npn \spath_gclose_with:Nnn #1#2#3
{
  \__spath_close_with:nn {#2}{#3}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gclose_with:Nnn {cnn, cVV, cvv, NVn}
\cs_new_protected_nopar:Npn \spath_gclose_with:Nn #1#2
{
  \spath_gclose_with:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_gclose_with:Nn {cn, cV, cv, NV}
```

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

`\spath_close_with_curve:N`
`\spath_gclose_with_curve:N`

Closes the path with a curve.

```
\cs_new_protected_nopar:Npn \__spath_close_with_curve:n #1
{
  \group_begin:
  \spath_finalpoint:Nn \l__spath_tmpa_tl {#1}
  \spath_finaltangent:Nn \l__spath_tmpb_tl {#1}
  \spath_finalmovepoint:Nn \l__spath_tmpc_tl {#1}
  \spath_finalmovetangent:Nn \l__spath_tmpd_tl {#1}
  \dim_compare:nTF
  {
    \dim_abs:n
    {
      \tl_item:Nn \l__spath_tmpa_tl {1}
      -
      \tl_item:Nn \l__spath_tmpc_tl {1}
    }
    +
    \dim_abs:n
    {
      \tl_item:Nn \l__spath_tmpa_tl {2}
      -
      \tl_item:Nn \l__spath_tmpc_tl {2}
    }
    < 0.01pt
  }
  {
    \__spath_close:n {#1}
  }
  {

    \spath_hobby_curve:NVVVV \l__spath_tmpe_tl
    \l__spath_tmpa_tl \l__spath_tmpb_tl \l__spath_tmpd_tl \l__spath_tmpc_tl

    \tl_set:Nn \l__spath_tmpa_tl {#1}
    \spath_append_no_move:NV \l__spath_tmpa_tl \l__spath_tmpe_tl
    \__spath_close:V \l__spath_tmpa_tl
  }
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_close_with_curve:Nn #1#2
{
  \__spath_close_with_curve:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_close_with_curve:Nn {cn, cV, cv, NV}
\cs_new_protected_nopar:Npn \spath_close_with_curve:N #1
{
  \spath_close_with_curve:NV #1#1
}
\cs_generate_variant:Nn \spath_close_with_curve:N {c}
\cs_new_protected_nopar:Npn \spath_gclose_with_curve:Nn #1#2
{
  \__spath_close_with_curve:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gclose_with_curve:Nn {cn, cV, cv, NV}
\cs_new_protected_nopar:Npn \spath_gclose_with_curve:N #1
{
  \spath_gclose_with_curve:NV #1#1
}
\cs_generate_variant:Nn \spath_gclose_with_curve:N {c}
```

(*End of definition for* `\spath_close_with_curve:N` *and* `\spath_gclose_with_curve:N`*.*)

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

This welds one path to another, moving the second so that its initial point coincides with the first's final point. It is called a *weld* because the initial move of the second path is removed.

```
\cs_new_protected_nopar:Npn \__spath_weld:nn #1#2
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \tl_set:Nn \l__spath_tmpb_tl {#2}
  \spath_finalpoint:Nn \l__spath_tmpc_tl {#1}
  \spath_translate_to:NV \l__spath_tmpb_tl \l__spath_tmpc_tl

  \__spath_append_no_move:VV \l__spath_tmpa_tl \l__spath_tmpb_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_weld:Nnn #1#2#3
{
  \__spath_weld:nn {#2}{#3}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_weld:Nnn {NVV,NVn}
\cs_new_protected_nopar:Npn \spath_weld:Nn #1#2
{
  \spath_weld:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_weld:Nn {NV, Nv, cV, cv}
\cs_new_protected_nopar:Npn \spath_gweld:Nnn #1#2#3
{
  \__spath_weld:nn {#2}{#3}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gweld:Nnn {NVV, NVn}
\cs_new_protected_nopar:Npn \spath_gweld:Nn #1#2
{
  \spath_gweld:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_gweld:Nn {NV, Nv, cV, cv}
```

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

`\spath_append_no_move:Nnn`
`\spath_append_no_move:Nn`
`\spath_gappend_no_move:Nnn`
`\spath_gappend_no_move:Nn`

Append the path from the second `spath` to the first, removing the adjoining move if neither path has a rectangle either side of the join or if the first path isn't closed.

```
\cs_new_protected_nopar:Npn \__spath_append_no_move:nn #1#2
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \tl_set:Nn \l__spath_tmpb_tl {#2}
  \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpb_tl}
  \spath_finalaction:Nn \l__spath_tmpd_tl {#1}
  \bool_if:nT {
    ! \tl_if_eq_p:NN \l__spath_tmpd_tl \c_spath_closepath_tl
    &&
    ! \tl_if_eq_p:NN \l__spath_tmpd_tl \c_spath_rectcorner_tl
    &&
    \tl_if_eq_p:NN \l__spath_tmpc_tl \c_spath_moveto_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
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
  \group_end:
}
\cs_generate_variant:Nn \__spath_append_no_move:nn {VV}
\cs_new_protected_nopar:Npn \spath_append_no_move:Nnn #1#2#3
{
  \__spath_append_no_move:nn {#2}{#3}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_append_no_move:Nnn {NVV, NVn, NnV}
\cs_new_protected_nopar:Npn \spath_append_no_move:Nn #1#2
{
  \spath_append_no_move:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_append_no_move:Nn {NV, cv, Nv, cV}
\cs_new_protected_nopar:Npn \spath_gappend_no_move:Nnn #1#2#3
{
  \__spath_append_no_move:nn {#2}{#3}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gappend_no_move:Nnn {NVV, NVn}
\cs_new_protected_nopar:Npn \spath_gappend_no_move:Nn #1#2
{
  \spath_gappend_no_move:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_gappend_no_move:Nn {NV, cv, Nv, cV}
```

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

`\spath_append:Nnn`
`\spath_append:Nn`
`\spath_gappend:Nnn`
`\spath_gappend:Nn`

Prepend the path from the second `spath` to the first.

```
\cs_new_protected_nopar:Npn \spath_append:Nnn #1#2#3
{
  \tl_set:Nn #1 {#2}
  \tl_put_right:Nn #1 {#3}
}
\cs_generate_variant:Nn \spath_append:Nnn {NVV, NVn}
\cs_new_protected_nopar:Npn \spath_append:Nn #1#2
{
  \spath_append:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_append:Nn {NV, Nv, cv, cV}
\cs_new_protected_nopar:Npn \spath_gappend:Nnn #1#2#3
{
  \tl_gset:Nn #1 {#2}
  \tl_gput_right:Nn #1 {#3}
}
\cs_generate_variant:Nn \spath_gappend:Nnn {NVV, NVn}
\cs_new_protected_nopar:Npn \spath_gappend:Nn #1#2
{
  \spath_gappend:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_gappend:Nn {NV, Nv, cv, cV}
```

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

\spath_prepend_no_move:Nnn
\spath_prepend_no_move:Nn
\spath_gprepend_no_move:Nnn
\spath_gprepend_no_move:Nn

Prepend the path from the second `spath` to the first, removing the adjoining move.

```
\cs_new_protected_nopar:Npn \spath_prepend_no_move:Nnn #1#2#3
{
  \spath_append_no_move:Nnn #1{#3}{#2}
}
\cs_generate_variant:Nn \spath_prepend_no_move:Nnn {NVV, NVn}
\cs_new_protected_nopar:Npn \spath_prepend_no_move:Nn #1#2
{
  \spath_prepend_no_move:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_prepend_no_move:Nn {NV, cv}
\cs_new_protected_nopar:Npn \spath_gprepend_no_move:Nnn #1#2#3
{
  \spath_gappend_no_move:Nnn #1{#3}{#2}
}
\cs_generate_variant:Nn \spath_gprepend_no_move:Nnn {NVV, NVn}
\cs_new_protected_nopar:Npn \spath_gprepend_no_move:Nn #1#2
{
  \spath_gprepend_no_move:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_gprepend_no_move:Nn {NV, cv}
```

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

\spath_prepend:Nnn
\spath_prepend:Nn
\spath_gprepend:Nnn
\spath_gprepend:Nn

Prepend the path from the second `spath` to the first.

```
\cs_new_protected_nopar:Npn \spath_prepend:Nnn #1#2#3
{
  \spath_append:Nnn #1{#3}{#2}
}
\cs_generate_variant:Nn \spath_prepend:Nnn {NVV, NVn}
\cs_new_protected_nopar:Npn \spath_prepend:Nn #1#2
{
  \spath_prepend:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_prepend:Nn {NV}
\cs_new_protected_nopar:Npn \spath_gprepend:Nnn #1#2#3
{
  \spath_gappend:Nnn #1{#3}{#2}
}
\cs_generate_variant:Nn \spath_gprepend:Nnn {NVV, NVn}
\cs_new_protected_nopar:Npn \spath_gprepend:Nn #1#2
{
  \spath_gprepend:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_gprepend:Nn {NV}
```

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

`\spath_bake_round:Nn`
`\spath_bake_round:N`
`\spath_gbake_round:Nn`
`\spath_gbake_round:N`

The corner rounding routine is applied quite late in the process of building a soft path so this ensures that it is done.

```
\cs_new_protected_nopar:Npn \__spath_bake_round:n #1
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \pgf@@processround \l__spath_tmpa_tl\l__spath_tmpb_tl
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_bake_round:Nn #1#2
{
  \__spath_bake_round:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_bake_round:Nn {NV}
\cs_new_protected_nopar:Npn \spath_bake_round:N #1
{
  \spath_bake_round:NV #1#1
}
\cs_generate_variant:Nn \spath_bake_round:N {c}
\cs_new_protected_nopar:Npn \spath_gbake_round:Nn #1#2
{
  \__spath_bake_round:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gbake_round:Nn {NV}
\cs_new_protected_nopar:Npn \spath_gbake_round:N #1
{
  \spath_gbake_round:NV #1#1
}
\cs_generate_variant:Nn \spath_gbake_round:N {c}
```

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

`\spath_bake_shorten:Nn`
`\spath_bake_shorten:N`
`\spath_gbake_shorten:Nn`
`\spath_gbake_shorten:N`

The shortening routine is applied quite late in the process of building a soft path so this ensures that it is done.

```
\cs_new_protected_nopar:Npn \__spath_bake_shorten:n #1
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \pgfsyssoftpath@getcurrentpath\l__spath_tmpb_tl
  \pgfsyssoftpath@setcurrentpath\l__spath_tmpa_tl
  \pgf@prepare@end@of@path
  \pgf@prepare@start@of@path
  \pgfsyssoftpath@getcurrentpath\l__spath_tmpa_tl
  \pgfsyssoftpath@setcurrentpath\l__spath_tmpb_tl
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
  \group_end:
}
```

```
\cs_new_protected_nopar:Npn \spath_bake_shorten:Nn #1#2
{
  \__spath_bake_shorten:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_bake_shorten:Nn {NV}
\cs_new_protected_nopar:Npn \spath_bake_shorten:N #1
{
  \spath_bake_shorten:NV #1#1
}
\cs_generate_variant:Nn \spath_bake_shorten:N {c}
\cs_new_protected_nopar:Npn \spath_gbake_shorten:Nn #1#2
{
  \__spath_bake_shorten:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gbake_shorten:Nn {NV}
\cs_new_protected_nopar:Npn \spath_gbake_shorten:N #1
{
  \spath_gbake_shorten:NV #1#1
}
\cs_generate_variant:Nn \spath_gbake_shorten:N {c}
```

Shortening the path when it is baked can cause issues with arrows. Putting an arrow in a path definition affects the path because the path gets shortened so that the arrow ends where the path was meant to end. So an arrow affects the path definition, but the arrow is not itself part of the path so if an arrow is used when the path is defined and again when the path is used, the path will be shortened twice which might not be what is intended. Therefore it is useful to have a way to disable the shortening and place an arrow tip at the actual end of the line. The following code achieves that.

Save the original command that computes the arrow shortening.

```
\cs_set_eq:Nc \__spath_pgf_arrow_compute_shortening:n {pgf@arrow@compute@shortening}
```

After `\pgf@arrow@compute@shortening` then `\pgf@xa` is the amount to shorten the line by, so we will be setting that to 0pt. Then `\pgf@xb` is the length of the arrow head which is used to position the arrow and so before zeroing `\pgf@xa` we subtract it from `\pgf@xb` so that the arrow is placed so that its back point is at the current position.

```
\cs_new_nopar:Npn \__spath_arrow_compute_shortening:n #1
{
  \__spath_pgf_arrow_compute_shortening:n {#1}
  \bool_if:NF \l_spath_arrow_shortening_bool
  {
    \dim_sub:cn {pgf@xb} {\dim_use:c {pgf@xa}}
    \dim_zero:c {pgf@xa}
  }
}

\cs_set_eq:cN {pgf@arrow@compute@shortening} \__spath_arrow_compute_shortening:n
```

(*End of definition for* `\spath_bake_shorten:Nn` *and others.*)

`\spath_close:Nn`
`\spath_close:N`
`\spath_gclose:Nn`
`\spath_gclose:N`

Appends a close path to the end of the path.

```
\cs_new_protected_nopar:Npn \__spath_close:n #1
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \spath_finalmovepoint:NV \l__spath_tmpb_tl \l__spath_tmpa_tl
  \tl_put_right:NV \l__spath_tmpa_tl \c_spath_closepath_tl
  \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
  \group_end:
}
\cs_generate_variant:Nn \__spath_close:n {V}
\cs_new_protected_nopar:Npn \spath_close:Nn #1#2
{
  \__spath_close:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_close:Nn {NV}
\cs_new_protected_nopar:Npn \spath_close:N #1
{
  \spath_close:NV #1#1
}
\cs_generate_variant:Nn \spath_close:N {c}
\cs_new_protected_nopar:Npn \spath_gclose:Nn #1#2
{
  \__spath_close:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gclose:Nn {NV}
\cs_new_protected_nopar:Npn \spath_gclose:N #1
{
  \spath_gclose:NV #1#1
}
\cs_generate_variant:Nn \spath_gclose:N {c}
```

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

`\spath_adjust_close:Nn`
`\spath_adjust_close:N`
`\spath_adjust_gclose:Nn`
`\spath_adjust_gclose:N`

This closes a path and adjusts the end point to be where the final move point (so where the close points to) is. The intention is that this should be used if the two points are visually the same point but mathematically different.

```
\cs_new_protected_nopar:Npn \__spath_adjust_close:n #1
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \spath_finalmovepoint:NV \l__spath_tmpb_tl \l__spath_tmpa_tl
  \spath_finalpoint:NV \l__spath_tmpc_tl \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_set:Nx \l__spath_tmpd_tl {\tl_head:N \l__spath_tmpa_tl}
  \tl_if_eq:NNT \l__spath_tmpd_tl \c_spath_curveto_tl
  {
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
    \tl_clear:N \l__spath_tmpe_tl
    \tl_set:Nx \l__spath_tmpe_tl {
      {
        \dim_eval:n
        {
          \tl_item:Nn \l__spath_tmpa_tl {1}
          -
          \tl_item:Nn \l__spath_tmpc_tl {2}
          +
          \tl_item:Nn \l__spath_tmpb_tl {2}
        }
      }
      {
        \dim_eval:n
        {
          \tl_item:Nn \l__spath_tmpa_tl {2}
          -
          \tl_item:Nn \l__spath_tmpc_tl {1}
          +
          \tl_item:Nn \l__spath_tmpb_tl {1}
        }
      }
    }
    \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_put_left:NV \l__spath_tmpa_tl \l__spath_tmpe_tl
    \tl_put_left:NV \l__spath_tmpa_tl \l__spath_tmpd_tl
  }
  \tl_reverse:N \l__spath_tmpa_tl
  \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
  \tl_put_right:NV \l__spath_tmpa_tl \c_spath_closepath_tl
  \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
  \group_end:
}
\cs_generate_variant:Nn \__spath_adjust_close:n {V}
\cs_new_protected_nopar:Npn \spath_adjust_close:Nn #1#2
{
  \__spath_adjust_close:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_adjust_close:Nn {NV}
\cs_new_protected_nopar:Npn \spath_adjust_close:N #1
{
  \spath_adjust_close:NV #1#1
}
\cs_generate_variant:Nn \spath_adjust_close:N {c}
\cs_new_protected_nopar:Npn \spath_adjust_gclose:Nn #1#2
{
  \__spath_adjust_close:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
```

```
\cs_generate_variant:Nn \spath_adjust_gclose:Nn {NV}
\cs_new_protected_nopar:Npn \spath_adjust_gclose:N #1
{
  \spath_adjust_gclose:NV #1#1
}
\cs_generate_variant:Nn \spath_adjust_gclose:N {c}
```

(*End of definition for* `\spath_adjust_close:Nn` *and others.*)

\spath_open:Nn
\spath_open:N
\spath_gopen:Nn
\spath_gopen:N

Removes all close paths from the path, replacing them by `lineto` if they move any distance. Rectangles are replaced by lines with the start/end at the lower left corner.

```
\cs_new_protected_nopar:Npn \__spath_open:n #1
{
  \group_begin:
  \spath_replace_rectangles:Nn \l__spath_tmpa_tl {#1}
  \tl_clear:N \l__spath_tmpb_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:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}

    \token_case_meaning:NnF \l__spath_tmpc_tl
    {
      \c_spath_closepath_tl {

        \bool_if:nF
        {
          \dim_compare_p:n
          {
            \l__spath_move_x_dim == \l__spath_tmpa_dim
          }
          &&
          \dim_compare_p:n
          {
            \l__spath_move_y_dim == \l__spath_tmpb_dim
          }
        }
        {
          \tl_put_right:NV \l__spath_tmpb_tl \c_spath_lineto_tl

          \tl_put_right:Nx \l__spath_tmpb_tl {
            { \dim_use:N \l__spath_move_x_dim }
            { \dim_use:N \l__spath_move_y_dim }
          }
        }

        \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}
      }

      \c_spath_moveto_tl {
        \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpc_tl

        \dim_set:Nn \l__spath_move_x_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_move_y_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 {
          { \dim_use:N \l__spath_move_x_dim }
          { \dim_use:N \l__spath_move_y_dim }
        }

        \dim_set_eq:NN \l__spath_tmpa_dim \l__spath_move_x_dim
        \dim_set_eq:NN \l__spath_tmpb_dim \l__spath_move_y_dim
      }
    }
    {
      \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpc_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_right:Nx \l__spath_tmpb_tl {
        { \dim_use:N \l__spath_tmpa_dim }
        { \dim_use:N \l__spath_tmpb_dim }
      }
    }
  }
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
  \group_end:
}
\cs_generate_variant:Nn \__spath_open:n {V}
\cs_new_protected_nopar:Npn \spath_open:Nn #1#2
{
  \__spath_open:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_open:Nn {NV}
\cs_new_protected_nopar:Npn \spath_open:N #1
{
  \spath_open:NV #1#1
}
\cs_new_protected_nopar:Npn \spath_gopen:Nn #1#2
{
  \__spath_open:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gopen:Nn {NV}
\cs_new_protected_nopar:Npn \spath_gopen:N #1
```

```
{
  \spath_gopen:NV #1#1
}
\cs_generate_variant:Nn \spath_open:N {c}
\cs_generate_variant:Nn \spath_gopen:N {c}
```

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

`\spath_replace_lines:Nn`
`\spath_replace_lines:Nn`
`\spath_replace_lines:Nn`
`\spath_replace_lines:Nn`

Replace any line segments by Bézier curves.

```
\cs_new_protected_nopar:Npn \__spath_replace_lines:n #1
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \tl_clear:N \l__spath_tmpb_tl
  \dim_set:Nn \l__spath_tmpa_dim {0pt}
  \dim_set:Nn \l__spath_tmpb_dim {0pt}

  \bool_do_until:nn
  {
    \tl_if_empty_p:N \l__spath_tmpa_tl
  }
  {
    \tl_set:Nx \l__spath_tmpc_tl {\tl_item:Nn \l__spath_tmpa_tl {1}}
    \tl_set:Nx \l__spath_tmpd_tl {\tl_item:Nn \l__spath_tmpa_tl {2}}
    \tl_set:Nx \l__spath_tmpe_tl {\tl_item:Nn \l__spath_tmpa_tl {3}}

    \tl_if_eq:NNTF \l__spath_tmpc_tl \c_spath_lineto_tl
    {
      \tl_put_right:NV \l__spath_tmpb_tl \c_spath_curvetoa_tl
      \tl_put_right:Nx \l__spath_tmpb_tl
      {
        {
          \fp_to_dim:n
          {
            2/3 * (\l__spath_tmpa_dim)
            +
            1/3 * (\l__spath_tmpd_tl)
          }
        }
      }
      \tl_put_right:Nx \l__spath_tmpb_tl
      {
        {
          \fp_to_dim:n
          {
            2/3 * (\l__spath_tmpb_dim)
            +
            1/3 * (\l__spath_tmpe_tl)
          }
        }
      }
      \tl_put_right:NV \l__spath_tmpb_tl \c_spath_curvetob_tl
      \tl_put_right:Nx \l__spath_tmpb_tl
      {
        {
          \fp_to_dim:n
          {
            1/3 * (\l__spath_tmpa_dim)
            +
            2/3 * (\l__spath_tmpd_tl)
          }
        }
      }
      \tl_put_right:Nx \l__spath_tmpb_tl
      {
        {
          \fp_to_dim:n
          {
            1/3 * (\l__spath_tmpb_dim)
            +
            2/3 * (\l__spath_tmpe_tl)
          }
        }
      }
      \tl_put_right:NV \l__spath_tmpb_tl \c_spath_curveto_tl
      \__spath_tl_put_right_braced:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
      \__spath_tl_put_right_braced:NV \l__spath_tmpb_tl \l__spath_tmpe_tl
    }
    {
      \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpc_tl
      \__spath_tl_put_right_braced:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
      \__spath_tl_put_right_braced:NV \l__spath_tmpb_tl \l__spath_tmpe_tl
    }

    \dim_set:Nn \l__spath_tmpa_dim {\l__spath_tmpd_tl}
    \dim_set:Nn \l__spath_tmpb_dim {\l__spath_tmpe_tl}

    \prg_replicate:nn {3}
    {
      \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
    }
  }
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
  \group_end:
}
\cs_generate_variant:Nn \__spath_replace_lines:n {V}
\cs_new_protected_nopar:Npn \spath_replace_lines:Nn #1#2
{
  \__spath_replace_lines:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_replace_lines:Nn {NV, cV, cv, Nv}
\cs_new_protected_nopar:Npn \spath_replace_lines:N #1
{
  \spath_replace_lines:NV #1#1
}
\cs_generate_variant:Nn \spath_replace_lines:N {c}
```

```
\cs_new_protected_nopar:Npn \spath_greplace_lines:Nn #1#2
{
  \__spath_replace_lines:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_greplace_lines:Nn {NV, cV, cv, Nv}
\cs_new_protected_nopar:Npn \spath_greplace_lines:N #1
{
  \spath_greplace_lines:NV #1#1
}
\cs_generate_variant:Nn \spath_greplace_lines:N {c}
```

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

\spath_replace_rectangles:Nn
\spath_replace_rectangles:Nn
\spath_replace_rectangles:Nn
\spath_replace_rectangles:Nn

Replace any rectangle components by lines.

```
\cs_new_protected_nopar:Npn \__spath_replace_rectangles:n #1
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \tl_clear:N \l__spath_tmpb_tl

  \bool_do_until: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_set:Nx \l__spath_tmpd_tl {\tl_head:N \l__spath_tmpa_tl }
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl }
    \tl_set:Nx \l__spath_tmpe_tl {\tl_head:N \l__spath_tmpa_tl }
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl }

    \tl_if_eq:NNTF \l__spath_tmpc_tl \c_spath_rectcorner_tl
    {

      \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl }
      \dim_set:Nn \l__spath_tmpa_dim
      {
        \tl_item:Nn \l__spath_tmpa_tl {1}
      }
      \dim_set:Nn \l__spath_tmpb_dim
      {
        \tl_item:Nn \l__spath_tmpa_tl {2}
      }

      \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_put_right:NV \l__spath_tmpb_tl \c_spath_moveto_tl
      \__spath_tl_put_right_braced:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
      \__spath_tl_put_right_braced:NV \l__spath_tmpb_tl \l__spath_tmpe_tl

      \tl_put_right:NV \l__spath_tmpb_tl \c_spath_lineto_tl
      \tl_put_right:Nx \l__spath_tmpb_tl
      {
        {
          \fp_to_dim:n { \l__spath_tmpd_tl + \l__spath_tmpa_dim }
        }
      }
      \__spath_tl_put_right_braced:NV \l__spath_tmpb_tl \l__spath_tmpe_tl

      \tl_put_right:NV \l__spath_tmpb_tl \c_spath_lineto_tl
      \tl_put_right:Nx \l__spath_tmpb_tl
      {
        {
          \fp_to_dim:n { \l__spath_tmpd_tl + \l__spath_tmpa_dim }
        }
      }
      \tl_put_right:Nx \l__spath_tmpb_tl
      {
        {
          \fp_to_dim:n { \l__spath_tmpe_tl + \l__spath_tmpb_dim }
        }
      }

      \tl_put_right:NV \l__spath_tmpb_tl \c_spath_lineto_tl
      \__spath_tl_put_right_braced:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
      \tl_put_right:Nx \l__spath_tmpb_tl
      {
        {
          \fp_to_dim:n { \l__spath_tmpe_tl + \l__spath_tmpb_dim }
        }
      }

      \tl_put_right:NV \l__spath_tmpb_tl \c_spath_closepath_tl
      \__spath_tl_put_right_braced:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
      \__spath_tl_put_right_braced:NV \l__spath_tmpb_tl \l__spath_tmpe_tl

    }
    {
      \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpc_tl
      \__spath_tl_put_right_braced:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
      \__spath_tl_put_right_braced:NV \l__spath_tmpb_tl \l__spath_tmpe_tl
    }
  }
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
  \group_end:
}
\cs_generate_variant:Nn \__spath_replace_rectangles:n {V}
\cs_new_protected_nopar:Npn \spath_replace_rectangles:Nn #1#2
{
  \__spath_replace_rectangles:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_replace_rectangles:Nn {NV, cV, cv, Nv}
```

```
\cs_new_protected_nopar:Npn \spath_replace_rectangles:N #1
{
  \spath_replace_rectangles:NV #1#1
}
\cs_generate_variant:Nn \spath_replace_rectangles:N {c}
\cs_new_protected_nopar:Npn \spath_greplace_rectangles:Nn #1#2
{
  \__spath_replace_rectangles:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_greplace_rectangles:Nn {NV, cV, cv, Nv}
\cs_new_protected_nopar:Npn \spath_greplace_rectangles:N #1
{
  \spath_greplace_rectangles:NV #1#1
}
\cs_generate_variant:Nn \spath_greplace_rectangles:N {c}
```

(*End of definition for* `\spath_replace_rectangles:Nn`.)

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

Remove any component that is simply a moveto.

```
\cs_new_protected_nopar:Npn \__spath_remove_empty_components:n #1
{
  \group_begin:
  \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
  \tl_clear:N \l__spath_tmpa_tl
  \seq_map_inline:Nn \l__spath_tmpa_seq
  {
    \int_compare:nF
    {
      \tl_count:n {##1} == 3
    }
    {
      \tl_put_right:Nn \l__spath_tmpa_tl {##1}
    }
  }
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_remove_empty_components:Nn #1#2
{
  \__spath_remove_empty_components:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_remove_empty_components:Nn {NV}
\cs_new_protected_nopar:Npn \spath_remove_empty_components:N #1
{
  \spath_remove_empty_components:NV #1#1
}
\cs_generate_variant:Nn \spath_remove_empty_components:N {c}
\cs_new_protected_nopar:Npn \spath_gremove_empty_components:Nn #1#2
{
  \__spath_remove_empty_components:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gremove_empty_components:Nn {NV}
\cs_new_protected_nopar:Npn \spath_gremove_empty_components:N #1
{
  \spath_gremove_empty_components:NV #1#1
}
\cs_generate_variant:Nn \spath_gremove_empty_components:N {c}
```

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

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

```
\prg_new_protected_conditional:Npnn \spath_if_eq:nn #1#2 { T, F, TF }
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \tl_set:Nn \l__spath_tmpb_tl {#2}
  \bool_gset_true:N \g__spath_tmpa_bool
  \int_compare:nNnTF
  {\tl_count:N \l__spath_tmpa_tl}
  =
  {\tl_count:N \l__spath_tmpb_tl}
  {
    \int_step_inline:nnnn {1} {3} {\tl_count:N \l__spath_tmpa_tl}
    {
      \tl_set:Nx \l__spath_tmpc_tl {\tl_item:Nn \l__spath_tmpa_tl {##1}}
      \tl_set:Nx \l__spath_tmpd_tl {\tl_item:Nn \l__spath_tmpb_tl {##1}}
      \tl_if_eq:NNF \l__spath_tmpc_tl \l__spath_tmpd_tl
      {
        \bool_gset_false:N \g__spath_tmpa_bool
      }
      \dim_set:Nn \l__spath_tmpa_dim {\tl_item:Nn \l__spath_tmpa_tl {##1+1}}
      \dim_set:Nn \l__spath_tmpb_dim {\tl_item:Nn \l__spath_tmpb_tl {##1+1}}
      \dim_compare:nF
      {
        \dim_abs:n
        {
          \l__spath_tmpa_dim - \l__spath_tmpb_dim
        }
        < 0.001pt
      }
      {
        \bool_gset_false:N \g__spath_tmpa_bool
      }
      \dim_set:Nn \l__spath_tmpa_dim {\tl_item:Nn \l__spath_tmpa_tl {##1+2}}
      \dim_set:Nn \l__spath_tmpb_dim {\tl_item:Nn \l__spath_tmpb_tl {##1+2}}
      \dim_compare:nF
      {
        \dim_abs:n
        {
          \l__spath_tmpa_dim - \l__spath_tmpb_dim
        }
        < 0.001pt
      }
      {
        \bool_gset_false:N \g__spath_tmpa_bool
      }
    }
  }
  {
    \bool_gset_false:N \g__spath_tmpa_bool
  }
  \group_end:
  \bool_if:NTF \g__spath_tmpa_bool
  {
    \prg_return_true:
  }
  {
    \prg_return_false:
  }
}
\prg_generate_conditional_variant:Nnn \spath_if_eq:nn {VV, Vn, nV, vv} {TF, T, F}
```

(*End of definition for* `\spath_if_eq:nn`.)

3.4 Splitting Commands

`\spath_split_curve:NNnn`
`\spath_gsplit_curve:NNnn`

Splits a Bezier cubic into pieces, storing the pieces in the first two arguments.

```
\cs_new_protected_nopar:Npn \__spath_split_curve:nn #1#2
{
  \group_begin:
  \tl_set_eq:NN \l__spath_tmpa_tl \c_spath_moveto_tl
  \tl_put_right:Nx \l__spath_tmpa_tl {
    {\tl_item:nn {#1} {2}}
    {\tl_item:nn {#1} {3}}
  }
  \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curvetoa_tl
  \tl_put_right:Nx \l__spath_tmpa_tl
  {
    {\fp_to_dim:n
    {
      (1 - #2) * \tl_item:nn {#1} {2} + (#2) * \tl_item:nn {#1} {5}
    }}
    {\fp_to_dim:n
    {
      (1 - #2) * \tl_item:nn {#1} {3} + (#2) * \tl_item:nn {#1} {6}
    }}
  }

  \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curvetob_tl
  \tl_put_right:Nx \l__spath_tmpa_tl
  {
    {\fp_to_dim:n
    {
      (1 - #2)^2 * \tl_item:nn {#1} {2}
      + 2 * (1 - #2) * (#2) * \tl_item:nn {#1} {5}
      + (#2)^2 * \tl_item:nn {#1} {8}
    }}
    {\fp_to_dim:n
    {
      (1 - #2)^2 * \tl_item:nn {#1} {3}
      + 2 * (1 - #2) * (#2) * \tl_item:nn {#1} {6}
      + (#2)^2 * \tl_item:nn {#1} {9}
    }}
  }

  \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curveto_tl
  \tl_put_right:Nx \l__spath_tmpa_tl
  {
    {\fp_to_dim:n
      {
        (1 - #2)^3 * \tl_item:nn {#1} {2}
        + 3 * (1 - #2)^2 * (#2) * \tl_item:nn {#1} {5}
        + 3 * (1 - #2) * (#2)^2 * \tl_item:nn {#1} {8}
        + (#2)^3 * \tl_item:nn {#1} {11}
    }}
    {\fp_to_dim:n
    {
      (1 - #2)^3 * \tl_item:nn {#1} {3}
      + 3 * (1 - #2)^2 * (#2) * \tl_item:nn {#1} {6}
      + 3 * (1 - #2) * (#2)^2 * \tl_item:nn {#1} {9}
      + (#2)^3 * \tl_item:nn {#1} {12}
    }}
  }

  \tl_gclear:N \g__spath_output_tl
  \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpa_tl

  \tl_clear:N \l__spath_tmpa_tl
  \tl_set_eq:NN \l__spath_tmpa_tl \c_spath_moveto_tl
  \tl_put_right:Nx \l__spath_tmpa_tl
  {
    {\fp_to_dim:n
      {
        (1 - #2)^3 * \tl_item:nn {#1} {2}
        + 3 * (1 - #2)^2 * (#2) * \tl_item:nn {#1} {5}
        + 3 * (1 - #2) * (#2)^2 * \tl_item:nn {#1} {8}
        + (#2)^3 * \tl_item:nn {#1} {11}
    }}
    {\fp_to_dim:n
    {
      (1 - #2)^3 * \tl_item:nn {#1} {3}
      + 3 * (1 - #2)^2 * (#2) * \tl_item:nn {#1} {6}
      + 3 * (1 - #2) * (#2)^2 * \tl_item:nn {#1} {9}
      + (#2)^3 * \tl_item:nn {#1} {12}
    }}
  }

  \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curvetoa_tl
  \tl_put_right:Nx \l__spath_tmpa_tl
  {
    {\fp_to_dim:n
    {
      (1 - #2)^2 * \tl_item:nn {#1} {5}
      + 2 * (1 - #2) * (#2) * \tl_item:nn {#1} {8}
      + (#2)^2 * \tl_item:nn {#1} {11}
    }}
    {\fp_to_dim:n
    {
      (1 - #2)^2 * \tl_item:nn {#1} {6}
      + 2 * (1 - #2) * (#2) * \tl_item:nn {#1} {9}
      + (#2)^2 * \tl_item:nn {#1} {12}
    }}
  }
  \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curvetob_tl
  \tl_put_right:Nx \l__spath_tmpa_tl
  {
    {\fp_to_dim:n
    {
      (1 - #2) * \tl_item:nn {#1} {8} + (#2) * \tl_item:nn {#1} {11}
    }}
    {\fp_to_dim:n
    {
      (1 - #2) * \tl_item:nn {#1} {9} + (#2) * \tl_item:nn {#1} {12}
    }}
  }
  \tl_put_right:NV \l__spath_tmpa_tl \c_spath_curveto_tl
  \tl_put_right:Nx \l__spath_tmpa_tl {
    {\tl_item:nn {#1} {11}}
    {\tl_item:nn {#1} {12}}
  }

  \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpa_tl
  \group_end:
}
\cs_generate_variant:Nn \__spath_split_curve:nn {nv, nV}
\cs_new_protected_nopar:Npn \spath_split_curve:NNnn #1#2#3#4
{
  \__spath_split_curve:nn {#3}{#4}
  \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
  \tl_set:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_split_curve:NNnn {NNnV, NNVn, NNVV}
\cs_new_protected_nopar:Npn \spath_gsplit_curve:NNnn #1#2#3#4
{
  \__spath_split_curve:nn {#3}{#4}
  \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
  \tl_gset:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gsplit_curve:NNnn {NNnV, NNVn, NNVV}
```

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

`\spath_maybe_split_curve:Nn`
`\spath_maybe_gsplit_curve:Nn`

Possibly splits a bezier curve to ensure that the pieces don't self-intersect. Figuring out whether a Bezier cubic self intersects is apparently a difficult problem so we don't bother. We compute a point such that if there is an intersection then it lies on either side of the point. I don't recall where the formula came from!

```
\cs_new_protected_nopar:Npn \__spath_maybe_split_curve:n #1
{
  \group_begin:
  \fp_set:Nn \l__spath_tmpa_fp
  {
    (
    \tl_item:nn {#1} {3}
    - 3 * \tl_item:nn {#1} {6}
    + 3 * \tl_item:nn {#1} {9}
    - \tl_item:nn {#1} {12}
    )
    *
    (3 * \tl_item:nn {#1} {8} - 3 * \tl_item:nn {#1} {11})
    -
    (
    \tl_item:nn {#1} {2}
    - 3 * \tl_item:nn {#1} {5}
    + 3 * \tl_item:nn {#1} {8}
    - \tl_item:nn {#1} {11}
    )
    *
    (3 * \tl_item:nn {#1} {9} - 3 * \tl_item:nn {#1} {12})
  }
  \fp_set:Nn \l__spath_tmpb_fp
  {
    (
    \tl_item:nn {#1} {2}
    - 3 * \tl_item:nn {#1} {5}
    + 3 * \tl_item:nn {#1} {8}
    - \tl_item:nn {#1} {11}
    )
    *
    (
    3 * \tl_item:nn {#1} {6}
    - 6 * \tl_item:nn {#1} {9}
    + 3 * \tl_item:nn {#1} {12}
    )
    -
    (
    \tl_item:nn {#1} {3}
    - 3 * \tl_item:nn {#1} {6}
    + 3 * \tl_item:nn {#1} {9}
    - \tl_item:nn {#1} {12}
    )
    *
    (
    3 * \tl_item:nn {#1} {5}
    - 6 * \tl_item:nn {#1} {8}
    + 3 * \tl_item:nn {#1} {11}
    )
  }
  \fp_compare:nTF
  {
    \l__spath_tmpb_fp != 0
  }
  {
    \fp_set:Nn \l__spath_tmpa_fp {.5 * \l__spath_tmpa_fp / \l__spath_tmpb_fp}
    \bool_if:nTF
    {
      \fp_compare_p:n {0 < \l__spath_tmpa_fp}
      &&
      \fp_compare_p:n {\l__spath_tmpa_fp < 1}
    }
    {
      \__spath_split_curve:nV {#1} \l__spath_tmpa_fp
    }
    {
      \tl_gset:Nn \g__spath_output_tl { {#1} {} }
    }
  }
  {
    \tl_gset:Nn \g__spath_output_tl { {#1} {} }
  }
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_maybe_split_curve:NNn #1#2#3
{
  \__spath_maybe_split_curve:n {#3}
  \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
  \tl_set:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_maybe_split_curve:NNn {NNn, NNV }
\cs_new_protected_nopar:Npn \spath_maybe_gsplit_curve:NNn #1#2#3
{
  \__spath_maybe_split_curve:n {#3}
  \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
  \tl_gset:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_maybe_gsplit_curve:NNn {NNn, NNV}
```

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

`\spath_split_curves:Nn`
`\spath_gsplit_curves:Nn`

Slurp through the path ensuring that beziers don't self-intersect.

```
\cs_new_protected_nopar:Npn \__spath_split_curves:n #1
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \tl_clear:N \l__spath_tmpb_tl
  \tl_clear:N \l__spath_tmpc_tl
  \bool_do_until:nn
  {
    \tl_if_empty_p:N \l__spath_tmpa_tl
  }
```

```
{
  \tl_set:Nx \l__spath_tmpc_tl {\tl_head:N \l__spath_tmpa_tl}
  \token_case_meaning:NnF \l__spath_tmpc_tl
  {
    \c_spath_curvetoa_tl
    {
      \tl_clear:N \l__spath_tmpd_tl
      \tl_set_eq:NN \l__spath_tmpd_tl \c_spath_moveto_tl
      \tl_put_right:Nx \l__spath_tmpd_tl
      {
        { \dim_use:N \l__spath_tmpa_dim }
        { \dim_use:N \l__spath_tmpb_dim }
      }
      \dim_set:Nn \l__spath_tmpa_dim
      {
        \tl_item:Nn \l__spath_tmpa_tl {8}
      }
      \dim_set:Nn \l__spath_tmpb_dim
      {
        \tl_item:Nn \l__spath_tmpa_tl {9}
      }
      \prg_replicate:nn {3}
      {
        \tl_put_right:Nx \l__spath_tmpd_tl
        {
          \tl_item:Nn \l__spath_tmpa_tl {1}
          {\tl_item:Nn \l__spath_tmpa_tl {2}}
          {\tl_item:Nn \l__spath_tmpa_tl {3}}
        }
        \prg_replicate:nn {3}
        {
          \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
        }
      }

      \spath_maybe_split_curve:NNV
      \l__spath_tmpd_tl
      \l__spath_tmpe_tl
      \l__spath_tmpd_tl
      \prg_replicate:nn {3}
      {
        \tl_set:Nx \l__spath_tmpd_tl {\tl_tail:N \l__spath_tmpd_tl}
        \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl}
      }
      \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpd_tl
      \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpe_tl
    }
  }
  {
    \dim_set:Nn \l__spath_tmpa_dim
    {
      \tl_item:Nn \l__spath_tmpa_tl {2}
    }
    \dim_set:Nn \l__spath_tmpb_dim
```

```
      {
        \tl_item:Nn \l__spath_tmpa_tl {3}
      }
      \tl_put_right:Nx \l__spath_tmpb_tl
      {
        \tl_item:Nn \l__spath_tmpa_tl {1}
        {\tl_item:Nn \l__spath_tmpa_tl {2}}
        {\tl_item:Nn \l__spath_tmpa_tl {3}}
      }
      \prg_replicate:nn {3}
      {
        \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
      }
    }
  }
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_split_curves:Nn #1#2
{
  \__spath_split_curves:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_split_curves:Nn {NV, cV, cn, cv }
\cs_new_protected_nopar:Npn \spath_split_curves:N #1
{
  \spath_split_curves:NV #1#1
}
\cs_generate_variant:Nn \spath_split_curves:N {c}
\cs_new_protected_nopar:Npn \spath_gsplit_curves:Nn #1#2
{
  \__spath_split_curves:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gsplit_curves:Nn {NV, cV, cn, cv }
\cs_new_protected_nopar:Npn \spath_gsplit_curves:N #1
{
  \spath_gsplit_curves:NV #1#1
}
\cs_generate_variant:Nn \spath_gsplit_curves:N {c}
```

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

`\spath_split_line:NNnn`
`\spath_gsplit_line:NNnn`

Splits a line segment.

```
\cs_new_protected_nopar:Npn \__spath_split_line:nn #1#2
{
  \group_begin:
  \tl_set_eq:NN \l__spath_tmpa_tl \c_spath_moveto_tl
  \tl_put_right:Nx \l__spath_tmpa_tl {
    {\tl_item:nn {#1} {2}}
    {\tl_item:nn {#1} {3}}
  }
  \tl_put_right:NV \l__spath_tmpa_tl \c_spath_lineto_tl
  \tl_put_right:Nx \l__spath_tmpa_tl
  {
    {\fp_to_dim:n
    {
      (1 - #2) * \tl_item:nn {#1} {2} + (#2) * \tl_item:nn {#1} {5}
    }}
    {\fp_to_dim:n
    {
      (1 - #2) * \tl_item:nn {#1} {3} + (#2) * \tl_item:nn {#1} {6}
    }}
  }
  \tl_gclear:N \g__spath_output_tl
  \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpa_tl

  \tl_clear:N \l__spath_tmpa_tl
  \tl_set_eq:NN \l__spath_tmpa_tl \c_spath_moveto_tl
  \tl_put_right:Nx \l__spath_tmpa_tl
  {
    {\fp_to_dim:n
    {
      (1 - #2) * \tl_item:nn {#1} {2} + (#2) * \tl_item:nn {#1} {5}
    }}
    {\fp_to_dim:n
    {
      (1 - #2) * \tl_item:nn {#1} {3} + (#2) * \tl_item:nn {#1} {6}
    }}
  }
  \tl_put_right:NV \l__spath_tmpa_tl \c_spath_lineto_tl
  \tl_put_right:Nx \l__spath_tmpa_tl {
    {\tl_item:nn {#1} {5}}
    {\tl_item:nn {#1} {6}}
  }

  \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpa_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_split_line:NNnn #1#2#3#4
{
  \__spath_split_line:nn {#3}{#4}
  \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
  \tl_set:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_split_line:NNnn {NNnV, NNVn, NNVV}
\cs_new_protected_nopar:Npn \spath_gsplit_line:NNnn #1#2#3#4
{
  \__spath_split_line:nn {#3}{#4}
  \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
  \tl_gset:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gsplit_line:NNnn {NNnV, NNVn, NNVV}
```

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

\spath_split_rectangle:Nnn
\spath_gsplit_rectangle:Nnn

Cuts a rectangle at a point.

```
\cs_new_protected_nopar:Npn \__spath_split_rectangle:nn #1#2
{
  \group_begin:
  \spath_open:Nn \l__spath_tmpa_tl {#1}
  \fp_set:Nn \l__spath_tmpa_fp {4*(#2)}
  \spath_split_at:NNVV
  \l__spath_tmpa_tl \l__spath_tmpb_tl \l__spath_tmpa_tl \l__spath_tmpa_fp
  \__spath_append_no_move:VV \l__spath_tmpb_tl \l__spath_tmpa_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_split_rectangle:Nnn #1#2#3
{
  \__spath_split_rectangle:nn {#2}{#3}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_split_rectangle:Nnn {NnV, NVn, NVV}
\cs_new_protected_nopar:Npn \spath_gsplit_rectangle:Nnn #1#2#3
{
  \__spath_split_rectangle:nn {#2}{#3}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gsplit_rectangle:Nnn {NnV, NVn, NVV}
```

(*End of definition for* `\spath_split_rectangle:Nnn` *and* `\spath_gsplit_rectangle:Nnn`*.*)

\spath_split_at:NNnn
\spath_split_at:Nnn
\spath_split_at:Nn
\spath_gsplit_at:NNnn␣\spath_gsplit_at:Nnn
\spath_gsplit_at:Nn
\spath_split_at_keep_start:Nnn
\spath_split_at_keep_start:Nn
\spath_gsplit_at_keep_start:Nnn
\spath_gsplit_at_keep_start:Nn
\spath_split_at_keep_end:Nnn
\spath_split_at_keep_end:Nn
\spath_gsplit_at_keep_end:Nnn
\spath_gsplit_at_keep_end:Nn
\spath_split_at_normalised:NNnn
\spath_split_at_normalised:Nnn
\spath_split_at_normalised:Nn
alised:NNnn␣\spath_gsplit_at_normalised:Nnn
\spath_gsplit_at_normalised:Nn
\spath_split_at_normalised_keep_start:Nnn
\spath_split_at_normalised_keep_start:Nn
\spath_gsplit_at_normalised_keep_start:Nnn
\spath_gsplit_at_normalised_keep_start:Nn
\spath_split_at_normalised_keep_end:Nnn
\spath_split_at_normalised_keep_end:Nn
\spath_gsplit_at_normalised_keep_end:Nnn
\spath_gsplit_at_normalised_keep_end:Nn

Split a path according to the parameter generated by the intersection routine. The versions with two `N` arguments stores the two parts in two macros, the version with a single `N` joins them back into a single path (as separate components). The `keep` versions throw away the other part of the curve.

```
\cs_new_protected_nopar:Npn \__spath_split_at:nn #1#2
{
  \group_begin:
  \int_set:Nn \l__spath_tmpa_int {\fp_to_int:n {floor(#2) + 1}}
  \fp_set:Nn \l__spath_tmpa_fp {#2 - floor(#2)}

  % Is split point near one end or other of a component?
  \fp_compare:nT
  {
    \l__spath_tmpa_fp < 0.01
  }
  {
    % Near the start, so we'll place it at the start
    \fp_set:Nn \l__spath_tmpa_fp {0}
  }
  \fp_compare:nT
  {
    \l__spath_tmpa_fp > 0.99
  }
  {
    % Near the end, so we'll place it at the end
    \fp_set:Nn \l__spath_tmpa_fp {0}
    \int_incr:N \l__spath_tmpa_int
  }

  \int_zero:N \l__spath_tmpb_int
  \bool_set_true:N \l__spath_tmpa_bool

  \tl_set:Nn \l__spath_tmpe_tl {#1}

  \dim_zero:N \l__spath_tmpa_dim
  \dim_zero:N \l__spath_tmpb_dim

  % Remember if the component is closed
  \spath_finalaction:NV \l__spath_tmpa_tl \l__spath_tmpe_tl

  \bool_set:Nn \l__spath_closed_bool
  {
    \tl_if_eq_p:NN \l__spath_tmpa_tl \c_spath_closepath_tl
    ||
    \tl_if_eq_p:NN \l__spath_tmpa_tl \c_spath_rectcorner_tl
  }

  % Open it
  \spath_open:N \l__spath_tmpe_tl

  \tl_clear:N \l__spath_tmpa_tl
  \tl_clear:N \l__spath_tmpb_tl
  \tl_clear:N \l__spath_tmpc_tl
  \tl_clear:N \l__spath_tmpd_tl

  \bool_until_do:nn {
    \tl_if_empty_p:N \l__spath_tmpe_tl
    ||
    \int_compare_p:n { \l__spath_tmpa_int == \l__spath_tmpb_int  }
  }
  {
    \tl_set:Nx \l__spath_tmpf_tl {\tl_head:N \l__spath_tmpe_tl}
    \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
    \token_case_meaning:Nn \l__spath_tmpf_tl
    {
      \c_spath_lineto_tl
      {
        \int_incr:N \l__spath_tmpb_int
      }
      \c_spath_curvetoa_tl
      {
        \int_incr:N \l__spath_tmpb_int
      }
      \c_spath_rectcorner_tl
      {
        \int_incr:N \l__spath_tmpb_int
      }
    }
    \int_compare:nT { \l__spath_tmpb_int < \l__spath_tmpa_int  }
    {
      \tl_put_right:NV \l__spath_tmpc_tl \l__spath_tmpf_tl

      \tl_put_right:Nx \l__spath_tmpc_tl
      {{ \tl_head:N \l__spath_tmpe_tl }}
      \dim_set:Nn \l__spath_tmpa_dim {\tl_head:N \l__spath_tmpe_tl}
      \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }

      \tl_put_right:Nx \l__spath_tmpc_tl
      {{ \tl_head:N \l__spath_tmpe_tl }}
      \dim_set:Nn \l__spath_tmpb_dim {\tl_head:N \l__spath_tmpe_tl}
      \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }

    }
  }

  \tl_clear:N \l__spath_tmpd_tl
  \tl_put_right:NV \l__spath_tmpd_tl \c_spath_moveto_tl
  \tl_put_right:Nx \l__spath_tmpd_tl
  {
    {\dim_use:N \l__spath_tmpa_dim}
    {\dim_use:N \l__spath_tmpb_dim}
  }

  \fp_compare:nTF
  {
    \l__spath_tmpa_fp == 0
  }
  {
    \tl_set_eq:NN \l__spath_tmpb_tl \l__spath_tmpd_tl
    \tl_if_empty:NF \l__spath_tmpe_tl
    {
      \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpf_tl
      \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpe_tl
    }
  }
  {

    \token_case_meaning:Nn \l__spath_tmpf_tl
    {
      \c_spath_lineto_tl
      {
        \tl_put_right:NV \l__spath_tmpd_tl \l__spath_tmpf_tl
        \tl_put_right:Nx \l__spath_tmpd_tl
        {{ \tl_head:N \l__spath_tmpe_tl }}
        \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }

        \tl_put_right:Nx \l__spath_tmpd_tl
        {{ \tl_head:N \l__spath_tmpe_tl }}
        \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }

        \spath_split_line:NNVV
        \l__spath_tmpa_tl
        \l__spath_tmpb_tl
        \l__spath_tmpd_tl
        \l__spath_tmpa_fp

        \prg_replicate:nn {3} {
          \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
        }

        \tl_put_right:NV \l__spath_tmpc_tl \l__spath_tmpa_tl
        \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpe_tl
      }
      \c_spath_curvetoa_tl
      {
        \tl_put_right:NV \l__spath_tmpd_tl \l__spath_tmpf_tl
        \tl_put_right:Nx \l__spath_tmpd_tl
        {{ \tl_head:N \l__spath_tmpe_tl }}
        \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }

        \tl_put_right:Nx \l__spath_tmpd_tl
        {{ \tl_head:N \l__spath_tmpe_tl }}
        \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }

        \prg_replicate:nn {2} {

          \tl_put_right:Nx \l__spath_tmpd_tl
          { \tl_head:N \l__spath_tmpe_tl }
          \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }

          \tl_put_right:Nx \l__spath_tmpd_tl
          {{ \tl_head:N \l__spath_tmpe_tl }}
          \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }

          \tl_put_right:Nx \l__spath_tmpd_tl
          {{ \tl_head:N \l__spath_tmpe_tl }}
          \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl }
        }

        \spath_split_curve:NNVV
        \l__spath_tmpa_tl
        \l__spath_tmpb_tl
        \l__spath_tmpd_tl \l__spath_tmpa_fp

        \prg_replicate:nn {3} {
          \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
        }

        \tl_put_right:NV \l__spath_tmpc_tl \l__spath_tmpa_tl
        \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpe_tl
      }

      \c_spath_rectcorner_tl
      {
        \tl_clear:N \l__spath_tmpd_tl
        \tl_put_right:NV \l__spath_tmpd_tl \l__spath_tmpf_tl

        \tl_put_right:Nx \l__spath_tmpd_tl {{\tl_head:N \l__spath_tmpe_tl}}
```

```
        \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl}
        \tl_put_right:Nx \l__spath_tmpd_tl {{\tl_head:N \l__spath_tmpe_tl}}
        \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl}

        \tl_put_right:Nx \l__spath_tmpd_tl {\tl_head:N \l__spath_tmpe_tl}
        \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl}

        \tl_put_right:Nx \l__spath_tmpd_tl {{\tl_head:N \l__spath_tmpe_tl}}
        \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl}
        \tl_put_right:Nx \l__spath_tmpd_tl {{\tl_head:N \l__spath_tmpe_tl}}
        \tl_set:Nx \l__spath_tmpe_tl {\tl_tail:N \l__spath_tmpe_tl}

        \spath_split_rectangle:NVV
        \l__spath_tmpa_tl
        \l__spath_tmpd_tl
        \l__spath_tmpa_fp

        \tl_put_right:NV \l__spath_tmpc_tl \l__spath_tmpa_tl
        \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpe_tl
      }

    }
  }

  \bool_if:NT \l__spath_closed_bool
  {
    \prg_replicate:nn {3}
    {
      \tl_set:Nx \l__spath_tmpc_tl {\tl_tail:N \l__spath_tmpc_tl}
    }
    \tl_put_right:NV \l__spath_tmpb_tl \l__spath_tmpc_tl
    \tl_set_eq:NN \l__spath_tmpc_tl \l__spath_tmpb_tl
    \tl_clear:N \l__spath_tmpb_tl
  }

  \tl_gclear:N \g__spath_output_tl
  \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpc_tl
  \__spath_tl_gput_right_braced:NV \g__spath_output_tl \l__spath_tmpb_tl
  \group_end:
}
\cs_generate_variant:Nn  \__spath_split_at:nn {nV, VV}
\cs_new_protected_nopar:Npn \__spath_split_at_normalised:nn #1#2
{
  \group_begin:
  \spath_reallength:Nn \l__spath_tmpa_int {#1}

  \tl_set:Nx \l__spath_tmpa_tl
  {\fp_to_decimal:n {(#2) * (\l__spath_tmpa_int)}}
  \__spath_split_at:nV {#1} \l__spath_tmpa_tl
  \group_end:
}
\cs_generate_variant:Nn  \__spath_split_at_normalised:nn {nV}
\cs_new_protected_nopar:Npn \spath_split_at:NNnn #1#2#3#4
{
  \__spath_split_at:nn {#3}{#4}
  \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
  \tl_set:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_split_at:NNnn {NNVn, NNVV, NNnV}
\cs_new_protected_nopar:Npn \spath_gsplit_at:NNnn #1#2#3#4
{
  \__spath_split_at:nn {#3}{#4}
  \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
  \tl_gset:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gsplit_at:NNnn {NNVn, NNVV, NNnV}
\cs_new_protected_nopar:Npn \spath_split_at_keep_start:Nnn #1#2#3
{
  \__spath_split_at:nn {#2}{#3}
  \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_split_at_keep_start:Nnn {NVn}
\cs_new_protected_nopar:Npn \spath_split_at_keep_start:Nn #1#2
{
  \spath_split_at_keep_start:NVn #1#1{#2}
}
\cs_new_protected_nopar:Npn \spath_gsplit_at_keep_start:Nnn #1#2#3
{
  \__spath_split_at:nn {#2}{#3}
  \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gsplit_at_keep_start:Nnn {NVn}
\cs_new_protected_nopar:Npn \spath_gsplit_at_keep_start:Nn #1#2
{
  \spath_gsplit_at_keep_start:NVn #1#1{#2}
}
\cs_new_protected_nopar:Npn \spath_split_at_keep_end:Nnn #1#2#3
{
  \__spath_split_at:nn {#2}{#3}
  \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {2}}
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_split_at_keep_end:Nnn {NVn}
\cs_new_protected_nopar:Npn \spath_split_at_keep_end:Nn #1#2
{
  \spath_split_at_keep_end:NVn #1#1{#2}
}
\cs_new_protected_nopar:Npn \spath_gsplit_at_keep_end:Nnn #1#2#3
{
  \__spath_split_at:nn {#2}{#3}
  \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {2}}
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gsplit_at_keep_end:Nnn {NVn}
```

```
\cs_new_protected_nopar:Npn \spath_gsplit_at_keep_end:Nn #1#2
{
  \spath_gsplit_at_keep_end:NVn #1#1{#2}
}
\cs_new_protected_nopar:Npn \spath_split_at_normalised:NNnn #1#2#3#4
{
  \__spath_split_at_normalised:nn {#3}{#4}
  \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
  \tl_set:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_split_at_normalised:NNnn {NNVn, NNVV, NNnV, ccvn}
\cs_new_protected_nopar:Npn \spath_gsplit_at_normalised:NNnn #1#2#3#4
{
  \__spath_split_at_normalised:nn {#3}{#4}
  \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
  \tl_gset:Nx #2 {\tl_item:Nn \g__spath_output_tl {2}}
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gsplit_at_normalised:NNnn {NNVn, NNVV, NNnV, ccvn}
\cs_new_protected_nopar:Npn \spath_split_at_normalised_keep_start:Nnn #1#2#3
{
  \__spath_split_at_normalised:nn {#2}{#3}
  \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_split_at_normalised_keep_start:Nnn {NVn}
\cs_new_protected_nopar:Npn \spath_split_at_normalised_keep_start:Nn #1#2
{
  \spath_split_at_normalised_keep_start:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_split_at_normalised_keep_start:Nn {cn}
\cs_new_protected_nopar:Npn \spath_gsplit_at_normalised_keep_start:Nnn #1#2#3
{
  \__spath_split_at_normalised:nn {#2}{#3}
  \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {1}}
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gsplit_at_normalised_keep_start:Nnn {NVn}
\cs_new_protected_nopar:Npn \spath_gsplit_at_normalised_keep_start:Nn #1#2
{
  \spath_gsplit_at_normalised_keep_start:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_gsplit_at_normalised_keep_start:Nn {cn}
\cs_new_protected_nopar:Npn \spath_split_at_normalised_keep_end:Nnn #1#2#3
{
  \__spath_split_at_normalised:nn {#2}{#3}
  \tl_set:Nx #1 {\tl_item:Nn \g__spath_output_tl {2}}
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_split_at_normalised_keep_end:Nnn {NVn}
\cs_new_protected_nopar:Npn \spath_split_at_normalised_keep_end:Nn #1#2
{
  \spath_split_at_normalised_keep_end:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_split_at_normalised_keep_end:Nn {cn}
\cs_new_protected_nopar:Npn \spath_gsplit_at_normalised_keep_end:Nnn #1#2#3
{
  \__spath_split_at_normalised:nn {#2}{#3}
  \tl_gset:Nx #1 {\tl_item:Nn \g__spath_output_tl {2}}
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gsplit_at_normalised_keep_end:Nnn {NVn}
\cs_new_protected_nopar:Npn \spath_gsplit_at_normalised_keep_end:Nn #1#2
{
  \spath_gsplit_at_normalised_keep_end:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_gsplit_at_normalised_keep_end:Nn {cn}
```

```
\cs_new_protected_nopar:Npn \spath_split_at:Nnn #1#2#3
{
  \__spath_split_at:nn {#2}{#3}
  \tl_set:Nx #1
  {
    \tl_item:Nn \g__spath_output_tl {1}
    \tl_item:Nn \g__spath_output_tl {2}
  }
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_split_at:Nnn {NVn, NVV}
\cs_new_protected_nopar:Npn \spath_split_at:Nn #1#2
{
  \spath_split_at:NVn #1#1{#2}
}
\cs_new_protected_nopar:Npn \spath_gsplit_at:Nnn #1#2#3
{
  \__spath_split_at:nn {#2}{#3}
  \tl_gset:Nx #1
  {
    \tl_item:Nn \g__spath_output_tl {1}
    \tl_item:Nn \g__spath_output_tl {2}
  }
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gsplit_at:Nnn {NVn, NVV}
\cs_new_protected_nopar:Npn \spath_gsplit_at:Nn #1#2
{
  \spath_gsplit_at:NVn #1#1{#2}
}
\cs_new_protected_nopar:Npn \spath_split_at_normalised:Nnn #1#2#3
{
  \__spath_split_at_normalised:nn {#2}{#3}
  \tl_set:Nx #1
  {
    \tl_item:Nn \g__spath_output_tl {1}
    \tl_item:Nn \g__spath_output_tl {2}
  }
  \tl_gclear:N \g__spath_output_tl
}
```

```
\cs_generate_variant:Nn \spath_split_at_normalised:Nnn {NVn, NVV}
\cs_new_protected_nopar:Npn \spath_split_at_normalised:Nn #1#2
{
  \spath_split_at_normalised:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_split_at_normalised:Nn {cn}
\cs_new_protected_nopar:Npn \spath_gsplit_at_normalised:Nnn #1#2#3
{
  \__spath_split_at_normalised:nn {#2}{#3}
  \tl_gset:Nx #1
  {
    \tl_item:Nn \g__spath_output_tl {1}
    \tl_item:Nn \g__spath_output_tl {2}
  }
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gsplit_at_normalised:Nnn {NVn, NVV}
\cs_new_protected_nopar:Npn \spath_gsplit_at_normalised:Nn #1#2
{
  \spath_gsplit_at_normalised:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_gsplit_at_normalised:Nn {cn}
```

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

3.5 Shortening Paths

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

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

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

```
\cs_new_protected_nopar:Npn \__spath_shorten_at_end:nn #1#2
{
  \int_compare:nTF
  {
    \tl_count:n {#1} > 3
  }
  {
    \group_begin:
    \tl_set:Nn \l__spath_tmpa_tl {#1}
    \tl_reverse:N \l__spath_tmpa_tl

    \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {3}}

    \tl_clear:N \l__spath_tmpe_tl
    \tl_if_eq:NNTF \l__spath_tmpb_tl \c_spath_curveto_tl
    {
      \int_set:Nn \l__spath_tmpa_int {3}
    }
    {
      \int_set:Nn \l__spath_tmpa_int {1}
    }

    \prg_replicate:nn { \l__spath_tmpa_int }
    {
      \tl_put_right:Nx \l__spath_tmpe_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_tmpe_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_tmpe_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_tmpe_tl
    {
      {\tl_item:Nn \l__spath_tmpa_tl {1}}
      {\tl_item:Nn \l__spath_tmpa_tl {2}}
    }
    \tl_put_right:NV \l__spath_tmpe_tl \c_spath_moveto_tl

    \tl_reverse:N \l__spath_tmpa_tl

    \fp_set:Nn \l__spath_tmpa_fp
    {
      \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {4}}
      -
      \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {1}}
    }

    \fp_set:Nn \l__spath_tmpb_fp
    {
      \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {5}}
      -
      \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {2}}
    }

    \fp_set:Nn \l__spath_tmpc_fp
    {
      sqrt(
      \l__spath_tmpa_fp * \l__spath_tmpa_fp
      +
      \l__spath_tmpb_fp *  \l__spath_tmpb_fp
      ) * \l__spath_tmpa_int
    }

    \fp_compare:nTF
    {
      \l__spath_tmpc_fp > #2
    }
    {

      \fp_set:Nn \l__spath_tmpc_fp
      {
        (\l__spath_tmpc_fp - #2)/ \l__spath_tmpc_fp
      }

      \tl_reverse:N \l__spath_tmpe_tl

      \tl_if_eq:NNTF \l__spath_tmpb_tl \c_spath_curveto_tl
      {
        \spath_split_curve:NNVV
        \l__spath_tmpc_tl
        \l__spath_tmpd_tl
        \l__spath_tmpe_tl
        \l__spath_tmpc_fp
      }
      {
        \spath_split_line:NNVV
        \l__spath_tmpc_tl
        \l__spath_tmpd_tl
        \l__spath_tmpe_tl
        \l__spath_tmpc_fp
      }

      \prg_replicate:nn {3}
      {
        \tl_set:Nx \l__spath_tmpc_tl {\tl_tail:N \l__spath_tmpc_tl}
      }

      \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpc_tl

    }
    {

      \int_compare:nT
      {
        \tl_count:N \l__spath_tmpa_tl > 3
      }
      {
        \dim_set:Nn \l__spath_tmpa_dim {\fp_to_dim:n {#2 - \l__spath_tmpc_fp } }
        \spath_shorten_at_end:NV \l__spath_tmpa_tl \l__spath_tmpa_dim
      }
    }

    \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
    \group_end:
  }
  {
    \tl_gset:Nn \g__spath_output_tl {#1}
  }
```

```
}
\cs_new_protected_nopar:Npn \spath_shorten_at_end:Nnn #1#2#3
{
  \__spath_shorten_at_end:nn {#2}{#3}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_shorten_at_end:Nnn {NVV, cnn, cVV, NVn}
\cs_new_protected_nopar:Npn \spath_shorten_at_end:Nn #1#2
{
  \spath_shorten_at_end:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_shorten_at_end:Nn {cn, cV, NV}
\cs_new_protected_nopar:Npn \spath_gshorten_at_end:Nnn #1#2#3
{
  \__spath_shorten_at_end:nn {#2}{#3}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gshorten_at_end:Nnn {NVV, cnn, cVV, NVn}
\cs_new_protected_nopar:Npn \spath_gshorten_at_end:Nn #1#2
{
  \spath_gshorten_at_end:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_gshorten_at_end:Nn {cn, cV, NV}
```

(*End of definition for* `\spath_shorten_at_end:Nnn`*.*)

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

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

```
\cs_new_protected_nopar:Npn \__spath_shorten_at_start:nn #1#2
{
  \int_compare:nTF
  {
    \tl_count:n {#1} > 3
  }
  {
  \group_begin:
  \tl_set:Nn \l__spath_tmpa_tl {#1}

  \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {4}}

    \tl_clear:N \l__spath_tmpe_tl

  \tl_if_eq:NNTF \l__spath_tmpb_tl \c_spath_curvetoa_tl
  {
    \int_set:Nn \l__spath_tmpa_int {3}
  }
  {
    \int_set:Nn \l__spath_tmpa_int {1}
  }

  \tl_set_eq:NN \l__spath_tmpe_tl \c_spath_moveto_tl
  \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl }

```

```
\prg_replicate:nn { \l__spath_tmpa_int }
{
  \__spath_tl_put_right_braced:Nx
  \l__spath_tmpe_tl
  {\tl_item:Nn \l__spath_tmpa_tl {1}}
  \__spath_tl_put_right_braced:Nx
  \l__spath_tmpe_tl
  {\tl_item:Nn \l__spath_tmpa_tl {2}}
  \tl_put_right:Nx \l__spath_tmpe_tl {\tl_item:Nn \l__spath_tmpa_tl {3}}

  \prg_replicate:nn {3}
  {
    \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl }
  }
}
\__spath_tl_put_right_braced:Nx
\l__spath_tmpe_tl
{\tl_item:Nn \l__spath_tmpa_tl {1}}
\__spath_tl_put_right_braced:Nx
\l__spath_tmpe_tl
{\tl_item:Nn \l__spath_tmpa_tl {2}}

\fp_set:Nn \l__spath_tmpa_fp
{
  \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {5}}
  -
  \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {2}}
}

\fp_set:Nn \l__spath_tmpb_fp
{
  \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {6}}
  -
  \dim_to_fp:n {\tl_item:Nn \l__spath_tmpe_tl {3}}
}

\fp_set:Nn \l__spath_tmpc_fp
{
  sqrt(
  \l__spath_tmpa_fp * \l__spath_tmpa_fp
  +
  \l__spath_tmpb_fp *  \l__spath_tmpb_fp
  )
  *
  \l__spath_tmpa_int
}

\fp_compare:nTF
{
  \l__spath_tmpc_fp > #2
}
{

  \fp_set:Nn \l__spath_tmpc_fp
```

```
    {
      #2/ \l__spath_tmpc_fp
    }

    \tl_if_eq:NNTF \l__spath_tmpb_tl \c_spath_curvetoa_tl
    {
      \spath_split_curve:NNVV
      \l__spath_tmpc_tl
      \l__spath_tmpd_tl
      \l__spath_tmpe_tl
      \l__spath_tmpc_fp
    }
    {
      \spath_split_line:NNVV
      \l__spath_tmpc_tl
      \l__spath_tmpd_tl
      \l__spath_tmpe_tl
      \l__spath_tmpc_fp
    }

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

    \tl_put_left:NV \l__spath_tmpa_tl \l__spath_tmpd_tl

  }
  {

    \tl_put_left:NV \l__spath_tmpa_tl \c_spath_moveto_tl

    \int_compare:nT
    {
      \tl_count:N \l__spath_tmpa_tl > 3
    }
    {
      \dim_set:Nn \l__spath_tmpa_dim {\fp_to_dim:n {#2 - \l__spath_tmpc_fp } }
      \spath_shorten_at_start:NV \l__spath_tmpa_tl \l__spath_tmpa_dim
    }
  }

  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
  \group_end:
  }
  {
    \tl_gset:Nn \g__spath_output_tl {#1}
  }
}
\cs_new_protected_nopar:Npn \spath_shorten_at_start:Nnn #1#2#3
{
  \__spath_shorten_at_start:nn {#2}{#3}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
```

```
}
\cs_generate_variant:Nn \spath_shorten_at_start:Nnn {NVV, cnn, cVV, NVn}
\cs_new_protected_nopar:Npn \spath_shorten_at_start:Nn #1#2
{
  \spath_shorten_at_start:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_shorten_at_start:Nn {cn, cV, NV}
\cs_new_protected_nopar:Npn \spath_gshorten_at_start:Nnn #1#2#3
{
  \__spath_shorten_at_start:nn {#2}{#3}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gshorten_at_start:Nnn {NVV, cnn, cVV, NVn}
\cs_new_protected_nopar:Npn \spath_gshorten_at_start:Nn #1#2
{
  \spath_gshorten_at_start:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_gshorten_at_start:Nn {cn, cV, NV}
```

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

`\spath_shorten_at_both_ends:Nnn`
`\spath_shorten_at_both_ends:Nn`
`\spath_gshorten_at_both_ends:Nnn`
`\spath_gshorten_at_both_ends:Nn`

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

```
\cs_new_protected_nopar:Npn \spath_shorten_at_both_ends:Nnn #1#2#3
{
  \spath_shorten_at_start:Nnn #1{#2}{#3}
  \spath_shorten_at_end:Nnn #1{#2}{#3}
}
\cs_new_protected_nopar:Npn \spath_shorten_at_both_ends:Nn #1#2
{
  \spath_shorten_at_start:Nn #1{#2}
  \spath_shorten_at_end:Nn #1{#2}
}
\cs_generate_variant:Nn \spath_shorten_at_both_ends:Nn {cn, cV, NV}
\cs_new_protected_nopar:Npn \spath_gshorten_at_both_ends:Nnn #1#2#3
{
  \spath_gshorten_at_start:Nnn #1{#2}{#3}
  \spath_gshorten_at_end:Nnn #1{#2}{#3}
}
\cs_new_protected_nopar:Npn \spath_gshorten_at_both_ends:Nn #1#2
{
  \spath_gshorten_at_start:Nn #1{#2}
  \spath_gshorten_at_end:Nn #1{#2}
}
\cs_generate_variant:Nn \spath_gshorten_at_both_ends:Nn {cn, cV, NV}
```

(*End of definition for* `\spath_shorten_at_both_ends:Nnn` *and others.*)

3.6 Points on a Path

`\spath_point_at:Nnn`
`\spath_gpoint_at:Nnn`

Get the location of a point on a path, using the same location specification as the intersection library.

```
\cs_new_protected_nopar:Npn \__spath_point_at:nn #1#2
{
```

```
\group_begin:
\int_set:Nn \l__spath_tmpa_int {\fp_to_int:n {floor(#2) + 1}}
\fp_set:Nn \l__spath_tmpa_fp {#2 - floor(#2)}

\spath_segments_to_seq:Nn \l__spath_tmpa_seq {#1}

\int_compare:nTF
{
  \l__spath_tmpa_int < 1
}
{
  \spath_initialpoint:Nn \l__spath_tmpc_tl {#1}
}
{
  \int_compare:nTF
  {
    \l__spath_tmpa_int > \seq_count:N \l__spath_tmpa_seq
  }
  {
    \spath_finalpoint:Nn \l__spath_tmpc_tl {#1}
  }
  {

    \tl_set:Nx
    \l__spath_tmpa_tl
    {\seq_item:Nn \l__spath_tmpa_seq { \l__spath_tmpa_int} }

    \int_compare:nTF
    {
      \tl_count:N \l__spath_tmpa_tl > 3
    }
    {
      \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {4}}
    }
    {
      \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {1}}
    }

    \tl_clear:N \l__spath_tmpc_tl

    \token_case_meaning:Nn \l__spath_tmpb_tl
    {
      \c_spath_moveto_tl
      {
        \tl_set:Nx \l__spath_tmpc_tl
        {
          {
            \tl_item:Nn \l__spath_tmpa_tl {2}
          }
          {
            \tl_item:Nn \l__spath_tmpa_tl {3}
          }
        }
      }

      \c_spath_lineto_tl
      {
        \tl_set:Nx \l__spath_tmpc_tl
        {
          {\fp_to_dim:n
            {
              (1 - \l__spath_tmpa_fp) * ( \tl_item:Nn \l__spath_tmpa_tl {2} )
              +
              \l__spath_tmpa_fp * ( \tl_item:Nn \l__spath_tmpa_tl {5} )
            }
          }
          {\fp_to_dim:n
            {
              (1 - \l__spath_tmpa_fp) * ( \tl_item:Nn \l__spath_tmpa_tl {3} )
              +
              \l__spath_tmpa_fp * ( \tl_item:Nn \l__spath_tmpa_tl {6} )
            }
          }
        }
      }

      \c_spath_rectsize_tl
      {
        \fp_compare:nTF
        {
          \l__spath_tmpa_fp <= .25
        }
        {
          \tl_set:Nx \l__spath_tmpc_tl
          {
            {\fp_to_dim:n
              {
                ( \tl_item:Nn \l__spath_tmpa_tl {2} )
                +
                4 * \l__spath_tmpa_fp * ( \tl_item:Nn \l__spath_tmpa_tl {5} )
              }
            }
            {\fp_to_dim:n {\tl_item:Nn \l__spath_tmpa_tl {3} } }
          }
        }
        {
          \fp_compare:nTF
          {
            \l__spath_tmpa_fp <= .5
          }
          {
            \tl_set:Nx \l__spath_tmpc_tl
            {
              {\fp_to_dim:n
                {
                  ( \tl_item:Nn \l__spath_tmpa_tl {2} )
                  +
                  ( \tl_item:Nn \l__spath_tmpa_tl {5} )
```

```
         }
        }
        {\fp_to_dim:n
         {
          ( \tl_item:Nn \l__spath_tmpa_tl {3} )
          +
          (4 * (\l__spath_tmpa_fp) - 1) * ( \tl_item:Nn \l__spath_tmpa_tl {6} )
         }
        }
       }
      }
      {
       \fp_compare:nTF
       {
        \l__spath_tmpa_fp <= .75
       }
       {
        \tl_set:Nx \l__spath_tmpc_tl
        {
         {\fp_to_dim:n
          {
           ( \tl_item:Nn \l__spath_tmpa_tl {2} )
           +
           (3 - 4 * (\l__spath_tmpa_fp)) *( \tl_item:Nn \l__spath_tmpa_tl {5} )
          }
         }
         {\fp_to_dim:n
          {
           ( \tl_item:Nn \l__spath_tmpa_tl {3} )
           +
           ( \tl_item:Nn \l__spath_tmpa_tl {6} )
          }
         }
        }

       }
       {
        \tl_set:Nx \l__spath_tmpc_tl
        {
         {\fp_to_dim:n
          {
           ( \tl_item:Nn \l__spath_tmpa_tl {2} )
          }
         }
         {\fp_to_dim:n
          {
           ( \tl_item:Nn \l__spath_tmpa_tl {3} )
           +
           (4 - 4 *(\l__spath_tmpa_fp)) * ( \tl_item:Nn \l__spath_tmpa_tl {6} )
          }
         }
        }
       }
      }
```

```
        }
      }

      \c_spath_closepath_tl
      {
        \tl_set:Nx \l__spath_tmpc_tl
        {
          {\fp_to_dim:n
            {
              (1 - \l__spath_tmpa_fp) * ( \tl_item:Nn \l__spath_tmpa_tl {2} )
              +
              \l__spath_tmpa_fp * ( \tl_item:Nn \l__spath_tmpa_tl {5} )
            }
          }
          {\fp_to_dim:n
            {
              (1 - \l__spath_tmpa_fp) * ( \tl_item:Nn \l__spath_tmpa_tl {3} )
              +
              \l__spath_tmpa_fp * ( \tl_item:Nn \l__spath_tmpa_tl {6} )
            }
          }
        }
      }

      \c_spath_curvetoa_tl
      {
        \tl_set:Nx \l__spath_tmpc_tl
        {
          {\fp_to_dim:n
            {
              (1 - \l__spath_tmpa_fp)^3 * \tl_item:Nn \l__spath_tmpa_tl {2}
              + 3 * (1 - \l__spath_tmpa_fp)^2 * (\l__spath_tmpa_fp)
              * \tl_item:Nn \l__spath_tmpa_tl {5}
              + 3 * (1 - \l__spath_tmpa_fp) * (\l__spath_tmpa_fp)^2
              * \tl_item:Nn \l__spath_tmpa_tl {8}
              + (\l__spath_tmpa_fp)^3 * \tl_item:Nn \l__spath_tmpa_tl {11}
          }}
          {\fp_to_dim:n
            {
              (1 - \l__spath_tmpa_fp)^3 * \tl_item:Nn \l__spath_tmpa_tl {3}
              + 3 * (1 - \l__spath_tmpa_fp)^2 * (\l__spath_tmpa_fp)
              * \tl_item:Nn \l__spath_tmpa_tl {6}
              + 3 * (1 - \l__spath_tmpa_fp) * (\l__spath_tmpa_fp)^2
              * \tl_item:Nn \l__spath_tmpa_tl {9}
              + (\l__spath_tmpa_fp)^3 * \tl_item:Nn \l__spath_tmpa_tl {12}
          }}
        }
      }
    }
  }
}

\tl_gclear:N \g__spath_output_tl
\tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpc_tl
```

```
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_point_at:Nnn #1#2#3
{
  \__spath_point_at:nn {#2}{#3}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_point_at:Nnn {NVn, NVV, NnV}
\cs_new_protected_nopar:Npn \spath_gpoint_at:Nnn #1#2#3
{
  \__spath_point_at:nn {#2}{#3}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gpoint_at:Nnn {NVn, NVV, NnV}
```

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

`\spath_tangent_at:Nnn`
`\spath_gtangent_at:Nnn`

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

```
\cs_new_protected_nopar:Npn \__spath_tangent_at:nn #1#2
{
  \group_begin:
  \int_set:Nn \l__spath_tmpa_int {\fp_to_int:n {floor(#2) + 1}}
  \fp_set:Nn \l__spath_tmpa_fp {#2 - floor(#2)}

  \spath_segments_to_seq:Nn \l__spath_tmpa_seq {#1}

  \int_compare:nTF
  {
    \l__spath_tmpa_int < 1
  }
  {
    \spath_initialpoint:Nn \l__spath_tmpc_tl {#1}
  }
  {
    \int_compare:nTF
    {
      \l__spath_tmpa_int > \seq_count:N \l__spath_tmpa_seq
    }
    {
      \spath_finalpoint:Nn \l__spath_tmpc_tl {#1}
    }
    {

      \tl_set:Nx
      \l__spath_tmpa_tl
      {\seq_item:Nn \l__spath_tmpa_seq { \l__spath_tmpa_int} }

      \int_compare:nTF
      {
        \tl_count:N \l__spath_tmpa_tl > 3
      }
      {
        \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {4}}
      }
      {
        \tl_set:Nx \l__spath_tmpb_tl {\tl_item:Nn \l__spath_tmpa_tl {1}}
      }

      \tl_clear:N \l__spath_tmpc_tl

      \token_case_meaning:Nn \l__spath_tmpb_tl
      {
        \c_spath_moveto_tl
        {
          \tl_set:Nx \l__spath_tmpc_tl
          {
            {
              \tl_item:Nn \l__spath_tmpa_tl {2}
            }
            {
              \tl_item:Nn \l__spath_tmpa_tl {3}
            }
          }
        }

        \c_spath_lineto_tl
        {
          \tl_set:Nx \l__spath_tmpc_tl
          {
            {\fp_to_dim:n
              {
                ( \tl_item:Nn \l__spath_tmpa_tl {5} )
                -
                ( \tl_item:Nn \l__spath_tmpa_tl {2} )
              }
            }
            {\fp_to_dim:n
              {
                ( \tl_item:Nn \l__spath_tmpa_tl {6} )
                -
                ( \tl_item:Nn \l__spath_tmpa_tl {3} )
              }
            }
          }
        }

        \c_spath_rectsize_tl
        {
          \fp_compare:nTF
          {
            \l__spath_tmpa_fp <= .25
          }
          {
            \tl_set:Nx \l__spath_tmpc_tl
            {
```

```
    {\fp_to_dim:n
      {
        \tl_item:Nn \l__spath_tmpa_tl {5}
      }
    }
    {0pt}
  }
}
{
  \fp_compare:nTF
  {
    \l__spath_tmpa_fp <= .5
  }
  {
    \tl_set:Nx \l__spath_tmpc_tl
    {
      {0pt}
      {\fp_to_dim:n
        {
          ( \tl_item:Nn \l__spath_tmpa_tl {6} )
        }
      }
    }
  }
  {
    \fp_compare:nTF
    {
      \l__spath_tmpa_fp <= .75
    }
    {
      \tl_set:Nx \l__spath_tmpc_tl
      {
        {\fp_to_dim:n
          {
            -( \tl_item:Nn \l__spath_tmpa_tl {5} )
          }
        }
        {0pt}
      }

    }
    {
      \tl_set:Nx \l__spath_tmpc_tl
      {
        {0pt}
        {\fp_to_dim:n
          {
            - ( \tl_item:Nn \l__spath_tmpa_tl {6} )
          }
        }
      }
    }
  }
}
```

```
      }

      \c_spath_closepath_tl
      {
        \tl_set:Nx \l__spath_tmpc_tl
        {
          {\fp_to_dim:n
            {
              ( \tl_item:Nn \l__spath_tmpa_tl {5} )
              -
              ( \tl_item:Nn \l__spath_tmpa_tl {2} )
            }
          }
          {\fp_to_dim:n
            {
              ( \tl_item:Nn \l__spath_tmpa_tl {6} )
              -
              ( \tl_item:Nn \l__spath_tmpa_tl {3} )
            }
          }
        }
      }

      \c_spath_curvetoa_tl
      {
        \tl_set:Nx \l__spath_tmpc_tl
        {
          {\fp_to_dim:n
            {
              3*(1 - \l__spath_tmpa_fp)^2 * (\tl_item:Nn \l__spath_tmpa_tl {5}
              - \tl_item:Nn \l__spath_tmpa_tl {2})
              + 6 * (1 - \l__spath_tmpa_fp) * (\l__spath_tmpa_fp) *
              (\tl_item:Nn \l__spath_tmpa_tl {8}
              - \tl_item:Nn \l__spath_tmpa_tl {5})
              + 3*(\l__spath_tmpa_fp)^2 * (\tl_item:Nn \l__spath_tmpa_tl {11}
              - \tl_item:Nn \l__spath_tmpa_tl {8})
            }
          }
          {\fp_to_dim:n
            {
              3*(1 - \l__spath_tmpa_fp)^2 * (\tl_item:Nn \l__spath_tmpa_tl {6}
              - \tl_item:Nn \l__spath_tmpa_tl {3})
              + 6 * (1 - \l__spath_tmpa_fp) * (\l__spath_tmpa_fp) *
              (\tl_item:Nn \l__spath_tmpa_tl {9}
              - \tl_item:Nn \l__spath_tmpa_tl {6})
              + 3*(\l__spath_tmpa_fp)^2 * (\tl_item:Nn \l__spath_tmpa_tl {12}
              - \tl_item:Nn \l__spath_tmpa_tl {9})
          }}
        }
      }
    }
  }
}

```

```
  \tl_gclear:N \g__spath_output_tl
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpc_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_tangent_at:Nnn #1#2#3
{
  \__spath_tangent_at:nn {#2}{#3}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_tangent_at:Nnn {NVn, NVV, NnV}
\cs_new_protected_nopar:Npn \spath_gtangent_at:Nnn #1#2#3
{
  \__spath_tangent_at:nn {#2}{#3}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gtangent_at:Nnn {NVn, NVV, NnV}
```

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

`\spath_transformation_at:Nnn`
`\spath_gtransformation_at:Nnn`

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

```
\cs_new_protected_nopar:Npn \__spath_transformation_at:nn #1#2
{
  \group_begin:
  \tl_clear:N \l__spath_tmpa_tl
  \__spath_tangent_at:nn {#1}{#2}
  \tl_set_eq:NN \l__spath_tmpb_tl \g__spath_output_tl
  \fp_set:Nn \l__spath_tmpa_fp
  {
    sqrt(
    (\tl_item:Nn \l__spath_tmpb_tl {1})^2
    +
    (\tl_item:Nn \l__spath_tmpb_tl {2})^2
    )
  }
  \fp_compare:nTF {\l__spath_tmpa_fp = 0}
  {
    \fp_set:Nn \l__spath_tmpa_fp {1}
    \fp_set:Nn \l__spath_tmpb_fp {0}
  }
  {
    \fp_set:Nn \l__spath_tmpb_fp
    { (\tl_item:Nn \l__spath_tmpb_tl {2}) / \l__spath_tmpa_fp }
    \fp_set:Nn \l__spath_tmpa_fp
    { (\tl_item:Nn \l__spath_tmpb_tl {1}) / \l__spath_tmpa_fp }
  }
  \tl_set:Nx \l__spath_tmpa_tl
  {
    { \fp_to_decimal:n { \l__spath_tmpa_fp } }
    { \fp_to_decimal:n { \l__spath_tmpb_fp } }
    { \fp_to_decimal:n {- \l__spath_tmpb_fp } }
    { \fp_to_decimal:n { \l__spath_tmpa_fp } }
  }
  \__spath_point_at:nn {#1}{#2}
  \tl_put_right:NV \l__spath_tmpa_tl \g__spath_output_tl
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_transformation_at:Nnn #1#2#3
{
  \__spath_transformation_at:nn {#2}{#3}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_transformation_at:Nnn {NVn, NVV, NnV, NvV}
\cs_new_protected_nopar:Npn \spath_gtransformation_at:Nnn #1#2#3
{
  \__spath_transformation_at:nn {#2}{#3}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gtransformation_at:Nnn {NVn, NVV, NnV}
```

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

3.7 Intersection Routines

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

`\spath_intersect:NN`
`\spath_intersect:nn`

Pass two spaths to pgf's intersection routine.

```
\cs_new_protected_nopar:Npn \spath_intersect:NN #1#2
{
  \pgfintersectionofpaths%
  {%
    \pgfsetpath #1
  }{%
    \pgfsetpath #2
  }
}
\cs_new_protected_nopar:Npn \spath_intersect:nn #1#2
{
  \tl_set:Nn \l__spath_intersecta_tl {#1}
  \tl_set:Nn \l__spath_intersectb_tl {#2}
  \spath_intersect:NN \l__spath_intersecta_tl \l__spath_intersectb_tl
}
```

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

`\spath_split_component_at_intersections:Nnn`

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

```
\cs_new_protected_nopar:Npn \__spath_split_component_at_intersections:nn #1#2
{
  \group_begin:
```

```
\tl_clear:N \l__spath_tmpe_tl
\seq_clear:N \l__spath_tmpb_seq

% Find the intersections of these segments
\tl_set:Nn \l__spath_tmpb_tl {#1}
\tl_set:Nn \l__spath_tmpc_tl {#2}

% Remember if the component is closed
\spath_finalaction:NV \l__spath_tmpa_tl \l__spath_tmpb_tl

\bool_set:Nn \l__spath_closed_bool
{
  \tl_if_eq_p:NN \l__spath_tmpa_tl \c_spath_closepath_tl
  ||
  \tl_if_eq_p:NN \l__spath_tmpa_tl \c_spath_rectcorner_tl
}

% Open it
\spath_open:N \l__spath_tmpb_tl

\spath_reallength:NV \l__spath_tmpa_int \l__spath_tmpb_tl

% Sort intersections along the component
\pgfintersectionsortbyfirstpath
\spath_intersect:NN \l__spath_tmpb_tl \l__spath_tmpc_tl

% If we get intersections
\int_compare:nT {\pgfintersectionsolutions > 0}
{
  % Find the times of the intersections on the component
  \int_step_inline:nnnn {1} {1} {\pgfintersectionsolutions}
  {
    \pgfintersectiongetsolutiontimes{##1}{\l__spath_tmph_tl}{\l__spath_tmpi_tl}
    \seq_put_left:NV \l__spath_tmpb_seq \l__spath_tmph_tl
  }

  \seq_get_left:NN \l__spath_tmpb_seq \l__spath_tmpa_tl
  \fp_compare:nT
  {
    \l__spath_tmpa_tl > \l__spath_tmpa_int - .01
  }
  {
    \bool_set_false:N \l__spath_closed_bool
  }

  \seq_get_right:NN \l__spath_tmpb_seq \l__spath_tmpa_tl
  \fp_compare:nT
  {
    \l__spath_tmpa_tl < .01
  }
  {
    \bool_set_false:N \l__spath_closed_bool
  }

```

```
    \tl_set:Nn \l__spath_tmpg_tl {-1}

    \seq_map_inline:Nn \l__spath_tmpb_seq
    {
      \tl_set:Nn \l__spath_tmph_tl {##1}

      \tl_set_eq:NN \l__spath_tmpa_tl \l__spath_tmph_tl
      \int_compare:nT
      {
        \fp_to_int:n {floor( \l__spath_tmph_tl) }
        =
        \fp_to_int:n {floor( \l__spath_tmpg_tl) }
      }
      {
        \tl_set:Nx \l__spath_tmph_tl
        {
          \fp_eval:n {
            floor( \l__spath_tmph_tl )
            +
            ( \l__spath_tmph_tl - floor( \l__spath_tmph_tl) )
            /
            ( \l__spath_tmpg_tl - floor( \l__spath_tmpg_tl) )
          }
        }
      }
      \tl_set_eq:NN \l__spath_tmpg_tl \l__spath_tmpa_tl

      \spath_split_at:NNVV
      \l__spath_tmpd_tl
      \l__spath_tmpf_tl
      \l__spath_tmpb_tl
      \l__spath_tmph_tl

      \tl_put_left:NV \l__spath_tmpe_tl \l__spath_tmpf_tl
      \tl_set_eq:NN \l__spath_tmpb_tl \l__spath_tmpd_tl

    }


    \tl_put_left:NV \l__spath_tmpe_tl \l__spath_tmpb_tl

    \spath_remove_empty_components:N \l__spath_tmpe_tl

    \tl_set_eq:NN \l__spath_tmpb_tl \l__spath_tmpe_tl
  }


  \bool_if:NT \l__spath_closed_bool
  {
    \spath_join_component:Nn \l__spath_tmpb_tl {1}
  }

  \tl_gclear:N \g__spath_output_tl
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpb_tl

  \group_end:
}
\cs_new_protected_nopar:Npn \spath_split_component_at_intersections:Nnn #1#2#3
{
  \__spath_split_component_at_intersections:nn {#2}{#3}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_split_component_at_intersections:Nnn {NVn, NVV}
\cs_new_protected_nopar:Npn \spath_split_component_at_intersections:Nn #1#2
{
  \spath_split_component_at_intersections:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_split_component_at_intersections:Nn {cn, cv}
\cs_new_protected_nopar:Npn \spath_gsplit_component_at_intersections:Nnn #1#2#3
{
  \__spath_split_component_at_intersections:nn {#2}{#3}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gsplit_component_at_intersections:Nnn {NVn, NVV}
\cs_new_protected_nopar:Npn \spath_gsplit_component_at_intersections:Nn #1#2
{
  \spath_gsplit_component_at_intersections:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_gsplit_component_at_intersections:Nn {cn, cv}
```

(*End of definition for* `\spath_split_component_at_intersections:Nnn`.)

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

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

```
\cs_new_protected_nopar:Npn \__spath_split_path_at_intersections:nn #1#2
{
  \group_begin:

  \seq_clear:N \l__spath_tmpa_seq
  \seq_clear:N \l__spath_tmpb_seq

  \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
  \seq_map_inline:Nn \l__spath_tmpa_seq
  {
    \spath_split_component_at_intersections:Nnn \l__spath_tmpa_tl {##1} {#2}
    \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
  }

  \tl_gclear:N \g__spath_output_tl
  \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpb_seq {} }
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_split_path_at_intersections:Nnn #1#2#3
{
  \__spath_split_path_at_intersections:nn {#2}{#3}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_split_path_at_intersections:Nnn
{NVn, NVV, cVn, cVV, cvn, cvv}
\cs_new_protected_nopar:Npn \spath_split_path_at_intersections:Nn #1#2
{
  \spath_split_path_at_intersections:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_split_path_at_intersections:Nn {cv, NV}
\cs_new_protected_nopar:Npn \spath_gsplit_path_at_intersections:Nnn #1#2#3
{
  \__spath_split_path_at_intersections:nn {#2}{#3}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gsplit_path_at_intersections:Nnn
{NVn, NVV, cVn, cVV, cvn, cvv}
\cs_new_protected_nopar:Npn \spath_gsplit_path_at_intersections:Nn #1#2
{
  \spath_gsplit_path_at_intersections:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_gsplit_path_at_intersections:Nn {cv, NV}
\cs_new_protected_nopar:Npn \spath_split_at_intersections:NNnn #1#2#3#4
{
  \__spath_split_path_at_intersections:nn {#3}{#4}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \__spath_split_path_at_intersections:nn {#4}{#3}
  \tl_set_eq:NN #2 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_split_at_intersections:NNnn
{NNVn, NNVV, ccVn, ccVV, ccvn, ccvv}
\cs_new_protected_nopar:Npn \spath_split_at_intersections:NN #1#2
{
  \spath_split_at_intersections:NNVV #1#2#1#2
}
\cs_generate_variant:Nn \spath_split_at_intersections:NN {cc}
\cs_new_protected_nopar:Npn \spath_gsplit_at_intersections:NNnn #1#2#3#4
{
  \__spath_split_path_at_intersections:nn {#3}{#4}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \__spath_split_path_at_intersections:nn {#4}{#3}
  \tl_gset_eq:NN #2 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gsplit_at_intersections:NNnn
{NNVn, NNVV, ccVn, ccVV, ccvn, ccvv}
\cs_new_protected_nopar:Npn \spath_gsplit_at_intersections:NN #1#2
{
  \spath_gsplit_at_intersections:NNVV #1#2#1#2
}
\cs_generate_variant:Nn \spath_gsplit_at_intersections:NN {cc}
```

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

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

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

```
\cs_new_protected_nopar:Npn \__spath_split_component_at_self_intersections:n #1
{
  \group_begin:
  \tl_set:Nn \l__spath_tmpe_tl {#1}

  % Remember if the component is closed
  \spath_finalaction:NV \l__spath_tmpa_tl \l__spath_tmpe_tl

  \bool_set:Nn \l__spath_closed_bool
  {
    \tl_if_eq_p:NN \l__spath_tmpa_tl \c_spath_closepath_tl
  }

  % Copy the path
  \tl_set:Nn \l__spath_tmpe_tl {#1}

  % Open the path
  \spath_open:N \l__spath_tmpe_tl
  % Ensure beziers don’t self-intersect
  \spath_split_curves:N \l__spath_tmpe_tl

  % Make a copy for later
  \tl_set_eq:NN \l__spath_tmpg_tl \l__spath_tmpe_tl

  % Clear some token lists and sequences
  \tl_clear:N \l__spath_tmpd_tl
  \seq_clear:N \l__spath_tmpb_seq
  \int_zero:N \l__spath_tmpa_int

  \pgfintersectionsortbyfirstpath

  % Split the path into a sequence of segments
  \spath_segments_to_seq:NV \l__spath_tmpa_seq \l__spath_tmpe_tl

  \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
  {
    \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
    {
      % Don’t intersect a segment with itself
      \int_compare:nF
      {
        ##1 == ####1
      }
      {
        \spath_intersect:nn {##2} {####2}

        \int_compare:nT {\pgfintersectionsolutions > 0}
        {
          % Find the times of the intersections on each path
          \int_step_inline:nnnn {1} {1} {\pgfintersectionsolutions}
          {
            \pgfintersectiongetsolutiontimes
            {########1}{\l__spath_tmpb_tl}{\l__spath_tmpc_tl}

            \bool_if:nT
            {
              !(
              \fp_compare_p:n { \l__spath_tmpb_tl > .99 }
              &&
              \int_compare_p:n {##1 + 1 == ####1}
              )
              &&
              !(
              \fp_compare_p:n { \l__spath_tmpb_tl < .01 }
              &&
              \int_compare_p:n {##1 - 1 == ####1}
              )
              &&
              !(
              \l__spath_closed_bool
              &&
              \fp_compare_p:n { \l__spath_tmpb_tl < .01 }
              &&
              \int_compare_p:n {##1 == 1}
              &&
              \int_compare_p:n {\seq_count:N \l__spath_tmpa_seq == ####1}
              )
              &&
              !(
              \l__spath_closed_bool
              &&
              \fp_compare_p:n { \l__spath_tmpb_tl > .99 }
              &&
              \int_compare_p:n {####1 == 1}
              &&
              \int_compare_p:n {\seq_count:N \l__spath_tmpa_seq == ##1}
              )
            }
            {
              \tl_set:Nx \l__spath_tmpa_tl
              {\fp_to_decimal:n {\l__spath_tmpb_tl +  ##1 - 1}}
              \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
            }
          }
        }
      }
    }
  }

  % Sort the sequence by reverse order along the path
  \seq_sort:Nn \l__spath_tmpb_seq
  {
    \fp_compare:nNnTF { ##1 } < { ##2 }
    { \sort_return_swapped: }
    { \sort_return_same: }
  }

```

```
\seq_get_left:NN \l__spath_tmpb_seq \l__spath_tmpa_tl
\fp_compare:nT
{
  \l__spath_tmpa_tl > \seq_count:N \l__spath_tmpa_seq - .01
}
{
  \bool_set_false:N \l__spath_closed_bool
}
\seq_get_right:NN \l__spath_tmpb_seq \l__spath_tmpa_tl
\fp_compare:nT
{
  \l__spath_tmpa_tl < .01
}
{
  \bool_set_false:N \l__spath_closed_bool
}

% Restore the original copy of the path
\tl_set_eq:NN \l__spath_tmpe_tl \l__spath_tmpg_tl

% Clear the token lists
\tl_clear:N \l__spath_tmpf_tl
\tl_clear:N \l__spath_tmph_tl
\tl_clear:N \l__spath_tmpg_tl

\tl_set:Nn \l__spath_tmpi_tl {-1}

\seq_map_inline:Nn \l__spath_tmpb_seq
{
  \tl_set:Nn \l__spath_tmpb_tl {##1}
  \tl_set_eq:NN \l__spath_tmpa_tl \l__spath_tmpb_tl
  \int_compare:nT
  {
    \fp_to_int:n {floor( \l__spath_tmpb_tl ) }
    =
    \fp_to_int:n {floor( \l__spath_tmpi_tl) }
  }
  {
    \tl_set:Nx \l__spath_tmpb_tl
    {
      \fp_eval:n {
        floor( \l__spath_tmpb_tl )
        +
        ( \l__spath_tmpb_tl - floor( \l__spath_tmpb_tl) )
        /
        ( \l__spath_tmpi_tl - floor( \l__spath_tmpi_tl) )
      }
    }
  }
  \tl_set_eq:NN \l__spath_tmpi_tl \l__spath_tmpa_tl

  \spath_split_at:NNVV
  \l__spath_tmpf_tl
  \l__spath_tmph_tl
```

```
    \l__spath_tmpe_tl
    \l__spath_tmpb_tl

    \tl_put_left:NV \l__spath_tmpg_tl \l__spath_tmph_tl
    \tl_set_eq:NN \l__spath_tmpe_tl \l__spath_tmpf_tl

  }

  \tl_put_left:NV \l__spath_tmpg_tl \l__spath_tmpe_tl

  \tl_if_empty:NT \l__spath_tmpg_tl
  {
    \tl_set_eq:NN \l__spath_tmpg_tl \l__spath_tmpe_tl
  }

  \spath_remove_empty_components:N \l__spath_tmpg_tl

  % Do something with closed
  \bool_if:NT \l__spath_closed_bool
  {
    \spath_join_component:Nn \l__spath_tmpg_tl {1}
  }

  \tl_gclear:N \g__spath_output_tl
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpg_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_split_component_at_self_intersections:Nn #1#2
{
  \__spath_split_component_at_self_intersections:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_split_component_at_self_intersections:Nn {NV}
\cs_new_protected_nopar:Npn \spath_split_component_at_self_intersections:N #1
{
  \spath_split_component_at_self_intersections:NV #1#1
}
\cs_generate_variant:Nn \spath_split_component_at_self_intersections:N {c}
\cs_new_protected_nopar:Npn \spath_gsplit_component_at_self_intersections:Nn #1#2
{
  \__spath_split_component_at_self_intersections:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gsplit_component_at_self_intersections:Nn {NV}
\cs_new_protected_nopar:Npn \spath_gsplit_component_at_self_intersections:N #1
{
  \spath_gsplit_component_at_self_intersections:NV #1#1
}
\cs_generate_variant:Nn \spath_gsplit_component_at_self_intersections:N {c}
```

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

`\spath_split_at_self_intersections:Nn`
`\spath_split_at_self_intersections:N`
`\spath_gsplit_at_self_intersections:Nn`
`\spath_gsplit_at_self_intersections:N`

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

```
\cs_new_protected_nopar:Npn \__spath_split_at_self_intersections:n #1
{
  \group_begin:
  \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
  \seq_clear:N \l__spath_tmpb_seq
  \seq_clear:N \l__spath_tmpc_seq

  % Iterate over the components of the original path.
  \bool_do_until:nn
  {
    \seq_if_empty_p:N \l__spath_tmpa_seq
  }
  {
    % Get the next component
    \seq_pop_left:NN \l__spath_tmpa_seq \l__spath_tmpa_tl
    % Copy for later
    \tl_set_eq:NN \l__spath_tmpc_tl \l__spath_tmpa_tl
    \int_compare:nT
    {
      \tl_count:N \l__spath_tmpa_tl > 3
    }
    {

      % Split against itself
      \spath_split_component_at_self_intersections:N \l__spath_tmpa_tl
      % Grab the rest of the path
      \tl_set:Nx \l__spath_tmpb_tl
      {
        \seq_use:Nn \l__spath_tmpb_seq {}
        \seq_use:Nn \l__spath_tmpa_seq {}
      }
      % Split against the rest of the path
      \spath_split_path_at_intersections:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
    }
    % Save the split path
    \seq_put_right:NV \l__spath_tmpc_seq \l__spath_tmpa_tl
    % Add the original copy to the sequence of processed components
    \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpc_tl
  }

  \tl_gclear:N \g__spath_output_tl
  \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpc_seq {} }
  \group_end:
}
\cs_generate_variant:Nn \__spath_split_at_self_intersections:n {V, v}
\cs_new_protected_nopar:Npn \spath_split_at_self_intersections:Nn #1#2
{
  \__spath_split_at_self_intersections:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_split_at_self_intersections:Nn {NV, cn, cV, cv}
```

```
\cs_new_protected_nopar:Npn \spath_split_at_self_intersections:N #1
{
  \spath_split_at_self_intersections:NV #1#1
}
\cs_generate_variant:Nn \spath_split_at_self_intersections:N {c}
\cs_new_protected_nopar:Npn \spath_gsplit_at_self_intersections:Nn #1#2
{
  \__spath_split_at_self_intersections:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gsplit_at_self_intersections:Nn {NV, cn, cV, cv}
\cs_new_protected_nopar:Npn \spath_gsplit_at_self_intersections:N #1
{
  \spath_gsplit_at_self_intersections:NV #1#1
}
\cs_generate_variant:Nn \spath_gsplit_at_self_intersections:N {c}
```

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

`\spath_join_component:Nnn`
`\spath_join_component:Nn`
`\spath_gjoin_component:Nnn`
`\spath_gjoin_component:Nn`

Join the specified component of the spath to its predecessor.

```
\cs_new_protected_nopar:Npn \__spath_join_component:nn #1#2
{
  \group_begin:
  \spath_numberofcomponents:Nn \l__spath_tmpa_int {#1}

  \bool_if:nTF
  {
    \int_compare_p:n { #2 >= 1 }
    &&
    \int_compare_p:n { #2 <= \l__spath_tmpa_int }
  }
  {
    \int_compare:nTF
    {
      #2 == 1
    }
    {
      \int_compare:nTF
      {
        \l__spath_tmpa_int == 1
      }
      {
        \tl_set:Nn \l__spath_tmpa_tl {#1}
        \spath_initialpoint:Nn \l__spath_tmpb_tl {#1}
        \tl_put_right:NV \l__spath_tmpa_tl \c_spath_closepath_tl
        \tl_put_right:NV \l__spath_tmpa_tl \l__spath_tmpb_tl
        \tl_gclear:N \g__spath_output_tl
        \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
      }
      {
        \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
        \seq_pop_left:NN \l__spath_tmpa_seq \l__spath_tmpa_tl

        \prg_replicate:nn {3}
        {
          \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
        }

        \seq_put_right:NV \l__spath_tmpa_seq \l__spath_tmpa_tl

        \tl_gclear:N \g__spath_output_tl
        \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpa_seq {}}
      }
    }
    {
      \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}

      \seq_clear:N \l__spath_tmpb_seq
      \seq_map_indexed_inline:Nn \l__spath_tmpa_seq
      {
        \tl_set:Nn \l__spath_tmpa_tl {##2}
        \int_compare:nT {##1 = #2}
        {
          \prg_replicate:nn {3}
          {
            \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
          }
        }
        \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
      }

      \tl_gclear:N \g__spath_output_tl
      \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpb_seq {}}
    }
  }
  {
    \tl_gclear:N \g__spath_output_tl
    \tl_gset:Nn \g__spath_output_tl {#1}
  }

  \group_end:
}
\cs_new_protected_nopar:Npn \spath_join_component:Nnn #1#2#3
{
  \__spath_join_component:nn {#2}{#3}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_join_component:Nnn {NVn, NVV}
\cs_new_protected_nopar:Npn \spath_join_component:Nn #1#2
{
  \spath_join_component:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_join_component:Nn {cn, NV, cV}
\cs_new_protected_nopar:Npn \spath_gjoin_component:Nnn #1#2#3
{
  \__spath_join_component:nn {#2}{#3}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_gjoin_component:Nnn {NVn, NVV}
\cs_new_protected_nopar:Npn \spath_gjoin_component:Nn #1#2
{
  \spath_gjoin_component:NVn #1#1{#2}
}
\cs_generate_variant:Nn \spath_gjoin_component:Nn {cn, NV, cV}
```

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

\spath_spot_weld_components:Nn
\spath_spot_weld_components:N
\spath_spot_gweld_components:Nn
\spath_spot_gweld_components:N

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

```
\cs_new_protected_nopar:Npn \__spath_spot_weld_components:n #1
{
  \group_begin:
  \dim_zero:N \l__spath_move_x_dim
  \dim_zero:N \l__spath_move_y_dim

  \spath_components_to_seq:Nn \l__spath_tmpa_seq {#1}
  \seq_clear:N \l__spath_tmpb_seq
  \dim_set:Nn \l__spath_move_x_dim {\tl_item:nn {#1} {2} + 10 pt}
  \dim_set:Nn \l__spath_move_y_dim {\tl_item:nn {#1} {3} + 10 pt}

  \int_set:Nn \l__spath_tmpa_int {\seq_count:N \l__spath_tmpa_seq}

  \seq_map_inline:Nn \l__spath_tmpa_seq
  {
    \tl_set:Nn \l__spath_tmpa_tl {##1}
    \bool_if:nT
    {
      \dim_compare_p:n
      {
        \dim_abs:n
        {\l__spath_move_x_dim - \tl_item:Nn \l__spath_tmpa_tl {2} }
        < 0.01pt
      }
      &&
      \dim_compare_p:n
      {
        \dim_abs:n
        {\l__spath_move_y_dim - \tl_item:Nn \l__spath_tmpa_tl {3} }
        < 0.01pt
      }
    }
    {
      \prg_replicate:nn {3}
      {
        \tl_set:Nx \l__spath_tmpa_tl {\tl_tail:N \l__spath_tmpa_tl}
      }
      \int_decr:N \l__spath_tmpa_int
    }
    \tl_reverse:N \l__spath_tmpa_tl
    \dim_set:Nn \l__spath_move_x_dim {\tl_item:Nn \l__spath_tmpa_tl {2}}
    \dim_set:Nn \l__spath_move_y_dim {\tl_item:Nn \l__spath_tmpa_tl {1}}
    \tl_reverse:N \l__spath_tmpa_tl
    \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpa_tl
  }

  \tl_set:Nx \l__spath_tmpa_tl {\seq_use:Nn \l__spath_tmpb_seq {} }
  \spath_components_to_seq:NV \l__spath_tmpb_seq \l__spath_tmpa_tl


  \spath_initialpoint:Nn \l__spath_tmpa_tl {#1}
  \spath_finalpoint:Nn \l__spath_tmpb_tl {#1}

  \bool_if:nT
  {
    \dim_compare_p:n
    {
      \dim_abs:n
      {
        \tl_item:Nn \l__spath_tmpa_tl {1} - \tl_item:Nn \l__spath_tmpb_tl {1}
      }
      <
      0.01pt
    }
    &&
    \dim_compare_p:n
    {
      \dim_abs:n
      {
        \tl_item:Nn \l__spath_tmpa_tl {2} - \tl_item:Nn \l__spath_tmpb_tl {2}
      }
      <
      0.01pt
    }
  }
  {
    \int_compare:nTF
    {
      \seq_count:N \l__spath_tmpb_seq > 1
    }
    {
      \seq_pop_left:NN \l__spath_tmpb_seq \l__spath_tmpb_tl

      \prg_replicate:nn {3}
      {
        \tl_set:Nx \l__spath_tmpb_tl {\tl_tail:N \l__spath_tmpb_tl}
      }
      \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpb_tl
    }
    {
      \tl_set:NV \l__spath_tmpb_tl \c_spath_closepath_tl
      \tl_put_right:Nx \l__spath_tmpb_tl
      {
        { \tl_item:Nn \l__spath_tmpa_tl {1} }
        { \tl_item:Nn \l__spath_tmpa_tl {2} }
      }
      \seq_put_right:NV \l__spath_tmpb_seq \l__spath_tmpb_tl
    }
  }

  \tl_gset:Nx \g__spath_output_tl {\seq_use:Nn \l__spath_tmpb_seq {}}
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_spot_weld_components:Nn #1#2
{
  \__spath_spot_weld_components:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_spot_weld_components:Nn {NV, cV, cn}
\cs_new_protected_nopar:Npn \spath_spot_weld_components:N #1
{
  \spath_spot_weld_components:NV #1#1
}
\cs_generate_variant:Nn \spath_spot_weld_components:N {c}
\cs_new_protected_nopar:Npn \spath_spot_gweld_components:Nn #1#2
{
  \__spath_spot_weld_components:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_generate_variant:Nn \spath_spot_gweld_components:Nn {NV, cV, cn}
\cs_new_protected_nopar:Npn \spath_spot_gweld_components:N #1
{
  \spath_spot_gweld_components:NV #1#1
}
\cs_generate_variant:Nn \spath_spot_gweld_components:N {c}
```

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

3.8 Exporting Commands

`\spath_convert_to_svg:Nn`
`\spath_gconvert_to_svg:Nn`

Convert the soft path to an SVG document.

```
\cs_new_protected_nopar:Npn \__spath_convert_to_svg:n #1
{
  \group_begin:
  \tl_clear:N \l__spath_tmpa_tl
  \tl_put_right:Nn \l__spath_tmpa_tl
  {
    <?xml~ version="1.0"~ standalone="no"?>
    \iow_newline:
    <!DOCTYPE~ svg~ PUBLIC~ "-//W3C//DTD SVG 1.1//EN"~
    "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
    \iow_newline:
    <svg~ xmlns="http://www.w3.org/2000/svg"~ version="1.1"~viewBox="
  }

  \spath_minbb:Nn \l__spath_tmpb_tl {#1}
```

```
\spath_maxbb:Nn \l__spath_tmpc_tl {#1}
\tl_put_right:Nx \l__spath_tmpa_tl
{
  \dim_to_decimal:n
  {
    \tl_item:Nn \l__spath_tmpb_tl {1} - 10pt
  }
  \exp_not:n {~}
  \dim_to_decimal:n
  {
    \tl_item:Nn \l__spath_tmpb_tl {2} - 10pt
  }
  \exp_not:n {~}
  \dim_to_decimal:n
  {
    \tl_item:Nn \l__spath_tmpc_tl {1}
    -
    \tl_item:Nn \l__spath_tmpb_tl {1}
    + 20pt
  }
  \exp_not:n {~}
  \dim_to_decimal:n
  {
    \tl_item:Nn \l__spath_tmpc_tl {2}
    -
    \tl_item:Nn \l__spath_tmpb_tl {2}
    + 20pt
  }
}

\tl_put_right:Nn \l__spath_tmpa_tl
{
  ">
  \iow_newline:
  <path~ d="
}
\tl_set:Nn \l__spath_tmpc_tl {use:n}
\tl_map_inline:nn {#1}
{
  \tl_set:Nn \l__spath_tmpb_tl {##1}
  \token_case_meaning:NnF \l__spath_tmpb_tl
  {
    \c_spath_moveto_tl
    {
      \tl_put_right:Nn \l__spath_tmpa_tl {M~}
      \tl_set:Nn \l__spath_tmpc_tl {use:n}
    }
    \c_spath_lineto_tl
    {
      \tl_put_right:Nn \l__spath_tmpa_tl {L~}
      \tl_set:Nn \l__spath_tmpc_tl {use:n}
    }
    \c_spath_closepath_tl
    {
        \tl_put_right:Nn \l__spath_tmpa_tl {Z~}
        \tl_set:Nn \l__spath_tmpc_tl {use_none:n}
      }
      \c_spath_curvetoa_tl
      {
        \tl_put_right:Nn \l__spath_tmpa_tl {C~}
        \tl_set:Nn \l__spath_tmpc_tl {use:n}
      }
      \c_spath_curvetob_tl {
        \tl_set:Nn \l__spath_tmpc_tl {use:n}
      }
      \c_spath_curveto_tl {
        \tl_set:Nn \l__spath_tmpc_tl {use:n}
      }
    }
    {
      \tl_put_right:Nx
      \l__spath_tmpa_tl
      {\use:c { \l__spath_tmpc_tl } {\dim_to_decimal:n {##1}} ~}
    }
  }
  \tl_put_right:Nn \l__spath_tmpa_tl
  {
    "~ fill="none"~ stroke="black"~ />
    \iow_newline:
    </svg>
    \iow_newline:
  }
  \tl_gset_eq:NN \g__spath_output_tl \l__spath_tmpa_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \spath_convert_to_svg:Nn #1#2
{
  \__spath_convert_to_svg:n {#2}
  \tl_set_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
\cs_new_protected_nopar:Npn \spath_gconvert_to_svg:Nn #1#2
{
  \__spath_convert_to_svg:n {#2}
  \tl_gset_eq:NN #1 \g__spath_output_tl
  \tl_gclear:N \g__spath_output_tl
}
```

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

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

```
\iow_new:N \g__spath_stream
\cs_new_protected_nopar:Npn \spath_export_to_svg:nn #1#2
{
  \group_begin:
  \spath_convert_to_svg:Nn \l__spath_tmpa_tl {#2}
  \iow_open:Nn \g__spath_stream {#1 .svg}
  \iow_now:Nx \g__spath_stream
  {
    \tl_use:N \l__spath_tmpa_tl
  }
  \iow_close:N \g__spath_stream
  \group_end:
}
\cs_generate_variant:Nn \spath_export_to_svg:nn {nv, nV}
```

(*End of definition for* `\spath_export_to_svg:nn`.)

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

```
\cs_new_protected_nopar:Npn \spath_show:n #1
{
  \int_step_inline:nnnn {1} {3} {\tl_count:n {#1}}
  {
    \iow_term:x {
      \tl_item:nn {#1} {##1}
      {\tl_item:nn {#1} {##1+1}}
      {\tl_item:nn {#1} {##1+2}}
    }
  }
}
\cs_generate_variant:Nn \spath_show:n {V, v}
```

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

3.9 PGF and TikZ Interface Functions

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

`\spath_get_current_path:N`
`\spath_gget_current_path:N`

Grab the current soft path from PGF.

```
\cs_new_protected_nopar:Npn \spath_get_current_path:N #1
{
  \pgfsyssoftpath@getcurrentpath #1
}
\cs_generate_variant:Nn \spath_get_current_path:N {c}

\cs_new_protected_nopar:Npn \spath_gget_current_path:N #1
{
  \pgfsyssoftpath@getcurrentpath #1
  \tl_gset_eq:NN #1 #1
}
\cs_generate_variant:Nn \spath_gget_current_path:N {c}
```

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

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

```
\cs_new_protected_nopar:Npn \spath_protocol_path:n #1
{
  \spath_minbb:Nn \l__spath_tmpa_tl {#1}
  \dim_set:Nn \l__spath_tmpa_dim {\tl_item:Nn \l__spath_tmpa_tl {1}}
  \dim_set:Nn \l__spath_tmpb_dim {\tl_item:Nn \l__spath_tmpa_tl {2}}
  \pgf@protocolsizes\l__spath_tmpa_dim\l__spath_tmpb_dim

  \spath_maxbb:Nn \l__spath_tmpa_tl {#1}
  \dim_set:Nn \l__spath_tmpa_dim {\tl_item:Nn \l__spath_tmpa_tl {1}}
  \dim_set:Nn \l__spath_tmpb_dim {\tl_item:Nn \l__spath_tmpa_tl {2}}
  \pgf@protocolsizes\l__spath_tmpa_dim\l__spath_tmpb_dim
}
\cs_generate_variant:Nn \spath_protocol_path:n {V}
```

(*End of definition for* `\spath_protocol_path:n`.)

`\spath_set_current_path:n`
`\spath_set_current_path:N`

Sets the current path to the specified soft path.

```
\cs_new_protected_nopar:Npn \spath_set_current_path:n #1
{
  \spath_protocol_path:n {#1}
  \tl_set:Nn \l__spath_tmpa_tl {#1}
  \pgfsyssoftpath@setcurrentpath\l__spath_tmpa_tl
}
\cs_new_protected_nopar:Npn \spath_set_current_path:N #1
{
  \spath_protocol_path:V #1
  \pgfsyssoftpath@setcurrentpath #1
}
\cs_generate_variant:Nn \spath_set_current_path:N {c}
```

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

`\spath_use_path:nn`

Uses the given soft path at the PGF level.

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

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

`\spath_tikz_path:nn`

Uses the given soft path at the TikZ level.

```
\cs_new_protected_nopar:Npn \spath_tikz_path:nn #1#2
{
  \tl_if_empty:nF {#2}
  {
    \path[#1] \pgfextra{
      \spath_set_current_path:n {#2}
      \tl_put_left:Nn \tikz@preactions {\def\tikz@actions@path{#2}}
    };
  }
}
\cs_generate_variant:Nn \spath_tikz_path:nn {Vn, VV, nv, Vv, nV}
```

(*End of definition for* `\spath_tikz_path:nn`.)

`\spath_set_tikz_data:n`

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

```
\cs_new_protected_nopar:Npn \spath_set_tikz_data:n #1
{
  \spath_finalpoint:Nn \l__spath_tmpa_tl {#1}
  \tl_set:Nx \l__spath_tmpa_tl
  {
    \exp_not:c {pgf@x}=\tl_item:Nn \l__spath_tmpa_tl {1} \relax
    \exp_not:c {pgf@y}=\tl_item:Nn \l__spath_tmpa_tl {2} \relax
  }
  \use:c {pgf@process}{%
    \tl_use:N \l__spath_tmpa_tl
    \pgftransforminvert
    \use:c {pgf@pos@transform@glob}
  }
  \tl_set:Nx \l__spath_tmpa_tl
  {
    \exp_not:c {tikz@lastx}=\exp_not:c {pgf@x} \relax
    \exp_not:c {tikz@lasty}=\exp_not:c {pgf@y} \relax
    \exp_not:c {tikz@lastxsaved}=\exp_not:c {pgf@x} \relax
    \exp_not:c {tikz@lastysaved}=\exp_not:c {pgf@y} \relax
  }
  \tl_use:N \l__spath_tmpa_tl
  \spath_finalmovepoint:Nn \l__spath_tmpa_tl {#1}
  \bool_if:NT \l_spath_movetorelevant_bool
  {
    \ifpgfsyssoftpathmovetorelevant%
    \tl_gset_eq:cN {pgfsyssoftpath@lastmoveto} \l__spath_tmpa_tl
    \fi
  }
  \tl_set:Nx \l__spath_tmpa_tl
  {
    \exp_not:c {pgf@x}=\tl_item:Nn \l__spath_tmpa_tl {1} \relax
    \exp_not:c {pgf@y}=\tl_item:Nn \l__spath_tmpa_tl {2} \relax
  }
  \use:c {pgf@process}{%
    \tl_use:N \l__spath_tmpa_tl
    \pgftransforminvert
    \use:c {pgf@pos@transform@glob}
  }
  \bool_if:NT \l_spath_movetorelevant_bool
  {
    \dim_if_exist:cT {tikz@lastmovetox}
    {
      \tl_set:Nx \l__spath_tmpa_tl
      {
        \exp_not:c {tikz@lastmovetox}=\exp_not:c {pgf@x} \relax
        \exp_not:c {tikz@lastmovetoy}=\exp_not:c {pgf@y} \relax
      }
      \tl_use:N \l__spath_tmpa_tl
    }
  }
  \tl_clear_new:c {tikz@timer}
  \tl_set:cn {tikz@timer}
  {
    \pgftransformreset
    \spath_reallength:Nn \l__spath_tmpa_int {#1}
    \tl_set_eq:Nc \l__spath_tmpb_tl {tikz@time}
    \tl_set:Nx \l__spath_tmpb_tl
    {\fp_to_decimal:n {(\l__spath_tmpb_tl) * (\l__spath_tmpa_int)}}
    \spath_transformation_at:NnV \l__spath_tmpc_tl {#1} \l__spath_tmpb_tl

    \tl_set:Nx \l__spath_tmpa_tl
    {
      \exp_not:N \pgfpoint
      { \tl_item:Nn \l__spath_tmpc_tl {5} }
      { \tl_item:Nn \l__spath_tmpc_tl {6} }
    }
    \exp_args:NV \pgftransformshift \l__spath_tmpa_tl

    \ifpgfresetnontranslationattime
    \pgftransformresetnontranslations
    \fi

    \ifpgfslopedattime

    \tl_set:Nx \l__spath_tmpa_tl
    {
      { \tl_item:Nn \l__spath_tmpc_tl {1} }
      { \tl_item:Nn \l__spath_tmpc_tl {2} }
      { \tl_item:Nn \l__spath_tmpc_tl {3} }
      { \tl_item:Nn \l__spath_tmpc_tl {4} }
    }
    \ifpgfallowupsidedownattime
    \else
    \fp_compare:nT { \tl_item:Nn \l__spath_tmpc_tl {4} < 0}
    {
      \tl_set:Nx \l__spath_tmpa_tl
      {
        { \fp_eval:n { - (\tl_item:Nn \l__spath_tmpc_tl {1})} }
        { \fp_eval:n { - (\tl_item:Nn \l__spath_tmpc_tl {2})} }
        { \fp_eval:n { - (\tl_item:Nn \l__spath_tmpc_tl {3})} }
        { \fp_eval:n { - (\tl_item:Nn \l__spath_tmpc_tl {4})} }
      }
    }
    \fi
    \tl_put_right:Nn \l__spath_tmpa_tl {{\pgfpointorigin}}
    \exp_last_unbraced:NV \pgftransformcm \l__spath_tmpa_tl
    \fi
  }
}
\cs_generate_variant:Nn \spath_set_tikz_data:n {V, v}
```

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

4 The TikZ interface

```
⟨@@=tikzspath⟩
```

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

```
\RequirePackage{spath3}
\RequirePackage{expl3}
\ExplSyntaxOn
```

```

\tl_new:N \l__tikzspath_tmpa_tl
\tl_new:N \l__tikzspath_tmpb_tl
\tl_new:N \l__tikzspath_tmpc_tl
\tl_new:N \l__tikzspath_tmpd_tl
\tl_new:N \l__tikzspath_tmpe_tl
\tl_new:N \l__tikzspath_tmpf_tl

\int_new:N \l__tikzspath_tmpa_int
\seq_new:N \l__tikzspath_tmpa_seq
\seq_new:N \l__tikzspath_tmpb_seq
\seq_new:N \l__tikzspath_tmpc_seq
\seq_new:N \l__tikzspath_tmpd_seq

\tl_new:N \l__tikzspath_current_tl
\tl_new:N \l__tikzspath_reverse_tl
\tl_new:N \l__tikzspath_prefix_tl
\tl_new:N \l__tikzspath_suffix_tl
\tl_new:N \g__tikzspath_smuggle_tl
\tl_new:N \g__tikzspath_output_tl
\tl_new:N \l__tikzspath_check_tl
\clist_new:N \g__tikzspath_output_clist
\seq_new:N \g__tikzspath_tmpa_seq
\seq_new:N \g__tikzspath_tmpb_seq
\seq_new:N \g__tikzspath_output_seq
\bool_new:N \l__tikzspath_draft_bool
```

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

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

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

```
\tl_set:Nn \l__tikzspath_prefix_tl {tikz@intersect@path@name@}
\tl_set:Nn \l__tikzspath_suffix_tl {}
```

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

Interestingly, the intersections library doesn't clear its naming code once it is used meaning that it keeps resetting the definition of a path back to its original one every time a path command is called.

Also, when the hook is restored outside a scope then no check is made to ensure that the inner one was actually invoked. This can cause issues when the syntax `\tikz .. ;` is used since the end of the path coincides with the end of the picture.

```
\tl_new:N \g__tikzspath_tikzfinish_tl
```

```
\tl_new:N \l__tikzspath_tikzfinish_outside_tl
\cs_new_protected_nopar:Npn \spath_at_end_of_path:
{
  \tl_use:N \g__tikzspath_tikzfinish_tl
  \tl_gclear:N \g__tikzspath_tikzfinish_tl
}
\tl_put_right:Nn \tikz@finish {\spath_at_end_of_path:}

\tikzset{
  every~ scope/.append~ style={
    execute~ at~ begin~ scope={
      \tl_set_eq:NN \l__tikzspath_tikzfinish_outside_tl \g__tikzspath_tikzfinish_tl
    },
    execute~ at~ end~ scope={
      \tl_use:N \g__tikzspath_tikzfinish_tl
      \tl_gclear:N \g__tikzspath_tikzfinish_tl
      \tl_gset_eq:NN \g__tikzspath_tikzfinish_tl \l__tikzspath_tikzfinish_outside_tl
    },
  },
}
```

This is for delaying something until the path is fully constructed (but no later), sometimes useful to be able to specify this in the path options rather than directly at the end of the path.

```
\tl_new:N \l__tikzspath_tikzpath_finish_tl

\cs_new_protected_nopar:Npn \__tikzspath_at_end_of_path_construction:
{
  \tl_use:N \l__tikzspath_tikzpath_finish_tl
  \tl_clear:N \l__tikzspath_tikzpath_finish_tl
}

\tl_put_left:Nn \tikz@finish {\__tikzspath_at_end_of_path_construction:}
```

Code for saving a path

```
\cs_new_protected_nopar:Npn \spath_save_path:Nn #1#2
{
  \tl_if_empty:NF \g__tikzspath_tikzfinish_tl
  {
    \tl_use:N \g__tikzspath_tikzfinish_tl
  }
  \tl_gput_right:Nn \g__tikzspath_tikzfinish_tl
  {
    \tl_clear_new:N #1
    \tl_set:Nn #1 {#2}
  }
}
\cs_generate_variant:Nn \spath_save_path:Nn {cn, NV, cV}

\cs_new_protected_nopar:Npn \spath_gsave_path:Nn #1#2
{
  \tl_gput_right:Nn \g__tikzspath_tikzfinish_tl
  {
    \tl_gclear_new:N #1
    \tl_gset:Nn #1 {#2}
  }
}
\cs_generate_variant:Nn \spath_gsave_path:Nn {cn, NV, cV}
```

__tikzspath_process_tikz_point:Nn

Process a point via TikZ and store the resulting dimensions.

```
\cs_new_protected_nopar:Npn \__tikzspath_process_tikz_point:Nn #1#2
{
  \group_begin:
  \use:c {tikz@scan@one@point} \use:n #2 \scan_stop:
  \tl_gset:Nx \g__tikzspath_output_tl
  {
    { \dim_use:c {pgf@x} }
    { \dim_use:c {pgf@y} }
  }
  \group_end:
  \tl_set_eq:NN #1 \g__tikzspath_output_tl
  \tl_gclear:N \g__tikzspath_output_tl
}
```

(*End of definition for* `\__tikzspath_process_tikz_point:Nn`*.*)

`\__tikzspath_tikzset:n`

Wrapper around `\tikzset` for expansion.

```
\cs_set_eq:NN \__tikzspath_tikzset:n \tikzset
\cs_generate_variant:Nn \__tikzspath_tikzset:n {V, v}
```

(*End of definition for* `\__tikzspath_tikzset:n`*.*)

s:nnnn␣__tikzspath_check_three_paths:nnnnn

Given a path name as the second argument, check if it exists and is not empty, and if so reinsert it after the first argument. The third argument is code to be executed in case of a missing or empty path.

```
\cs_new_protected_nopar:Npn \__tikzspath_check_path:nnn #1#2#3
{
  \tl_set:Nn \l__tikzspath_check_tl {#3}
  \tl_if_exist:cTF {\__tikzspath_path_name:n {#2}}
  {
    \tl_if_empty:cTF {\__tikzspath_path_name:n {#2}}
    {
      \msg_warning:nnn { spath3 } { empty soft path } { #2 }
    }
    {
      \tl_set:Nn \l__tikzspath_check_tl {
        #1 {\__tikzspath_path_name:n {#2}}
      }
    }
  }
  {
    \msg_warning:nnx { spath3 } { missing soft path } { #2 }
  }
  \tl_use:N \l__tikzspath_check_tl
}
\cs_new_protected_nopar:Npn \__tikzspath_check_two_paths:nnnn #1#2#3#4
{
  \__tikzspath_check_path:nnn {
    \__tikzspath_check_path:nnn {#1}{#2}{#4}
  }{#3}{#4}
}
\cs_new_protected_nopar:Npn \__tikzspath_check_three_paths:nnnnn #1#2#3#4#5
{
  \__tikzspath_check_path:nnn {
    \__tikzspath_check_path:nnn {
      \__tikzspath_check_path:nnn {#1}{#2}{#5}
    }{#3}{#5}
  }{#4}{#5}
}
\cs_generate_variant:Nn \__tikzspath_check_path:nnn {nVn}
\cs_generate_variant:Nn \__tikzspath_check_two_paths:nnnn {nnVn}
```

(*End of definition for* `\__tikzspath_check_path:nnn` `\__tikzspath_check_two_paths:nnnn` `\__tikzspath_check_three_paths:nnnnn`.)

h_maybe_current_two_paths_reuse_second:nnnn

If the named path is "current" then get the current path and use that. The second version puts the resulting path back as the current path.

```
\cs_new_protected_nopar:Npn \__tikzspath_maybe_current_path:nn #1#2
{
  \tl_if_eq:nnT {#2} {current}
  {
    \spath_get_current_path:c {\__tikzspath_path_name:n {#2}}
  }
  #1 {#2}
}
\cs_new_protected_nopar:Npn \__tikzspath_maybe_current_path_reuse:nnn #1#2#3
{
  \bool_set_true:N \l_spath_movetorelevant_bool
  \tl_if_eq:nnT {#2} {current}
  {
    \spath_get_current_path:c {\__tikzspath_path_name:n {#2}}
  }
  #1 {#2} #3
  \tl_if_eq:nnT {#2} {current}
  {
    \tl_if_empty:cF {\__tikzspath_path_name:n {#2}}
    {
      \spath_set_current_path:c {\__tikzspath_path_name:n {#2}}
      \spath_set_tikz_data:v {\__tikzspath_path_name:n {#2}}
    }
  }
}
\cs_new_protected_nopar:Npn \__tikzspath_maybe_current_two_paths_reuse_both:nnnn #1#2#3#4
{
  \bool_set_true:N \l_spath_movetorelevant_bool
  \tl_if_eq:nnT {#2} {current}
  {
    \spath_get_current_path:c {\__tikzspath_path_name:n {#2}}
  }
  \tl_if_eq:nnT {#3} {current}
  {
    \spath_get_current_path:c {\__tikzspath_path_name:n {#3}}
  }
  #1 {#2} {#3} #4
  \tl_if_eq:nnT {#2} {current}
  {
    \tl_if_empty:cF {\__tikzspath_path_name:n {#2}}
    {
      \spath_set_current_path:c {\__tikzspath_path_name:n {#2}}
      \spath_set_tikz_data:v {\__tikzspath_path_name:n {#2}}
    }
  }
  \tl_if_eq:nnT {#3} {current}
  {
    \tl_if_empty:cF {\__tikzspath_path_name:n {#3}}
    {
      \spath_set_current_path:c {\__tikzspath_path_name:n {#3}}
      \spath_set_tikz_data:v {\__tikzspath_path_name:n {#3}}
    }
  }
}
\cs_new_protected_nopar:Npn \__tikzspath_maybe_current_two_paths_reuse_first:nnnn #1#2#3#4
{
  \bool_set_true:N \l_spath_movetorelevant_bool
  \tl_if_eq:nnT {#2} {current}
  {
    \spath_get_current_path:c {\__tikzspath_path_name:n {#2}}
  }
  \tl_if_eq:nnT {#3} {current}
  {
    \spath_get_current_path:c {\__tikzspath_path_name:n {#3}}
  }
  #1 {#2} {#3} #4
  \tl_if_eq:nnT {#2} {current}
  {
    \tl_if_empty:cF {\__tikzspath_path_name:n {#2}}
    {
      \spath_set_current_path:c {\__tikzspath_path_name:n {#2}}
      \spath_set_tikz_data:v {\__tikzspath_path_name:n {#2}}
    }
  }
}
\cs_new_protected_nopar:Npn \__tikzspath_maybe_current_two_paths_reuse_second:nnnn #1#2#3#4
{
  \bool_set_true:N \l_spath_movetorelevant_bool
  \tl_if_eq:nnT {#2} {current}
  {
    \spath_get_current_path:c {\__tikzspath_path_name:n {#2}}
  }
  \tl_if_eq:nnT {#3} {current}
  {
    \spath_get_current_path:c {\__tikzspath_path_name:n {#3}}
  }
  #1 {#2} {#3} #4
  \tl_if_eq:nnT {#3} {current}
  {
    \tl_if_empty:cF {\__tikzspath_path_name:n {#3}}
    {
      \spath_set_current_path:c {\__tikzspath_path_name:n {#3}}
      \spath_set_tikz_data:v {\__tikzspath_path_name:n {#3}}
    }
  }
}
\cs_generate_variant:Nn \__tikzspath_maybe_current_path:nn {nV}
\cs_generate_variant:Nn \__tikzspath_maybe_current_path_reuse:nnn {nVn}
```

(*End of definition for `\__tikzspath_maybe_current_path:nn` `\__tikzspath_maybe_current_path_reuse:nnn` `\__tikzspath_maybe_current_two_paths_reuse_both:nnnn` `\__tikzspath_maybe_current_two_paths_reuse_first:nnnn` `\__tikzspath_maybe_current_two_paths_reuse_second:nnnn`.*)

`\__tikzspath_seq_from_foreach:NNn` Convert a PGF foreach list, as the third argument, to a sequence. The second argument is the maximum number on the list.

```
\cs_new_protected_nopar:Npn \__tikzspath_seq_from_foreach:Nnn #1#2#3
{
  \group_begin:
  \seq_gclear:N \g__tikzspath_output_seq

  \tl_if_empty:nTF {#3}
  {
    \int_step_inline:nnnn {1}{1} {#2}
    {
      \seq_gput_right:Nn \g__tikzspath_output_seq {##1}
    }
  }
  {
    \foreach \l__tikzspath_tmpa_tl in {#3}
    {
      \int_compare:nTF { \l__tikzspath_tmpa_tl > 0 }
      {
        \seq_gput_right:NV \g__tikzspath_output_seq \l__tikzspath_tmpa_tl
      }
      {
        \seq_gput_right:Nx \g__tikzspath_output_seq
        {\int_eval:n {#2 - \l__tikzspath_tmpa_tl}}
      }
    }
    \seq_gsort:Nn \g__tikzspath_output_seq
    {
      \int_compare:nNnTF {##1} < {##2}
      { \sort_return_same: }
      { \sort_return_swapped: }
    }
  }
  \group_end:
  \seq_set_eq:NN #1 \g__tikzspath_output_seq
  \seq_gclear:N \g__tikzspath_output_seq
}
\cs_generate_variant:Nn \__tikzspath_seq_from_foreach:Nnn {NVV, NVn}
%
```

(*End of definition for `\__tikzspath_seq_from_foreach:NNn`.*)

`\__tikzspath_path_name:n` Wrap the argument in the prefix and suffix to generate the proper name.

```
\cs_new:Npn \__tikzspath_path_name:n #1
{
  \tl_use:N \l__tikzspath_prefix_tl
  #1
  \tl_use:N \l__tikzspath_suffix_tl
}
\cs_generate_variant:Nn \__tikzspath_path_name:n {V}
```

(*End of definition for* `\__tikzspath_path_name:n`.)

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

```
\bool_new:N \l__tikzspath_reverse_bool
\bool_new:N \l__tikzspath_weld_bool
\bool_new:N \l__tikzspath_move_bool
\bool_new:N \l__tikzspath_global_bool
\bool_new:N \l__tikzspath_current_transformation_bool
\tl_new:N \l__tikzspath_joinpath_tl
\tl_new:N \l__tikzspath_transformation_tl

\cs_new_protected_nopar:Npn \__tikzspath_set_bool:Nn #1#2
{
  \tl_if_eq:nnTF {#2}{false}
  {
    \bool_set_false:N #1
  }
  {
    \bool_set_true:N #1
  }
}
\tikzset {
  spath/join/.is~ family,
  spath/join/.cd,
  reverse/.code = {
    \__tikzspath_set_bool:Nn \l__tikzspath_reverse_bool {#1}
  },
  reverse/.default = true,
  weld/.code = {
    \__tikzspath_set_bool:Nn \l__tikzspath_weld_bool {#1}
  },
  weld/.default = true,
  no~ weld/.code = {
    \__tikzspath_set_bool:Nn \l__tikzspath_weld_bool {#1}
    \bool_set:Nn \l__tikzspath_weld_bool {! \l__tikzspath_weld_bool}
  },
  no~ weld/.default = true,
  move/.code = {
    \__tikzspath_set_bool:Nn \l__tikzspath_move_bool {#1}
  },
  move/.default = true,
  no~ move/.code = {
    \__tikzspath_set_bool:Nn \l__tikzspath_move_bool {#1}
    \bool_set:Nn \l__tikzspath_move_bool {! \l__tikzspath_move_bool}
  },
  no~ move/.default = true,
  global/.code = {
    \__tikzspath_set_bool:Nn \l__tikzspath_global_bool {#1}
  },
  global/.default = true,
  use~ current~ transformation/.code={
    \__tikzspath_set_bool:Nn \l__tikzspath_current_transformation_bool {#1}
  },
  use~ current~ transformation/.default = true,
  transform/.store~in=\l__tikzspath_transformation_tl,
  .unknown/.code = {
    \tl_set_eq:NN \l__tikzspath_joinpath_tl \pgfkeyscurrentname
  }
}
```

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

```
\cs_set_eq:NN \getComponentOf \clist_item:Nn
```

4.1 Helper Functions

`\__tikzspath_use_path:n`

Use a path, possibly manipulating it first.

```
\cs_new_protected_nopar:Npn \__tikzspath_use_path:n #1
{
  \tl_set:Nn \l__tikzspath_joinpath_tl {#1}
  \spath_get_current_path:N \l__tikzspath_current_tl

  \bool_if:NT \l__tikzspath_reverse_bool
  {
    \spath_reverse:N \l__tikzspath_joinpath_tl
  }

  \bool_if:NT \l__tikzspath_current_transformation_bool
  {
    \pgfgettransform \l__tikzspath_tmpa_tl
    \spath_transform:NV
    \l__tikzspath_joinpath_tl
    \l__tikzspath_tmpa_tl
  }

  \tl_if_empty:NF \l__tikzspath_transformation_tl
  {
    \group_begin:
    \pgftransformreset
    \__tikzspath_tikzset:V \l__tikzspath_transformation_tl
    \pgfgettransform \l__tikzspath_tmpa_tl
    \tl_gset:Nn \g__tikzspath_smuggle_tl
    {
      \spath_transform:Nnnnnnn
      \l__tikzspath_joinpath_tl
    }
    \tl_gput_right:NV \g__tikzspath_smuggle_tl \l__tikzspath_tmpa_tl
    \group_end:
    \tl_use:N \g__tikzspath_smuggle_tl
  }

  \bool_if:NT \l__tikzspath_move_bool
  {
    \tl_if_empty:NTF \l__tikzspath_current_tl
    {
      \tl_set:Nn \l__tikzspath_tmpc_tl { {0pt} {0pt} }
    }
    {
      \spath_finalpoint:NV
      \l__tikzspath_tmpc_tl
      \l__tikzspath_current_tl
    }
    \spath_translate_to:NV \l__tikzspath_joinpath_tl \l__tikzspath_tmpc_tl
  }

  \tl_if_empty:NTF \l__tikzspath_current_tl
  {
    \tl_if_empty:NTF \l__tikzspath_joinpath_tl
    {
      \tl_set_eq:NN \l__tikzspath_current_tl \c_spath_moveto_tl
      \tl_put_right:Nn \l__tikzspath_current_tl {{0pt}{0pt}}
    }
    {
      \tl_set_eq:NN \l__tikzspath_current_tl \l__tikzspath_joinpath_tl
    }
  }
  {

    \tl_clear:N \l__tikzspath_tmpa_tl
    \tl_set:Nn \l__tikzspath_tmpa_tl {spath_}

    \tl_put_right:Nn \l__tikzspath_tmpa_tl {append}

    \bool_if:NT \l__tikzspath_weld_bool
    {
      \tl_put_right:Nn \l__tikzspath_tmpa_tl {_no_move}
      \spath_numberofcomponents:NV \l__tikzspath_tmpa_int \l__tikzspath_joinpath_tl
      \int_compare:nT {\l__tikzspath_tmpa_int == 1}
      {
        \bool_set_false:N \l_spath_movetorelevant_bool
      }
    }
    \tl_put_right:Nn \l__tikzspath_tmpa_tl {:NV}

    \use:c {\tl_use:N \l__tikzspath_tmpa_tl }
    \l__tikzspath_current_tl
    \l__tikzspath_joinpath_tl
  }

  \spath_set_current_path:N \l__tikzspath_current_tl
```

```
  \spath_set_tikz_data:V \l__tikzspath_joinpath_tl
}
\cs_generate_variant:Nn \__tikzspath_use_path:n {V, v}
```

(*End of definition for* `\__tikzspath_use_path:n`*.*)

`\__tikzspath_join_with:nn`

```
\cs_new_protected_nopar:Npn \__tikzspath_join_with:Nn #1#2
{
  \tl_set:Nn \l__tikzspath_joinpath_tl {#2}

  \bool_if:NT \l__tikzspath_reverse_bool
  {
    \spath_reverse:N \l__tikzspath_joinpath_tl
  }

  \tl_if_empty:NF \l__tikzspath_transformation_tl
  {
    \group_begin:
    \pgftransformreset
    \__tikzspath_tikzset:V \l__tikzspath_transformation_tl
    \pgfgettransform \l__tikzspath_tmpa_tl
    \tl_gset:Nn \g__tikzspath_smuggle_tl
    {
      \spath_transform:Nnnnnnn
      \l__tikzspath_joinpath_tl
    }
    \tl_gput_right:NV \g__tikzspath_smuggle_tl \l__tikzspath_tmpa_tl
    \group_end:
    \tl_use:N \g__tikzspath_smuggle_tl
  }

  \bool_if:NT \l__tikzspath_move_bool
  {
    \spath_finalpoint:NV
    \l__tikzspath_tmpc_tl
    #1
    \spath_translate_to:NV \l__tikzspath_joinpath_tl \l__tikzspath_tmpc_tl
  }

  \tl_clear:N \l__tikzspath_tmpa_tl
  \tl_set:Nn \l__tikzspath_tmpa_tl {spath_}

  \bool_if:NT \l__tikzspath_global_bool
  {
    \tl_put_right:Nn \l__tikzspath_tmpa_tl {g}
  }

  \tl_put_right:Nn \l__tikzspath_tmpa_tl {append}

  \bool_if:NT \l__tikzspath_weld_bool
  {
    \tl_put_right:Nn \l__tikzspath_tmpa_tl {_no_move}
  }
  \tl_put_right:Nn \l__tikzspath_tmpa_tl {:NV}

  \cs_if_exist:cF {\tl_use:N \l__tikzspath_tmpa_tl}
  {
    \tl_show:N \l__tikzspath_tmpa_tl
  }

  \use:c {\tl_use:N \l__tikzspath_tmpa_tl } #1
  \l__tikzspath_joinpath_tl
}
\cs_generate_variant:Nn \__tikzspath_join_with:Nn {cv, cn}
```

(*End of definition for* `\__tikzspath_join_with:nn`.)

_tikzspath_join_components_upright_with:Nnn

Join the specified components of the first path by splicing in the second.

```
\cs_new_protected_nopar:Npn \__tikzspath_join_components_with_aux:nnn #1#2#3
{
  \group_begin:
  \tl_set:Nn \l__tikzspath_tmpc_tl {#1}
  \tl_if_empty:nT {#3}
  {
    \spath_spot_weld_components:N \l__tikzspath_tmpc_tl
  }

  \spath_numberofcomponents:NV \l__tikzspath_tmpa_int \l__tikzspath_tmpc_tl
  \__tikzspath_seq_from_foreach:NVn \l__tikzspath_tmpb_seq \l__tikzspath_tmpa_int {#3}

  \spath_components_to_seq:NV \l__tikzspath_tmpa_seq \l__tikzspath_tmpc_tl

  \seq_pop_left:NN \l__tikzspath_tmpa_seq \l__tikzspath_tmpa_tl
  \seq_pop_left:NN \l__tikzspath_tmpb_seq \l__tikzspath_tmpb_tl

  \seq_map_indexed_inline:Nn \l__tikzspath_tmpa_seq
  {
    \int_compare:nTF
    {
      ##1 == \l__tikzspath_tmpb_tl
    }
    {
      \seq_pop_left:NNF \l__tikzspath_tmpb_seq \l__tikzspath_tmpb_tl
      {
        \tl_set:Nn \l__tikzspath_tmpb_tl {-1}
      }
      \spath_splice_between:Nnn \l__tikzspath_tmpa_tl {#2} {##2}
    }
    {
      \tl_put_right:Nn \l__tikzspath_tmpa_tl {##2}
    }
  }
  \tl_gset_eq:NN \g__tikzspath_output_tl \l__tikzspath_tmpa_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \__tikzspath_join_components_with:Nnnn #1#2#3#4
{
  \__tikzspath_join_components_with_aux:nnn {#2}{#3}{#4}
  \tl_set_eq:NN #1 \g__tikzspath_output_tl
  \tl_gclear:N \g__tikzspath_output_tl
}
\cs_generate_variant:Nn \__tikzspath_join_components_with:Nnnn {NVnn}
\cs_new_protected_nopar:Npn \__tikzspath_join_components_with:Nnn #1#2#3
{
  \__tikzspath_join_components_with:NVnn #1#1{#2}{#3}
}
\cs_generate_variant:Nn \__tikzspath_join_components_with:Nnn {cvV}
\cs_new_protected_nopar:Npn \__tikzspath_gjoin_components_with:Nnnn #1#2#3#4
{
  \__tikzspath_join_components_with_aux:nnn {#2}{#3}{#4}
  \tl_gset_eq:NN #1 \g__tikzspath_output_tl
  \tl_gclear:N \g__tikzspath_output_tl
}
\cs_generate_variant:Nn \__tikzspath_gjoin_components_with:Nnnn {NVnn}
\cs_new_protected_nopar:Npn \__tikzspath_gjoin_components_with:Nnn #1#2#3
{
  \__tikzspath_gjoin_components_with:NVnn #1#1{#2}{#3}
}
\cs_generate_variant:Nn \__tikzspath_gjoin_components_with:Nnn {cvV}
\cs_new_protected_nopar:Npn \__tikzspath_join_components_upright_with_aux:nnn #1#2#3
{
  \group_begin:
  \tl_set:Nn \l__tikzspath_tmpc_tl {#1}
  \tl_if_empty:nT {#3}
  {
    \spath_spot_weld_components:N \l__tikzspath_tmpc_tl
  }

  \spath_numberofcomponents:NV \l__tikzspath_tmpa_int \l__tikzspath_tmpc_tl
  \__tikzspath_seq_from_foreach:NVn \l__tikzspath_tmpb_seq \l__tikzspath_tmpa_int {#3}

  \spath_components_to_seq:NV \l__tikzspath_tmpa_seq \l__tikzspath_tmpc_tl

  \seq_pop_left:NN \l__tikzspath_tmpa_seq \l__tikzspath_tmpa_tl
  \seq_pop_left:NN \l__tikzspath_tmpb_seq \l__tikzspath_tmpb_tl

  \tl_set:Nn \l__tikzspath_tmpc_tl {#2}
  \spath_transform:NVnnnnnn \l__tikzspath_tmpd_tl \l__tikzspath_tmpc_tl {1}{0}{0}{-1}{0pt}{0pt}

  \seq_map_indexed_inline:Nn \l__tikzspath_tmpa_seq
  {
    \int_compare:nTF
    {
      ##1 == \l__tikzspath_tmpb_tl
    }
    {
      \seq_pop_left:NNF \l__tikzspath_tmpb_seq \l__tikzspath_tmpb_tl
      {
        \tl_set:Nn \l__tikzspath_tmpb_tl {-1}
      }

      \spath_finalpoint:NV \l__tikzspath_tmpe_tl \l__tikzspath_tmpa_tl
      \spath_initialpoint:Nn \l__tikzspath_tmpf_tl {##2}

      \dim_compare:nTF
      {
        \tl_item:Nn \l__tikzspath_tmpe_tl {1}
        >
        \tl_item:Nn \l__tikzspath_tmpf_tl {1}
      }
      {
        \spath_splice_between:NVn
        \l__tikzspath_tmpa_tl
        \l__tikzspath_tmpd_tl
        {##2}
      }
      {
        \spath_splice_between:NVn
        \l__tikzspath_tmpa_tl
        \l__tikzspath_tmpc_tl
        {##2}
      }
    }
    {
      \tl_put_right:Nn \l__tikzspath_tmpa_tl {##2}
    }
  }
  \tl_gset_eq:NN \g__tikzspath_output_tl \l__tikzspath_tmpa_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \__tikzspath_join_components_upright_with:Nnnn #1#2#3#4
{
  \__tikzspath_join_components_upright_with_aux:nnn {#2}{#3}{#4}
  \tl_set_eq:NN #1 \g__tikzspath_output_tl
  \tl_gclear:N \g__tikzspath_output_tl
}
\cs_generate_variant:Nn \__tikzspath_join_components_upright_with:Nnnn {NVnn}
\cs_new_protected_nopar:Npn \__tikzspath_join_components_upright_with:Nnn #1#2#3
{
  \__tikzspath_join_components_upright_with:NVnn #1#1{#2}{#3}
}
\cs_generate_variant:Nn \__tikzspath_join_components_upright_with:Nnn {cvV}
\cs_new_protected_nopar:Npn \__tikzspath_gjoin_components_upright_with:Nnnn #1#2#3#4
{
  \__tikzspath_join_components_upright_with_aux:nnn {#2}{#3}{#4}
  \tl_gset_eq:NN #1 \g__tikzspath_output_tl
  \tl_gclear:N \g__tikzspath_output_tl
}
\cs_generate_variant:Nn \__tikzspath_gjoin_components_upright_with:Nnnn {NVnn}
\cs_new_protected_nopar:Npn \__tikzspath_gjoin_components_upright_with:Nnn #1#2#3
{
  \__tikzspath_gjoin_components_upright_with:NVnn #1#1{#2}{#3}
}
\cs_generate_variant:Nn \__tikzspath_gjoin_components_upright_with:Nnn {cvV}
```

(*End of definition for* `\__tikzspath_join_components_with:Nnn` `\__tikzspath_join_components_upright_with:Nnn`.)

`\__tikzspath_get_components:Nn`

Get the components of the named path to the token list.

```
\cs_new_protected_nopar:Npn \__tikzspath_get_components_aux:n #1
{
  \clist_gclear_new:N \g__tikzspath_output_clist
  \spath_components_to_seq:Nn \l__tikzspath_tmpa_seq {#1}

  \seq_map_inline:Nn \l__tikzspath_tmpa_seq
  {
    \spath_anonymous:N \l__tikzspath_tmpa_tl
    \tl_new:c {\__tikzspath_path_name:V \l__tikzspath_tmpa_tl}
    \tl_set:cn {\__tikzspath_path_name:V \l__tikzspath_tmpa_tl} {##1}
    \clist_gput_right:NV \g__tikzspath_output_clist \l__tikzspath_tmpa_tl
  }
}
\cs_new_protected_nopar:Npn \__tikzspath_get_components:Nn #1#2
{
  \clist_clear_new:N #1
  \__tikzspath_get_components_aux:n {#2}
  \clist_set_eq:NN #1 \g__tikzspath_output_clist
  \clist_gclear:N \g__tikzspath_output_clist
}
\cs_generate_variant:Nn \__tikzspath_get_components:Nn {NV, Nv}
\cs_new_protected_nopar:Npn \__tikzspath_gget_components:Nn #1#2
{
  \clist_gclear_new:N #1
  \__tikzspath_get_components_aux:n {#2}
  \clist_gset_eq:NN #1 \g__tikzspath_output_clist
  \clist_gclear:N \g__tikzspath_output_clist
}
\cs_generate_variant:Nn \__tikzspath_gget_components:Nn {NV, Nv}
```

(*End of definition for* `\__tikzspath_get_components:Nn`.)

`\__tikzspath_render_components:n`

```
\cs_new_protected_nopar:Npn \__tikzspath_render_components:nn #1#2
{
  \group_begin:
  \spath_components_to_seq:Nn \l__tikzspath_tmpa_seq {#2}
  \seq_map_indexed_inline:Nn \l__tikzspath_tmpa_seq
  {
    \spath_tikz_path:nn
    {
      every~ spath~ component/.try,
      spath ~component~ ##1/.try,
      spath ~component/.try={##1},
      every~ #1~ component/.try,
      #1 ~component~ ##1/.try,
      #1 ~component/.try={##1},
    }
    {
      ##2
    }
  }
  \group_end:
}
\cs_generate_variant:Nn \__tikzspath_render_components:nn {nv}
```

(*End of definition for* `\__tikzspath_render_components:n`*.*)

__tikzspath_insert_gaps_after_components:nn

```
\cs_new_protected_nopar:Npn \__tikzspath_insert_gaps_after_components_aux:nnn #1#2#3
{
  \group_begin:
  \spath_numberofcomponents:Nn \l__tikzspath_tmpa_int {#1}
  \__tikzspath_seq_from_foreach:NVn \l__tikzspath_tmpa_seq \l__tikzspath_tmpa_int {#3}

  \tl_if_empty:nT {#3}
  {
    \seq_pop_right:NN \l__tikzspath_tmpa_seq \l__tikzspath_tmpa_tl
  }

  \seq_clear:N \l__tikzspath_tmpb_seq
  \seq_map_inline:Nn \l__tikzspath_tmpa_seq {
    \seq_put_right:Nx
    \l__tikzspath_tmpb_seq
    {\int_eval:n
      {
        \int_mod:nn { ##1 }{ \l__tikzspath_tmpa_int } + 1
      }
    }
  }

  \spath_components_to_seq:Nn \l__tikzspath_tmpc_seq {#1}

  \seq_clear:N \l__tikzspath_tmpd_seq
  \seq_map_indexed_inline:Nn \l__tikzspath_tmpc_seq
  {
    \tl_set:Nn \l__tikzspath_tmpa_tl {##2}
    \seq_if_in:NnT \l__tikzspath_tmpa_seq {##1}
    {
      \spath_shorten_at_end:Nn \l__tikzspath_tmpa_tl {(#2)/2}
    }
    \seq_if_in:NnT \l__tikzspath_tmpb_seq {##1}
    {
      \spath_shorten_at_start:Nn \l__tikzspath_tmpa_tl {(#2)/2}
    }
    \seq_put_right:NV \l__tikzspath_tmpd_seq \l__tikzspath_tmpa_tl
  }
  \tl_gset:Nx \g__tikzspath_output_tl {\seq_use:Nn \l__tikzspath_tmpd_seq {} }
  \group_end:
}
\cs_new_protected_nopar:Npn \__tikzspath_insert_gaps_after_components:Nnnn #1#2#3#4
{
  \__tikzspath_insert_gaps_after_components_aux:nnn {#2}{#3}{#4}
  \tl_set_eq:NN #1 \g__tikzspath_output_tl
  \tl_gclear:N \g__tikzspath_output_tl
}
\cs_generate_variant:Nn \__tikzspath_insert_gaps_after_components:Nnnn {NVnn}
\cs_new_protected_nopar:Npn \__tikzspath_insert_gaps_after_components:Nnn #1#2#3
{
  \__tikzspath_insert_gaps_after_components:NVnn #1#1{#2}{#3}
}
\cs_generate_variant:Nn \__tikzspath_insert_gaps_after_components:Nnn {cnn, cVV}
\cs_new_protected_nopar:Npn \__tikzspath_ginsert_gaps_after_components:Nnnn #1#2#3#4
{
  \__tikzspath_insert_gaps_after_components_aux:nnn {#2}{#3}{#4}
  \tl_gset_eq:NN #1 \g__tikzspath_output_tl
  \tl_gclear:N \g__tikzspath_output_tl
}
\cs_generate_variant:Nn \__tikzspath_ginsert_gaps_after_components:Nnnn {NVnn}
\cs_new_protected_nopar:Npn \__tikzspath_ginsert_gaps_after_components:Nnn #1#2#3
{
  \__tikzspath_ginsert_gaps_after_components:NVnn #1#1{#2}{#3}
}
\cs_generate_variant:Nn \__tikzspath_ginsert_gaps_after_components:Nnn {cnn, cVV}
```

(*End of definition for* `\__tikzspath_insert_gaps_after_components:nn`.)

`\__tikzspath_insert_gaps_after_segments:Nn`

```
\cs_new_protected_nopar:Npn \__tikzspath_insert_gaps_after_segments_aux:nnn #1#2#3
{
  \group_begin:
  \spath_reallength:Nn \l__tikzspath_tmpa_int {#1}
  \__tikzspath_seq_from_foreach:NVn \l__tikzspath_tmpa_seq \l__tikzspath_tmpa_int {#3}

  \tl_if_empty:nT {#3}
  {
    \seq_pop_right:NN \l__tikzspath_tmpb_seq \l__tikzspath_tmpa_tl
  }

  \seq_clear:N \l__tikzspath_tmpb_seq
  \seq_map_inline:Nn \l__tikzspath_tmpa_seq {
    \seq_put_right:Nx
    \l__tikzspath_tmpb_seq
    {\int_eval:n
      {
        \int_mod:nn { ##1 }{ \l__tikzspath_tmpa_int } + 1
      }
    }
  }

  \spath_segments_to_seq:Nn \l__tikzspath_tmpc_seq {#1}

  \seq_clear:N \l__tikzspath_tmpd_seq
  \seq_map_indexed_inline:Nn \l__tikzspath_tmpc_seq
  {
    \tl_set:Nn \l__tikzspath_tmpa_tl {##2}
    \seq_if_in:NnT \l__tikzspath_tmpa_seq {##1}
    {
      \spath_shorten_at_end:Nn \l__tikzspath_tmpa_tl {(#2)/2}
    }
    \seq_if_in:NnT \l__tikzspath_tmpb_seq {##1}
    {
      \spath_shorten_at_start:Nn \l__tikzspath_tmpa_tl {(#2)/2}
    }
    \seq_put_right:NV \l__tikzspath_tmpd_seq \l__tikzspath_tmpa_tl
  }
  \tl_gset:Nx \g__tikzspath_output_tl {\seq_use:Nn \l__tikzspath_tmpd_seq {} }
  \group_end:
}
\cs_new_protected_nopar:Npn \__tikzspath_insert_gaps_after_segments:Nnnn #1#2#3#4
{
  \__tikzspath_insert_gaps_after_segments_aux:nnn {#2}{#3}{#4}
  \tl_set_eq:NN #1 \g__tikzspath_output_tl
  \tl_gclear:N \g__tikzspath_output_tl
}
\cs_generate_variant:Nn \__tikzspath_insert_gaps_after_segments:Nnnn {NVnn}
\cs_new_protected_nopar:Npn \__tikzspath_insert_gaps_after_segments:Nnn #1#2#3
{
  \__tikzspath_insert_gaps_after_segments:NVnn #1#1{#2}{#3}
}
\cs_generate_variant:Nn \__tikzspath_insert_gaps_after_segments:Nnn {cnn, cVV}
\cs_new_protected_nopar:Npn \__tikzspath_ginsert_gaps_after_segments:Nnnn #1#2#3#4
{
  \__tikzspath_insert_gaps_after_segments_aux:nnn {#2}{#3}{#4}
  \tl_gset_eq:NN #1 \g__tikzspath_output_tl
  \tl_gclear:N \g__tikzspath_output_tl
}
\cs_generate_variant:Nn \__tikzspath_ginsert_gaps_after_segments:Nnnn {NVnn}
\cs_new_protected_nopar:Npn \__tikzspath_ginsert_gaps_after_segments:Nnn #1#2#3
{
  \__tikzspath_ginsert_gaps_after_segments:NVnn #1#1{#2}{#3}
}
\cs_generate_variant:Nn \__tikzspath_ginsert_gaps_after_segments:Nnn {cnn, cVV}
```

(*End of definition for* `\__tikzspath_insert_gaps_after_segments:Nn`.)

`\__tikzspath_join_components:Nn`

```
\cs_new_protected_nopar:Npn \__tikzspath_join_components_aux:nn #1#2
{
  \group_begin:

  \tl_set:Nn \l__tikzspath_tmpa_tl {#1}
  \spath_numberofcomponents:NV \l__tikzspath_tmpa_int \l__tikzspath_tmpa_tl
  \__tikzspath_seq_from_foreach:NVn \l__tikzspath_tmpa_seq \l__tikzspath_tmpa_int {#2}

  \seq_reverse:N \l__tikzspath_tmpa_seq

  \seq_map_inline:Nn \l__tikzspath_tmpa_seq
  {
    \spath_join_component:Nn \l__tikzspath_tmpa_tl {##1}
  }
  \tl_gset_eq:NN \g__tikzspath_output_tl \l__tikzspath_tmpa_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \__tikzspath_join_components:Nnn #1#2#3
{
  \__tikzspath_join_components_aux:nn {#2}{#3}
  \tl_set_eq:NN #1 \g__tikzspath_output_tl
  \tl_gclear:N \g__tikzspath_output_tl
}
\cs_generate_variant:Nn \__tikzspath_join_components:Nnn {NVn}
\cs_new_protected_nopar:Npn \__tikzspath_join_components:Nn #1#2
{
  \__tikzspath_join_components:NVn #1#1{#2}
}
\cs_generate_variant:Nn \__tikzspath_join_components:Nn {cn}
\cs_new_protected_nopar:Npn \__tikzspath_gjoin_components:Nnn #1#2#3
{
  \__tikzspath_join_components_aux:nn {#2}{#3}
  \tl_gset_eq:NN #1 \g__tikzspath_output_tl
  \tl_gclear:N \g__tikzspath_output_tl
}
\cs_generate_variant:Nn \__tikzspath_gjoin_components:Nnn {NVn}
\cs_new_protected_nopar:Npn \__tikzspath_gjoin_components:Nn #1#2
{
  \__tikzspath_gjoin_components:NVn #1#1{#2}
}
\cs_generate_variant:Nn \__tikzspath_gjoin_components:Nn {cn}
```

(*End of definition for* `\__tikzspath_join_components:Nn`*.*)

`\__tikzspath_join_components_with_bezier:Nn`

```
\cs_new_protected_nopar:Npn \__tikzspath_join_components_with_bezier_aux:nn #1#2
{
  \group_begin:
  \tl_set:Nn \l__tikzspath_tmpc_tl {#1}
  \tl_if_empty:nT {#2}
  {
    \spath_spot_weld_components:N \l__tikzspath_tmpc_tl
  }

  \spath_numberofcomponents:NV \l__tikzspath_tmpa_int \l__tikzspath_tmpc_tl
  \__tikzspath_seq_from_foreach:NVn \l__tikzspath_tmpb_seq \l__tikzspath_tmpa_int {#2}

  \spath_components_to_seq:NV \l__tikzspath_tmpa_seq \l__tikzspath_tmpc_tl

  \seq_pop_left:NN \l__tikzspath_tmpa_seq \l__tikzspath_tmpa_tl
  \seq_pop_left:NN \l__tikzspath_tmpb_seq \l__tikzspath_tmpb_tl

  \seq_map_indexed_inline:Nn \l__tikzspath_tmpa_seq
  {
    \int_compare:nTF
    {
      ##1 == \l__tikzspath_tmpb_tl
    }
    {
      \seq_pop_left:NNF \l__tikzspath_tmpb_seq \l__tikzspath_tmpb_tl
      {
        \tl_set:Nn \l__tikzspath_tmpb_tl {-1}
      }
      \spath_curve_between:Nn \l__tikzspath_tmpa_tl {##2}
    }
    {
      \tl_put_right:Nn \l__tikzspath_tmpa_tl {##2}
    }
  }
  \tl_gset_eq:NN \g__tikzspath_output_tl \l__tikzspath_tmpa_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \__tikzspath_join_components_with_bezier:Nnn #1#2#3
{
  \__tikzspath_join_components_with_bezier_aux:nn {#2}{#3}
  \tl_set_eq:NN #1 \g__tikzspath_output_tl
  \tl_gclear:N \g__tikzspath_output_tl
}
\cs_generate_variant:Nn \__tikzspath_join_components_with_bezier:Nnn {NVn}
\cs_new_protected_nopar:Npn \__tikzspath_join_components_with_bezier:Nn #1#2
{
  \__tikzspath_join_components_with_bezier:NVn #1#1{#2}
}
\cs_generate_variant:Nn \__tikzspath_join_components_with_bezier:Nn {cV}
\cs_new_protected_nopar:Npn \__tikzspath_gjoin_components_with_bezier:Nnn #1#2#3
{
  \__tikzspath_join_components_with_bezier_aux:nn {#2}{#3}
  \tl_gset_eq:NN #1 \g__tikzspath_output_tl
  \tl_gclear:N \g__tikzspath_output_tl
}
\cs_generate_variant:Nn \__tikzspath_gjoin_components_with_bezier:Nnn {NVn}
\cs_new_protected_nopar:Npn \__tikzspath_gjoin_components_with_bezier:Nn #1#2
{
  \__tikzspath_gjoin_components_with_bezier:NVn #1#1{#2}
}
\cs_generate_variant:Nn \__tikzspath_gjoin_components_with_bezier:Nn {cV}
```

(*End of definition for* `\__tikzspath_join_components_with_bezier:Nn`.)

__tikzspath_remove_components:nn

```
\cs_new_protected_nopar:Npn \__tikzspath_remove_components_aux:nn #1#2
{
  \group_begin:

  \spath_numberofcomponents:Nn \l__tikzspath_tmpa_int {#1}
  \__tikzspath_seq_from_foreach:NVn \l__tikzspath_tmpa_seq \l__tikzspath_tmpa_int {#2}

  \spath_components_to_seq:Nn \l__tikzspath_tmpb_seq {#1}

  \seq_pop_left:NNF \l__tikzspath_tmpa_seq \l__tikzspath_tmpa_tl
  {
    \tl_clear:N \l__tikzspath_tmpa_tl
  }

  \seq_clear:N \l__tikzspath_tmpc_seq
  \seq_map_indexed_inline:Nn \l__tikzspath_tmpb_seq
  {
    \tl_set:Nn \l__tikzspath_tmpb_tl {##1}
    \tl_if_eq:NNTF \l__tikzspath_tmpb_tl \l__tikzspath_tmpa_tl
    {
      \seq_pop_left:NNF \l__tikzspath_tmpa_seq \l__tikzspath_tmpa_tl
      {
        \tl_clear:N \l__tikzspath_tmpa_tl
      }
    }
    {
      \seq_put_right:Nn \l__tikzspath_tmpc_seq {##2}
    }
  }

  \tl_gset:Nx \g__tikzspath_output_tl {\seq_use:Nn \l__tikzspath_tmpc_seq {} }
  \group_end:
}
\cs_new_protected_nopar:Npn \__tikzspath_remove_components:Nnn #1#2#3
{
  \__tikzspath_remove_components_aux:nn {#2}{#3}
  \tl_set_eq:NN #1 \g__tikzspath_output_tl
  \tl_gclear:N \g__tikzspath_output_tl
}
\cs_generate_variant:Nn \__tikzspath_remove_components:Nnn {NVn}
\cs_new_protected_nopar:Npn \__tikzspath_remove_components:Nn #1#2
{
  \__tikzspath_remove_components:NVn #1#1{#2}
}
\cs_generate_variant:Nn \__tikzspath_remove_components:Nn {cn}
\cs_new_protected_nopar:Npn \__tikzspath_gremove_components:Nnn #1#2#3
{
  \__tikzspath_remove_components_aux:nn {#2}{#3}
  \tl_gset_eq:NN #1 \g__tikzspath_output_tl
  \tl_gclear:N \g__tikzspath_output_tl
}
\cs_generate_variant:Nn \__tikzspath_gremove_components:Nnn {NVn}
\cs_new_protected_nopar:Npn \__tikzspath_gremove_components:Nn #1#2
{
  \__tikzspath_gremove_components:NVn #1#1{#2}
}
\cs_generate_variant:Nn \__tikzspath_gremove_components:Nn {cn}
```

(*End of definition for* `\__tikzspath_remove_components:nn`.)

`\__tikzspath_transform_to:nn`
`\__tikzspath_transform_upright_to:nn`

```
\cs_new_protected_nopar:Npn \__tikzspath_transform_to_aux:nn #1#2
{
  \group_begin:
  \spath_reallength:Nn \l__tikzspath_tmpa_int {#2}

  \tl_set:Nx \l__tikzspath_tmpb_tl
  {\fp_to_decimal:n {(#1) * (\l__tikzspath_tmpa_int)}}
  \spath_transformation_at:NnV \l__tikzspath_tmpc_tl {#2} \l__tikzspath_tmpb_tl
  \tl_gset_eq:NN \g__tikzspath_output_tl \l__tikzspath_tmpc_tl
  \group_end:
}
\cs_new_protected_nopar:Npn \__tikzspath_transform_to:nn #1#2
{
  \__tikzspath_transform_to_aux:nn {#1}{#2}
  \exp_last_unbraced:NV \pgfsettransformentries \g__tikzspath_output_tl
  \tl_gclear:N \g__tikzspath_output_tl
}
\cs_generate_variant:Nn \__tikzspath_transform_to:nn {nv}
\cs_new_protected_nopar:Npn \__tikzspath_transform_upright_to:nn #1#2
{
  \__tikzspath_transform_to_aux:nn {#1}{#2}
  \fp_compare:nT { \tl_item:Nn \g__tikzspath_output_tl {4} < 0}
  {
    \tl_gset:Nx \g__tikzspath_output_tl
    {
      { \fp_eval:n { - (\tl_item:Nn \g__tikzspath_output_tl {1})} }
      { \fp_eval:n { - (\tl_item:Nn \g__tikzspath_output_tl {2})} }
      { \fp_eval:n { - (\tl_item:Nn \g__tikzspath_output_tl {3})} }
      { \fp_eval:n { - (\tl_item:Nn \g__tikzspath_output_tl {4})} }
      { \tl_item:Nn \g__tikzspath_output_tl {5} }
      { \tl_item:Nn \g__tikzspath_output_tl {6} }
    }
  }
  \exp_last_unbraced:NV \pgfsettransformentries \g__tikzspath_output_tl
  \tl_gclear:N \g__tikzspath_output_tl
}
\cs_generate_variant:Nn \__tikzspath_transform_upright_to:nn {nv}
```

(*End of definition for* `\__tikzspath_transform_to:nn` *and* `\__tikzspath_transform_upright_to:nn`.)

4.2 Keys

Now we define all of our keys.

```
\tikzset{
```

We're in the `spath` key family.

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

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

```
  set~ prefix/.store~ in=\l__tikzspath_prefix_tl,
  prefix/.is~choice,
  prefix/default/.style={
    /tikz/spath/set~ prefix=tikz@intersect@path@name@
  },
  set~ suffix/.store~ in=\l__tikzspath_suffix_tl,
  suffix/.is~choice,
  suffix/default/.style={
    /tikz/spath/set~ suffix={}
  },
  set~ name/.style={
    /tikz/spath/prefix=#1,
    /tikz/spath/suffix=#1
  },
```

Hook in to the end of the path construction

```
  at~ end~ path~ construction/.code={
    \tl_put_right:Nn \l__tikzspath_tikzpath_finish_tl {#1}
  },
```

Keys for saving and cloning a soft path.

```
  save/.code={
    \tikz@addmode{
      \spath_get_current_path:N \l__tikzspath_tmpa_tl
      \spath_bake_round:NV \l__tikzspath_tmpa_tl \l__tikzspath_tmpa_tl
      \spath_bake_shorten:NV \l__tikzspath_tmpa_tl \l__tikzspath_tmpa_tl
      \spath_save_path:cV {\__tikzspath_path_name:n {#1}}
      \l__tikzspath_tmpa_tl
    }
  },
  save~ global/.code={
    \tikz@addmode{
      \spath_get_current_path:N \l__tikzspath_tmpa_tl
      \spath_bake_round:NV \l__tikzspath_tmpa_tl \l__tikzspath_tmpa_tl
      \spath_bake_shorten:NV \l__tikzspath_tmpa_tl \l__tikzspath_tmpa_tl
      \spath_gsave_path:cV {\__tikzspath_path_name:n {#1}}
      \l__tikzspath_tmpa_tl
    }
  },
  clone/.code~ 2~ args={
    \__tikzspath_maybe_current_path:nn
    {
      \__tikzspath_check_path:nnn {
        \tl_clear_new:c {\__tikzspath_path_name:n {#1}}
        \tl_set_eq:cc {\__tikzspath_path_name:n {#1}}
      }
    }
    {#2}{}
  },
  clone~ global/.code~ 2~ args={
    \__tikzspath_maybe_current_path:nn
    {
      \__tikzspath_check_path:nnn {
        \tl_gclear_new:c {\__tikzspath_path_name:n {#1}}
        \tl_gset_eq:cc {\__tikzspath_path_name:n {#1}}
      }
    }
    {#2}{}
  },
```

Saves a soft path to the aux file.

```
  save~ to~ aux/.code={
    \__tikzspath_maybe_current_path:nn
    {
      \__tikzspath_check_path:nnn {
        \spath_save_to_aux:c
      }
    }
    {#1}
    {}
  },
```

Exports the path as an SVG file.

```
  export~ to~ svg/.code={
    \__tikzspath_maybe_current_path:nn
    {
      \__tikzspath_check_path:nnn {
        \spath_export_to_svg:nv {#1}
      }
    }
    {#1}
    {}
  },
```

Inserts the named path at the current point in the path, with options for how this is accomplished. The inserted path can be transformed, reversed, moved to the current point, and welded to the current path. If this is used before the path has been started then it becomes the start of the path (and the "current point" is taken as the origin).

```
  use/.code={
    \bool_set_false:N \l__tikzspath_reverse_bool
    \bool_set_false:N \l__tikzspath_weld_bool
    \bool_set_false:N \l__tikzspath_move_bool
    \bool_set_false:N \l__tikzspath_current_transformation_bool
    \bool_set_true:N \l_spath_movetorelevant_bool
    \tl_clear:N \l__tikzspath_joinpath_tl
    \tl_clear:N \l__tikzspath_transformation_tl
    \tikzset{
      spath/join/.cd,
      #1
    }

    \__tikzspath_check_path:nVn
    {
      \__tikzspath_use_path:v
    } \l__tikzspath_joinpath_tl {}

  },
```

Some aliases for the above.

```
  restore/.style={/tikz/spath/use={#1}},
  restore~ reverse/.style={/tikz/spath/use={reverse, #1}},
  append/.style={/tikz/spath/use={move, weld, #1}},
  append~ no~ move/.style={/tikz/spath/use={weld, #1}},
  append~ reverse/.style={/tikz/spath/use={move, weld, reverse, #1}},
  append~ reverse~ no~ move/.style={/tikz/spath/use={weld, reverse, #1}},
  insert/.style={/tikz/spath/use={#1}},
  insert~ reverse/.style={/tikz/spath/use={reverse, #1}},
```

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

```
  show~current~path/.code={
    \tikz@addmode{
      \pgfsyssoftpath@getcurrentpath\l__tikzspath_tmpa_tl
      \iow_term:n {---~ current~ soft~ path~ ---}
      \spath_show:V \l__tikzspath_tmpa_tl
    }
  },
```

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

```
  show/.code={
    \__tikzspath_check_path:nnn {
      \iow_term:n {---~ soft~ path~ #1~ ---}
      \spath_show:v
    } {#1} {}
  },
```

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

```
  join~ with/.code~ 2~ args={
    \bool_set_false:N \l__tikzspath_reverse_bool
    \bool_set_false:N \l__tikzspath_weld_bool
    \bool_set_false:N \l__tikzspath_move_bool
    \bool_set_false:N \l__tikzspath_global_bool
    \bool_set_false:N \l__tikzspath_current_transformation_bool
    \tl_clear:N \l__tikzspath_joinpath_tl
    \tl_clear:N \l__tikzspath_transformation_tl
    \tikzset{
      spath/join/.cd,
      #2
    }

    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_two_paths:nnVn
      {
        \__tikzspath_join_with:cv
      }
    } {#1} { \l__tikzspath_joinpath_tl {} }
  },
```

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

```
  spot~ weld/.code={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_spot_weld_components:c
      }
    } {#1} { {} }
  },
  spot~ weld~ globally/.code={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_spot_gweld_components:c
      }
    } {#1} { {} }
  },
```

Reverses the named path.

```
  reverse/.code={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_reverse:c
      }
    } {#1} { {} }
  },
  reverse~ globally/.code={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_greverse:c
      }
    } {#1} { {} }
  },
```

Adjust a path to span between two points.

```
  span/.code ~n~ args={3}{
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \__tikzspath_process_tikz_point:Nn \l__tikzspath_tmpa_tl {#2}
        \__tikzspath_process_tikz_point:Nn \l__tikzspath_tmpb_tl {#3}
        \spath_span:cVV
      }
    } {#1} { {} \l__tikzspath_tmpa_tl \l__tikzspath_tmpb_tl }
  },
  span~ global/.code ~n~ args={3}{
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \__tikzspath_process_tikz_point:Nn \l__tikzspath_tmpa_tl {#2}
        \__tikzspath_process_tikz_point:Nn \l__tikzspath_tmpb_tl {#3}
        \spath_span:cVV
      }
    } {#1} { {} \l__tikzspath_tmpa_tl \l__tikzspath_tmpb_tl }
  },
```

Defines a to path

```
to/.style={
  to~path={
    [
      spath/span={#1}{(\tikztostart)}{(\tikztotarget)},
      spath/use={#1,weld},
    ]
    \tikztonodes
  }
},
```

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

```
splice/.code ~n~ args={3}{
  \__tikzspath_maybe_current_path_reuse:nnn
  {
    \__tikzspath_check_three_paths:nnnnn
    {
      \spath_splice_between:cvv
    }
  } {#1} { {#2} {#3} {} }
},
splice~ global/.code ~n~ args={3}{
  \__tikzspath_maybe_current_path_reuse:nnn
  {
    \__tikzspath_check_three_paths:nnnnn
    {
      \spath_gsplice_between:cvv
    }
  } {#1} { {#2} {#3} {} }
},
```

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

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

```
join~ components~ with/.code~2~args={
  \tl_if_head_is_group:nTF {#2}
  {
    \tl_set:Nx \l__tikzspath_tmpc_tl { \tl_item:nn {#2} {1} }
    \tl_set:Nx \l__tikzspath_tmpd_tl { \tl_item:nn {#2} {2} }
  }
  {
    \tl_set:Nn \l__tikzspath_tmpc_tl {#2}
    \tl_clear:N \l__tikzspath_tmpd_tl
  }

  \__tikzspath_maybe_current_path_reuse:nnn
  {
    \__tikzspath_check_two_paths:nnVn
    {
      \__tikzspath_join_components_with:cvV
    }
  } {#1} { \l__tikzspath_tmpc_tl {} \l__tikzspath_tmpd_tl  }
},
join~ components~ globally~ with/.code~2~args={
  \tl_if_head_is_group:nTF {#2}
  {
    \tl_set:Nx \l__tikzspath_tmpc_tl { \tl_item:nn {#2} {1} }
    \tl_set:Nx \l__tikzspath_tmpd_tl { \tl_item:nn {#2} {2} }
  }
  {
    \tl_set:Nn \l__tikzspath_tmpc_tl {#2}
    \tl_clear:N \l__tikzspath_tmpd_tl
  }

  \__tikzspath_maybe_current_path_reuse:nnn
  {
    \__tikzspath_check_two_paths:nnVn
    {
      \__tikzspath_gjoin_components_with:cvV
    }
  } {#1} { \l__tikzspath_tmpc_tl {} \l__tikzspath_tmpd_tl  }
},
join~ components~ upright~ with/.code~2~args={
  \tl_if_head_is_group:nTF {#2}
  {
    \tl_set:Nx \l__tikzspath_tmpc_tl { \tl_item:nn {#2} {1} }
    \tl_set:Nx \l__tikzspath_tmpd_tl { \tl_item:nn {#2} {2} }
  }
  {
    \tl_set:Nn \l__tikzspath_tmpc_tl {#2}
    \tl_clear:N \l__tikzspath_tmpd_tl
  }

  \__tikzspath_maybe_current_path_reuse:nnn
  {
    \__tikzspath_check_two_paths:nnVn
    {
      \__tikzspath_join_components_upright_with:cvV
    }
  } {#1} { \l__tikzspath_tmpc_tl {} \l__tikzspath_tmpd_tl  }
},
join~ components~ globally~ upright~ with/.code~2~args={
  \tl_if_head_is_group:nTF {#2}
  {
    \tl_set:Nx \l__tikzspath_tmpc_tl { \tl_item:nn {#2} {1} }
    \tl_set:Nx \l__tikzspath_tmpd_tl { \tl_item:nn {#2} {2} }
  }
  {
    \tl_set:Nn \l__tikzspath_tmpc_tl {#2}
    \tl_clear:N \l__tikzspath_tmpd_tl
  }

  \__tikzspath_maybe_current_path_reuse:nnn
  {
    \__tikzspath_check_two_paths:nnVn
```

```
      {
        \__tikzspath_gjoin_components_upright_with:cvV
      }
    } {#1} { \l__tikzspath_tmpc_tl {} \l__tikzspath_tmpd_tl  }
  },
  join~ components~ with~ bezier/.code={
    \tl_if_head_is_group:nTF {#1}
    {
      \tl_set:Nx \l__tikzspath_tmpc_tl { \tl_item:nn {#1} {1} }
      \tl_set:Nx \l__tikzspath_tmpd_tl { \tl_item:nn {#1} {2} }
    }
    {
      \tl_set:Nn \l__tikzspath_tmpc_tl {#1}
      \tl_clear:N \l__tikzspath_tmpd_tl
    }

    \__tikzspath_maybe_current_path_reuse:nVn
    {
      \__tikzspath_check_path:nnn
      {
        \__tikzspath_join_components_with_bezier:cV
      }
    } \l__tikzspath_tmpc_tl { {} \l__tikzspath_tmpd_tl }
  },
  join~ components~ globally~ with~ bezier/.code~2~args={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \__tikzspath_gjoin_components_with_bezier:cn
      }
    } {#1} { {} {#2} }
  },
```

Close a path.

```
  close/.code={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_close:c
      }
    } {#1} { {} }
  },
  close~ globally/.code={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_gclose:c
      }
    } {#1} { {} }
  },
```

Open a path.

```
  open/.code={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_open:c
      }
    } {#1} { {} }
  },
  open~ globally/.code={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_gopen:c
      }
    } {#1} { {} }
  },
```

Close a path, ensuring that the end point is exactly where it will close up to.

```
  adjust~ and~ close/.code={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_adjust_close:c
      }
    } {#1} { {} }
  },
  adjust~ and~ close~ globally/.code={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_adjust_gclose:c
      }
    } {#1} { {} }
  },
```

Close a path with another path.

```
  close~ with/.code~ 2~ args={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_two_paths:nnnn
      {
        \spath_close_with:cv
      }
    } {#1} { {#2} {} }
  },
  close~ globally~ with/.code~ 2~ args={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_two_paths:nnnn
      {
        \spath_gclose_with:cv
      }
    } {#1} { {#2} {} }
  },
```

Close a path with a curve.

```
  close~ with~ curve/.code={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_close_with_curve:c
      }
    } {#1} { {} }
  },
  close~ globally~ with~ curve/.code={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_gclose_with_curve:c
      }
    } {#1} { {} }
  },
```

These keys shorten the path by a dimension.

```
  shorten~ at~ end/.code~ 2~ args={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_shorten_at_end:cn
      }
    } {#1} { {} {#2} }
  },
  shorten~ at~ start/.code~ 2~ args ={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_shorten_at_start:cn
      }
    } {#1} { {} {#2} }
  },
  shorten~ at~ both~ ends/.code~ 2~ args={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_shorten_at_both_ends:cn
      }
    } {#1} { {} {#2} }
  },
  shorten~ globally~ at~ end/.code~ 2~ args={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_gshorten_at_end:cn
      }
    } {#1} { {} {#2} }
  },
  shorten~ globally~ at~ start/.code~ 2~ args ={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_gshorten_at_start:cn
      }
    } {#1} { {} {#2} }
  },
  shorten~ globally~ at~ both~ ends/.code~ 2~ args={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_gshorten_at_both_ends:cn
      }
    } {#1} { {} {#2} }
  },
```

These keys split a path at a parameter, the `keep` versions only keep one part of the resultant path.

```
  split~ at/.code~ 2~ args={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_split_at_normalised:cn
      }
    } {#1} { {} {#2} }
  },
  split~ globally~ at/.code~ 2~ args={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_gsplit_at_normalised:cn
      }
    } {#1} { {} {#2} }
  },
  split~ at~ into/.code~ n~ args={4}{
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_split_at_normalised:ccvn {\__tikzspath_path_name:n {#1}}
        {\__tikzspath_path_name:n {#2}}
      }
    } {#3} { {} {#4} }
  },
  split~ globally~ at~ into/.code~ n~ args={4}{
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_gsplit_at_normalised:ccvn {\__tikzspath_path_name:n {#1}}
        {\__tikzspath_path_name:n {#2}}
      }
    } {#3} { {} {#4} }
  },
  split~ at~ keep~ start/.code~ 2~ args={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_split_at_normalised_keep_start:cn
      }
    } {#1} { {} {#2} }
  },
  split~ globally~ at~ keep~ start/.code~ 2~ args={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_gsplit_at_normalised_keep_start:cn
      }
    } {#1} { {} {#2} }
  },
  split~ at~ keep~ end/.code~ 2~ args={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_split_at_normalised_keep_end:cn
      }
    } {#1} { {} {#2} }
  },
  split~ globally~ at~ keep~ end/.code~ 2~ args={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_gsplit_at_normalised_keep_end:cn
      }
    } {#1} { {} {#2} }
  },
  split~ at~ keep~ middle/.style~ n~ args={3}{
    /tikz/spath/split~ at~ keep~ start={#1}{#3},
    /tikz/spath/split~ at~ keep~ end={#1}{(#2)/(#3)},
  },
  split~ globally~ at~ keep~ middle/.style~ n~ args={3}{
    /tikz/spath/split~ globally~ at~ keep~ start={#1}{#3},
    /tikz/spath/split~ globally~ at~ keep~ end={#1}{(#2)/(#3)},
  },
```

This translates the named path.

```
  translate/.code~ n~ args={3}{
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_translate:cnn
      }
    } {#1} { {} {#2}{#3} }
  },
  translate~ globally/.code~ n~ args={3}{
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_gtranslate:cnn
      }
    } {#1} { {} {#2}{#3} }
  },
```

This normalises the named path.

```
  normalise/.code={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_normalise:c
      }
    } {#1} { {} }
  },
  normalise~ globally/.code={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_gnormalise:c
      }
    } {#1} { {} }
  },
```

Transforms the named path using TikZ transformation specifications.

```
  transform/.code~ 2~ args={
    \group_begin:
    \pgftransformreset
    \tikzset{#2}
    \pgfgettransform \l__tikzspath_tmpa_tl
    \tl_gset_eq:NN \g__tikzspath_smuggle_tl \l__tikzspath_tmpa_tl
    \group_end:

    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_transform:cV
      }
    } {#1} { {} \g__tikzspath_smuggle_tl }
  },
  transform~globally/.code~ 2~ args={
    \group_begin:
    \pgftransformreset
    \tikzset{#2}
    \pgfgettransform \l__tikzspath_tmpa_tl
    \tl_gset_eq:NN \g__tikzspath_smuggle_tl \l__tikzspath_tmpa_tl
    \group_end:

    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_gtransform:cV
      }
    } {#1} { {} \g__tikzspath_smuggle_tl }
  },
```

Splits first path where it intersects with the second.

```
  split~ at~ intersections~ with/.code~ 2~ args={
    \tl_if_exist:cTF
    {
      tikz@library@intersections@loaded
    }
    {
      \__tikzspath_maybe_current_two_paths_reuse_first:nnnn
      {
        \__tikzspath_check_two_paths:nnnn
        {
          \spath_split_path_at_intersections:cv
        }
      } {#1} {#2} { {} }
    }
    {
      \msg_warning:nn { spath3 } { load intersections }
    }
  },
  split~ globally~ at~ intersections~ with/.code~ n~ args={2}{
    \tl_if_exist:cTF
    {
      tikz@library@intersections@loaded
    }
    {
      \__tikzspath_maybe_current_two_paths_reuse_first:nnnn
      {
        \__tikzspath_check_two_paths:nnnn
        {
          \spath_gsplit_path_at_intersections:cv
        }
      } {#1} {#2} { {} }
    }
    {
      \msg_warning:nn { spath3 } { load intersections }
    }
  },
```

Splits two paths at their mutual intersections.

```
split~ at~ intersections/.code~ n~ args={2}{
  \tl_if_exist:cTF
  {
    tikz@library@intersections@loaded
  }
  {
    \__tikzspath_maybe_current_two_paths_reuse_both:nnnn
    {
      \__tikzspath_check_two_paths:nnnn
      {
        \spath_split_at_intersections:cc
      }
    } {#1} {#2} { {} }
  }
  {
    \msg_warning:nn { spath3 } { load intersections }
  }
},
split~ globally~ at~ intersections/.code~ n~ args={2}{
  \tl_if_exist:cTF
  {
    tikz@library@intersections@loaded
  }
  {
    \__tikzspath_maybe_current_two_paths_reuse_both:nnnn
    {
      \__tikzspath_check_two_paths:nnnn
      {
        \spath_gsplit_at_intersections:cc
      }
    } {#1} {#2} { {} }
  }
  {
    \msg_warning:nn { spath3 } { load intersections }
  }
},
```

Splits a path at its self-intersections.

```
split~ at~ self~ intersections/.code={
  \tl_if_exist:cTF
  {
    tikz@library@intersections@loaded
  }
  {
    \__tikzspath_maybe_current_path_reuse:nnn
```

```
      {
        \__tikzspath_check_path:nnn
        {
          \spath_split_at_self_intersections:c
        }
      } {#1} { {} }
    }
    {
      \msg_warning:nn { spath3 } { load intersections }
    }
  },
  split~ globally~ at~ self~ intersections/.code={
    \tl_if_exist:cTF
    {
      tikz@library@intersections@loaded
    }
    {
      \__tikzspath_maybe_current_path_reuse:nnn
      {
        \__tikzspath_check_path:nnn
        {
          \spath_gsplit_at_self_intersections:c
        }
      } {#1} { {} }
    }
    {
      \msg_warning:nn { spath3 } { load intersections }
    }
  },
```

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

```
  get~ components~ of/.code~ 2~ args={
    \__tikzspath_maybe_current_path:nn
    {
      \__tikzspath_check_path:nnn {
        \__tikzspath_get_components:Nv #2
      }
    }
    {#1}
    {}
  },
  get~ components~ of~ globally/.code~ 2~ args={
    \__tikzspath_maybe_current_path:nn
    {
      \__tikzspath_check_path:nnn {
        \__tikzspath_gget_components:Nv #2
      }
    }
    {#1}
    {}
  },
```

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

```
render~ components/.code={
  \__tikzspath_maybe_current_path:nn
  {
    \__tikzspath_check_path:nnn {
      \__tikzspath_render_components:nv {#1}
    }
  }
  {#1}
  {}
},
```

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

```
insert~ gaps~ after~ components/.code~ 2~ args={
  \tl_if_head_is_group:nTF {#2}
  {
    \tl_set:Nx \l__tikzspath_tmpc_tl { \tl_item:nn {#2} {1} }
    \tl_set:Nx \l__tikzspath_tmpd_tl { \tl_item:nn {#2} {2} }
  }
  {
    \tl_set:Nn \l__tikzspath_tmpc_tl {#2}
    \tl_clear:N \l__tikzspath_tmpd_tl
  }

  \__tikzspath_maybe_current_path_reuse:nnn
  {
    \__tikzspath_check_path:nnn
    {
      \__tikzspath_insert_gaps_after_components:cVV
    }
  } {#1} { {} \l__tikzspath_tmpc_tl \l__tikzspath_tmpd_tl  }
},
insert~ gaps~ globally~ after~ components/.code~ 2~ args={
  \tl_if_head_is_group:nTF {#2}
  {
    \tl_set:Nx \l__tikzspath_tmpc_tl { \tl_item:nn {#2} {1} }
    \tl_set:Nx \l__tikzspath_tmpd_tl { \tl_item:nn {#2} {2} }
  }
  {
    \tl_set:Nn \l__tikzspath_tmpc_tl {#2}
    \tl_clear:N \l__tikzspath_tmpd_tl
  }

  \__tikzspath_maybe_current_path_reuse:nnn
  {
    \__tikzspath_check_path:nnn
    {
      \__tikzspath_ginsert_gaps_after_components:cVV
    }
  } {#1} { {} \l__tikzspath_tmpc_tl \l__tikzspath_tmpd_tl  }
},
```

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

```
insert~ gaps~ after~ segments/.code~ 2~ args={
  \tl_if_head_is_group:nTF {#2}
  {
    \tl_set:Nx \l__tikzspath_tmpc_tl { \tl_item:nn {#2} {1} }
    \tl_set:Nx \l__tikzspath_tmpd_tl { \tl_item:nn {#2} {2} }
  }
  {
    \tl_set:Nn \l__tikzspath_tmpc_tl {#2}
    \tl_clear:N \l__tikzspath_tmpd_tl
  }

  \__tikzspath_maybe_current_path_reuse:nnn
  {
    \__tikzspath_check_path:nnn
    {
      \__tikzspath_insert_gaps_after_segments:cVV
    }
  } {#1} { {} \l__tikzspath_tmpc_tl \l__tikzspath_tmpd_tl  }
},
insert~ gaps~ globally~ after~ segments/.code~ 2~ args={
  \tl_if_head_is_group:nTF {#2}
  {
    \tl_set:Nx \l__tikzspath_tmpc_tl { \tl_item:nn {#2} {1} }
    \tl_set:Nx \l__tikzspath_tmpd_tl { \tl_item:nn {#2} {2} }
  }
  {
    \tl_set:Nn \l__tikzspath_tmpc_tl {#2}
    \tl_clear:N \l__tikzspath_tmpd_tl
  }

  \__tikzspath_maybe_current_path_reuse:nnn
  {
    \__tikzspath_check_path:nnn
    {
      \__tikzspath_ginsert_gaps_after_segments:cVV
    }
  } {#1} { {} \l__tikzspath_tmpc_tl \l__tikzspath_tmpd_tl  }
},
```

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

```
join~ components/.code~ 2~ args={
  \__tikzspath_maybe_current_path_reuse:nnn
  {
    \__tikzspath_check_path:nnn
    {
      \__tikzspath_join_components:cn
    }
  } {#1} { {} {#2} }
},
join~ components~ globally/.code~ 2~ args={
  \__tikzspath_maybe_current_path_reuse:nnn
  {
    \__tikzspath_check_path:nnn
    {
      \__tikzspath_gjoin_components:cn
    }
  } {#1} { {} {#2} }
},
```

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

```
  remove~ empty~ components/.code={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_remove_empty_components:c
      }
    } {#1} { {} }
  },
  remove~ empty~ components~ globally/.code={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_gremove_empty_components:c
      }
    } {#1} { {} }
  },
```

Replace all line segments by Bézier curves.

```
  replace~ lines/.code={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_replace_lines:c
      }
    } {#1} { {} }
  },
  replace~ lines~ globally/.code={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \spath_greplace_lines:c
      }
    } {#1} { {} }
  },
```

Remove the specified components.

```
  remove~ components/.code~ 2~ args={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \__tikzspath_remove_components:cn
      }
    } {#1} { {} {#2} }
  },
  remove~ components~ globally/.code~ 2~ args={
    \__tikzspath_maybe_current_path_reuse:nnn
    {
      \__tikzspath_check_path:nnn
      {
        \__tikzspath_gremove_components:cn
      }
    } {#1} { {} {#2} }
  },
```

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

```
  draft~ mode/.code={
    \__tikzspath_set_bool:Nn \l__tikzspath_draft_bool {#1}
  },
  maybe~ spot~ weld/.code={
    \bool_if:NF \l__tikzspath_draft_bool
    {
      \__tikzspath_maybe_current_path_reuse:nnn
      {
        \__tikzspath_check_path:nnn
        {
          \spath_spot_weld_components:c
        }
      } {#1} { {} }
    }
  },
  maybe~ spot~ weld~ globally/.code={
    \bool_if:NF \l__tikzspath_draft_bool
    {
      \__tikzspath_maybe_current_path_reuse:nnn
      {
        \__tikzspath_check_path:nnn
        {
          \spath_spot_gweld_components:c
        }
      } {#1} { {} }
    }
  },
```

Set the transformation to lie along a path.

```
  transform~ to/.code~ 2~ args={
    \__tikzspath_maybe_current_path:nn
    {
      \__tikzspath_check_path:nnn {
        \__tikzspath_transform_to:nv {#2}
      }
    }
    {#1}
    {
```

```
    \pgfsettransformentries {1}{0}{0}{1}{0pt}{0pt}
  }
},
```

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

```
upright~ transform~ to/.code~ 2~ args={
  \__tikzspath_maybe_current_path:nn
  {
    \__tikzspath_check_path:nnn {
      \__tikzspath_transform_upright_to:nv {#2}
    }
  }
  {#1}
  {
    \pgfsettransformentries {1}{0}{0}{1}{0pt}{0pt}
  }
},
```

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

```
knot/.style~ n~ args={3}{
  /tikz/spath/split~ at~ self~ intersections=#1,
  /tikz/spath/remove~ empty~ components=#1,
  /tikz/spath/insert~ gaps~ after~ components={#1}{#2}{#3},
  /tikz/spath/maybe~ spot~ weld=#1,
  /tikz/spath/render~ components=#1
},
global~ knot/.style~ n~ args={3}{
  /tikz/spath/split~ globally~ at~ self~ intersections=#1,
  /tikz/spath/remove~ empty~ components~ globally=#1,
  /tikz/spath/insert~ gaps~ globally ~after~ components={#1}{#2}{#3},
  /tikz/spath/maybe~ spot~ weld~ globally=#1,
  /tikz/spath/render~ components=#1
},
```

```
arrow~ shortening/.code={
  \__tikzspath_set_bool:Nn \l_spath_arrow_shortening_bool {#1}
},
```

For single argument commands which take a path as their argument, set the default to be `current` so that they use the current path.

```
show/.default=current,
spot~ weld/.default=current,
spot~ weld~ globally/.default=current,
reverse/.default=current,
reverse~ globally/.default=current,
close/.default=current,
close~ globally/.default=current,
open/.default=current,
open~ globally/.default=current,
adjust~ and~ close/.default=current,
adjust~ and~ close~ globally/.default=current,
close~ with~ curve/.default=current,
close~ globally~ with~ curve/.default=current,
remove~ empty~ components/.default=current,
  remove~ empty~ components~ globally/.default=current,
  replace~ lines/.default=current,
  replace~ lines~ globally/.default=current,
  maybe~ spot~ weld/.default=current,
  maybe~ spot~ weld~ globally/.default=current,
}
```

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

```
\cs_new_protected_nopar:Npn \__tikzspath_get_point_at:nn #1#2
{
  \group_begin:
  \spath_reallength:Nn \l__tikzspath_tmpa_int {#2}
  \tl_set:Nx \l__tikzspath_tmpb_tl
  {\fp_to_decimal:n {(#1) * (\l__tikzspath_tmpa_int)}}
  \spath_point_at:NnV \l__tikzspath_tmpc_tl {#2} \l__tikzspath_tmpb_tl

  \tl_clear:N \l__tikzspath_tmpd_tl
  \tl_put_right:Nn \l__tikzspath_tmpd_tl {\pgf@x=}
  \tl_put_right:Nx \l__tikzspath_tmpd_tl {\tl_item:Nn \l__tikzspath_tmpc_tl {1}}
  \tl_put_right:Nn \l__tikzspath_tmpd_tl {\relax}
  \tl_put_right:Nn \l__tikzspath_tmpd_tl {\pgf@y=}
  \tl_put_right:Nx \l__tikzspath_tmpd_tl {\tl_item:Nn \l__tikzspath_tmpc_tl {2}}
  \tl_put_right:Nn \l__tikzspath_tmpd_tl {\relax}

  \tl_gset_eq:NN \g__tikzspath_output_tl \l__tikzspath_tmpd_tl
  \group_end:
 }
\cs_generate_variant:Nn \__tikzspath_get_point_at:nn {VV, Vn, Vv}

\tikzdeclarecoordinatesystem{spath}{%
  \group_begin:
  \tl_set:Nn \l__tikzspath_tmpa_tl {#1}
  \tl_trim_spaces:N \l__tikzspath_tmpa_tl

  \seq_set_split:NnV \l__tikzspath_tmpa_seq {~} \l__tikzspath_tmpa_tl
  \seq_pop_right:NN \l__tikzspath_tmpa_seq \l__tikzspath_tmpb_tl

  \tl_set:Nx \l__tikzspath_tmpa_tl { \seq_use:Nn \l__tikzspath_tmpa_seq {~} }

  \__tikzspath_maybe_current_path:nV
  {
    \__tikzspath_check_path:nnn {
      \__tikzspath_get_point_at:Vv \l__tikzspath_tmpb_tl
    }
  }
  \l__tikzspath_tmpa_tl
  {
    \tl_gset_eq:NN \g__tikzspath_output_tl \pgfpointorigin
  }

  \group_end:
  \use:c {pgf@process}{%
    \tl_use:N \g__tikzspath_output_tl
    \pgftransforminvert
    \use:c {pgf@pos@transform@glob}
  }
  \tl_gclear:N \g__tikzspath_output_tl
}

\ExplSyntaxOff
```

5 The Calligraphy Package

```
⟨@@=cal⟩
```

5.1 Initialisation

```
\RequirePackage{spath3}
\ExplSyntaxOn

\tl_new:N \l__cal_tmpa_tl
\tl_new:N \l__cal_tmpb_tl
\tl_new:N \l__cal_tmp_path_tl
\tl_new:N \l__cal_tmp_rpath_tl
\tl_new:N \l__cal_tmp_rpathb_tl
\tl_new:N \l__cal_tmp_patha_tl

\seq_new:N \l__cal_tmpa_seq

\int_new:N \l__cal_tmpa_int
\int_new:N \l__cal_tmpb_int
\int_new:N \g__cal_path_component_int
\int_new:N \g__cal_label_int

\fp_new:N \l__cal_tmpa_fp
\fp_new:N \l__cal_tmpb_fp
\fp_new:N \l__cal_tmpc_fp
\fp_new:N \l__cal_tmpd_fp
\fp_new:N \l__cal_tmpe_fp

\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_annotate_bool
\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

\cs_generate_variant:Nn \tl_put_right:Nn {Nv}

\msg_new:nnn { calligraphy } { undefined pen } { The~ pen~ "#1"~ is~ not~ defined. }
```

5.2 TikZ Keys

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

```
\tikzset{
  define~pen/.code={
    \tikzset{pen~name=#1}
    \pgf@relevantforpicturesizefalse
    \tikz@addmode{
      \pgfsyssoftpath@getcurrentpath\l__cal_tmpa_tl
      \spath_components_to_seq:NV \l__cal_tmpa_seq \l__cal_tmpa_tl
      \seq_gclear_new:c {g__cal_pen_\pgfkeysvalueof{/tikz/pen~name}_seq}
      \seq_gset_eq:cN
      {g__cal_pen_\pgfkeysvalueof{/tikz/pen~name}_seq} \l__cal_tmpa_seq
      \pgfusepath{discard}%
    }
  },
  define~pen/.default={default},
  use~pen/.code={
    \tikzset{pen~name=#1}
    \int_gzero:N \g__cal_path_component_int
    \cs_set_eq:NN \pgfpathmoveto \cal_moveto:n
    \tikz@addmode{
      \pgfsyssoftpath@getcurrentpath\l__cal_tmpa_tl
      \spath_components_to_seq:NV \l__cal_tmpa_seq \l__cal_tmpa_tl
      \tl_if_exist:cTF {g__cal_pen_\pgfkeysvalueof{/tikz/pen~name}_seq}
      {
        \cal_path_create:Nc \l__cal_tmpa_seq
        {g__cal_pen_\pgfkeysvalueof{/tikz/pen~name}_seq}
      }
      {
        \msg_warning:nnx { calligraphy } { undefined pen }
        { \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 \g__cal_path_component_int}
    \tl_set:cn
    {l__cal_stroke_inline_style_ \int_use:N \g__cal_path_component_int} {#1}
  },
  annotate/.style={
    annotate~if,
    annotate~reset,
    annotation~style/.update~value={#1},
  },
  annotate~if/.default={true},
  annotate~if/.code={
    \tl_if_eq:nnTF {#1} {true}
    {
      \bool_set_true:N \l__cal_annotate_bool
    }
    {
      \bool_set_false:N \l__cal_annotate_bool
    }
  },
  annotate~reset/.code={
    \int_gzero:N \g__cal_label_int
  },
  annotation~style/.initial={draw,->},
  annotation~shift/.initial={(0,1ex)},
  every~annotation~node/.initial={anchor=south~west},
  annotation~node~style/.code~2~args={
    \tl_clear_new:c {l__cal_annotation_style_ #1 _tl}
    \tl_set:cn {l__cal_annotation_style_ #1 _tl}{#2}
  },
  tl~use:N/.code={
    \exp_args:NV \pgfkeysalso #1
  },
  tl~use:c/.code={
    \tl_if_exist:cT {#1}
    {
      \exp_args:Nv \pgfkeysalso {#1}
    }
  },
  /handlers/.update~style/.code={
    \tl_if_eq:nnF {#1} {\pgfkeysnovalue}
    {
      \pgfkeys{\pgfkeyscurrentpath/.code=\pgfkeysalso{#1}}
    }
  },
  /handlers/.update~value/.code={
    \tl_if_eq:nnF {#1} {\pgfkeysnovalue}
    {
      \pgfkeyssetvalue{\pgfkeyscurrentpath}{#1}
    }
  },
}
```

Some wrappers around the TikZ keys.

```
\NewDocumentCommand \pen { O{} }
{
  \path[define~ pen,every~ calligraphy~ pen/.try,#1]
}

\NewDocumentCommand \definepen { O{} }
{
  \tikz \path[define~ pen,every~ calligraphy~ pen/.try,#1]
}

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

5.3 The Path Creation

`\cal_path_create:NN`

This is the main command for creating the calligraphic paths. First argument is the given path Second argument is the pen path

```
\cs_new_protected_nopar:Npn \cal_path_create:NN #1#2
{
  \int_zero:N \l__cal_tmpa_int
  \seq_map_inline:Nn #1
  {
    \int_compare:nT {\tl_count:n {##1} > 3}
    {

      \int_incr:N \l__cal_tmpa_int
      \int_zero:N \l__cal_tmpb_int

      \tl_set:Nn \l__cal_tmp_path_tl {##1}
      \spath_open:N \l__cal_tmp_path_tl
      \spath_reverse:NV \l__cal_tmp_rpath_tl \l__cal_tmp_path_tl

      \seq_map_inline:Nn #2
      {
        \int_incr:N \l__cal_tmpb_int
        \group_begin:
        \pgfsys@beginscope
        \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_initialpoint:Nn \l__cal_tmpa_tl {####1}
        \tl_set_eq:NN \l__cal_tmp_patha_tl \l__cal_tmp_path_tl
        \spath_translate:NV \l__cal_tmp_patha_tl \l__cal_tmpa_tl

        \int_compare:nTF {\tl_count:n {####1} == 3}
        {
          \cal_at_least_three:N \l__cal_tmp_patha_tl
          \spath_protocol_path:V \l__cal_tmp_patha_tl

          \tikz@options
          \dim_set:Nn \l__cal_line_width_dim {\pgflinewidth}
          \cal_maybe_taper:N \l__cal_tmp_patha_tl
        }
        {
          \spath_weld:Nn \l__cal_tmp_patha_tl {####1}
          \spath_weld:NV \l__cal_tmp_patha_tl \l__cal_tmp_rpath_tl
          \spath_reverse:Nn \l__cal_tmp_rpathb_tl {####1}
          \spath_weld:NV \l__cal_tmp_patha_tl \l__cal_tmp_rpathb_tl

          \tl_clear:N \l__cal_tmpa_tl
          \tl_set:Nn \l__cal_tmpa_tl
          {
            fill=\pgfkeysvalueof{/tikz/pen~colour},
            draw=none
          }
          \tl_if_exist:cT  {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_int}
          {
            \tl_put_right:Nv \l__cal_tmpa_tl
            {l__cal_stroke_style_ \int_use:N \l__cal_tmpa_int}
          }
          \tl_if_exist:cT  {l__cal_stroke_inline_style_ \int_use:N \l__cal_tmpa_int}
          {
            \tl_put_right:Nn \l__cal_tmpa_tl {,}
            \tl_put_right:Nv \l__cal_tmpa_tl
            {l__cal_stroke_inline_style_ \int_use:N \l__cal_tmpa_int}
          }
          \tl_if_exist:cT  {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}
          {
            \tl_put_right:Nn \l__cal_tmpa_tl {,}
            \tl_put_right:Nv \l__cal_tmpa_tl
            {l__cal_nib_style_ \int_use:N \l__cal_tmpb_int}
          }
          \spath_tikz_path:VV \l__cal_tmpa_tl \l__cal_tmp_patha_tl
        }
        \pgfsys@endscope
        \group_end:
      }

      \bool_if:NT \l__cal_annotate_bool
      {
        \seq_get_right:NN #2 \l__cal_tmpa_tl
        \spath_finalpoint:NV \l__cal_tmpa_tl \l__cal_tmpa_tl
        \spath_translate:NV \l__cal_tmp_path_tl \l__cal_tmpa_tl
        \tikz@scan@one@point
        \pgfutil@firstofone
        \pgfkeysvalueof{/tikz/annotation~shift}

        \spath_translate:Nnn \l__cal_tmp_path_tl {\pgf@x} {\pgf@y}

        \pgfkeysgetvalue{/tikz/annotation~style}{\l__cal_tmpa_tl}
        \spath_tikz_path:VV \l__cal_tmpa_tl \l__cal_tmp_path_tl

        \spath_finalpoint:NV \l__cal_tmpa_tl \l__cal_tmp_path_tl
```

```

        \exp_last_unbraced:NV \pgfqpoint \l__cal_tmpa_tl
        \begin{scope}[reset~ cm]
        \node[
          every~annotation~node/.try,
          tl~use:c = {l__cal_annotation_style_ \int_use:N \l__cal_tmpa_int _tl}
        ] at (\pgf@x,\pgf@y) {\int_use:N \l__cal_tmpa_int};
        \end{scope}
      }
    }
  }
}
\cs_generate_variant:Nn \cal_path_create:NN {Nc}
```

(*End of definition for* `\cal_path_create:NN`*.*)

`\cal_moveto:n`

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

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

(*End of definition for* `\cal_moveto:n`*.*)

`\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}
```

(*End of definition for* `\cal_apply_style:N`*.*)

`\cal_at_least_three:Nn`

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

```
\cs_new_protected_nopar:Npn \cal_at_least_three:Nn #1#2
{
  \spath_reallength:Nn \l__cal_tmpa_int {#2}
  \tl_clear:N \l__cal_tmpb_tl
  \tl_set:Nn \l__cal_tmpb_tl {#2}
  \int_compare:nTF {\l__cal_tmpa_int = 1}
  {
    \spath_split_at:Nn \l__cal_tmpb_tl {2/3}
    \spath_split_at:Nn \l__cal_tmpb_tl {1/2}
  }
  {
    \int_compare:nT {\l__cal_tmpa_int = 2}
    {
      \spath_split_at:Nn \l__cal_tmpb_tl {1.5}
      \spath_split_at:Nn \l__cal_tmpb_tl {.5}
    }
  }
  \tl_set_eq:NN #1 \l__cal_tmpb_tl
}
\cs_generate_variant:Nn \cal_at_least_three:Nn {NV}
\cs_new_protected_nopar:Npn \cal_at_least_three:N #1
{
  \cal_at_least_three:NV #1#1
}
\cs_generate_variant:Nn \cal_at_least_three:N {c}
```

(*End of definition for* `\cal_at_least_three:Nn`.)

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

```
\cs_new_protected_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}}

    \token_case_meaning:NnF \l__cal_tmpb_tl
    {
      \c_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 \c_spath_moveto_tl
      }
      \c_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 \c_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}}

  \token_case_meaning:NnF \l__cal_tmpb_tl
  {
    \c_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
    }
    \c_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}

}
```

(*End of definition for* `\cal_maybe_taper:N`.)

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

```
\cs_new_protected_nopar:Npn \__cal_taper_aux:
{
  \tl_clear:N \l__cal_tmpb_tl
  \tl_put_right:NV \l__cal_tmpb_tl \c_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
  }
  \fp_set:Nn \l__cal_tmpb_fp
  {
    .5*\l__cal_taper_width_dim
    *
    \l__cal_tmpb_fp / \l__cal_tmpe_fp
```

```
}

\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 \c_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 \c_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 \c_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 \c_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__cal_tmpd_fp
      }
    }
  }
  {
    \dim_eval:n
    {
      \fp_to_dim:N \l__cal_tmpd_fp + \l__cal_tmph_dim
      + \fp_to_dim:n {1.32* \l__cal_tmpc_fp
      }
    }
  }
}

\tl_put_right:NV \l__cal_tmpb_tl \c_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__cal_tmpd_fp
      }
    }
  }
  {
    \dim_eval:n
    {
      -\fp_to_dim:N \l__cal_tmpd_fp + \l__cal_tmph_dim
      + \fp_to_dim:n {1.32 * \l__cal_tmpc_fp
      }
    }
  }
}

\tl_put_right:NV \l__cal_tmpb_tl \c_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 \c_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 \c_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 \c_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 \c_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__cal_tmpb_fp}
    }
  }
  {
    \dim_eval:n
    {
      -\fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim
      - \fp_to_dim:n {1.32* \l__cal_tmpa_fp}
    }
  }
}

\tl_put_right:NV \l__cal_tmpb_tl \c_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__cal_tmpb_fp}
      }
    }
    {
      \dim_eval:n
      {
        \fp_to_dim:N \l__cal_tmpb_fp + \l__cal_tmpb_dim
        - \fp_to_dim:n {1.32 * \l__cal_tmpa_fp}
      }
    }
  }

  \tl_put_right:NV \l__cal_tmpb_tl \c_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}
}
```

(*End of definition for* `\__cal_taper_aux:`*.*)

Defines a copperplate pen.

```
\tl_set:Nn \l__cal_tmpa_tl {\pgfsyssoftpath@movetotoken{0pt}{0pt}}
\spath_components_to_seq:NV \l__cal_tmpa_seq \l__cal_tmpa_tl
\seq_gclear_new:N \g__cal_pen_copperplate_seq
\seq_gset_eq:NN \g__cal_pen_copperplate_seq \l__cal_tmpa_seq
```

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

```
\DeclareDocumentCommand \CopperplatePath { m }
{
  \spath_components_to_seq:NV \l__cal_tmpa_seq #1
  \cal_path_create:NN \l__cal_tmpa_seq \g__cal_pen_copperplate_seq
}
```

(*End of definition for* `\CopperplatePath`*.*)

```
\ExplSyntaxOff
```

5.4 Decorations

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

First, a brace decoration.

```
\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\pgfdecorationsegmentamplitude}%
      }%
      \pgfpathcurveto
      {%
        \pgfqpoint%
        {-.5\pgfdecorationsegmentamplitude}%
        {.5\pgfdecorationsegmentamplitude}%
      }%
      {%
        \pgfqpoint%
        {-.15\pgfdecorationsegmentamplitude}%
        {.7\pgfdecorationsegmentamplitude}%
      }%
      {%
        \pgfqpoint%
        {0\pgfdecorationsegmentamplitude}%
        {1\pgfdecorationsegmentamplitude}%
      }%
      \pgfpathmoveto%
      {%
        \pgfqpoint%
        {0\pgfdecorationsegmentamplitude}%
        {1\pgfdecorationsegmentamplitude}%
      }%
      \pgfpathcurveto%
      {%
        \pgfqpoint%
        {.15\pgfdecorationsegmentamplitude}%
        {.7\pgfdecorationsegmentamplitude}%
      }%
      {%
        \pgfqpoint%
        {.5\pgfdecorationsegmentamplitude}%
        {.5\pgfdecorationsegmentamplitude}%
      }%
      {%
        \pgfqpoint%
        {\pgfdecorationsegmentamplitude}%
        {.5\pgfdecorationsegmentamplitude}%
      }%
    }%
    {%
      \pgftransformxshift{+\pgfdecoratedremainingdistance}%
      \pgfpathlineto%
      {%
        \pgfqpoint%
        {-\pgfdecorationsegmentamplitude}%
        {.5\pgfdecorationsegmentamplitude}%
      }%
      \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%
    \CopperplatePath{\cal@tmp@path}%
  }%
  \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}%
        {\pgfdecorationsegmentamplitude}%
      }%
      \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%
    \CopperplatePath{\cal@tmp@path}%
  }%
  \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%
    \CopperplatePath{\cal@tmp@path}%
  }%
  \state{final}{}%
}
```

End the conditional for if pgfdecoration module is loaded

```
\fi
```

6 Drawing Knots

```
⟨@@=knot⟩
```

6.1 Initialisation

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

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

\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_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
\tl_new:N \l__knot_aux_tl
\tl_new:N \l__knot_auxa_tl
\tl_new:N \l__knot_prefix_tl

\seq_new:N \l__knot_segments_seq

\int_new:N \l__knot_tmpa_int
\int_new:N \l__knot_strands_int
\int_new:N \g__knot_intersections_int
\int_new:N \g__knot_filaments_int
\int_new:N \l__knot_component_start_int

\fp_new:N \l__knot_tmpa_fp
```

```
\fp_new:N \l__knot_tmpb_fp

\dim_new:N \l__knot_tmpa_dim
\dim_new:N \l__knot_tmpb_dim
\dim_new:N \l__knot_tolerance_dim
\dim_new:N \l__knot_redraw_tolerance_dim
\dim_new:N \l__knot_clip_bg_radius_dim
\dim_new:N \l__knot_clip_draw_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_new:N \l__knot_save_bool
\bool_new:N \l__knot_debugging_bool

\seq_new:N \g__knot_nodes_seq

\bool_set_true:N \l__knot_ignore_ends_bool
```

Configuration is via TikZ keys and styles.

```
\tikzset{
  spath/prefix/knot/.style={
    spath/set~ prefix=knot strand,
  },
  spath/suffix/knot/.style={
    spath/set~ suffix={},
  },
  knot/.code={
    \tl_if_eq:nnTF {#1} {none}
    {
      \tikz@addmode{\tikz@mode@doublefalse}
    }
    {
      \tikz@addmode{\tikz@mode@doubletrue}
      \tl_if_eq:nnTF {\pgfkeysnovalue} {#1}
      {
        \tikz@addoption{\pgfsetinnerstrokecolor{.}}
      }
      {
        \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},
save~ intersections/.is~ choice,
save~ intersections/.default=true,
save~ intersections/true/.code={
  \bool_set_true:N \l__knot_save_bool
},
save~ intersections/false/.code={
  \bool_set_false:N \l__knot_save_bool
},
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_bg_radius_dim {#1}
  \dim_set:Nn \l__knot_clip_draw_radius_dim {#1+2pt}
},
clip~ draw~ radius/.code={
  \dim_set:Nn \l__knot_clip_draw_radius_dim {#1}
},
clip~ background~ radius/.code={
  \dim_set:Nn \l__knot_clip_bg_radius_dim {#1}
},
clip~ radius=10pt,
end~ tolerance/.code={
  \dim_set:Nn \l__knot_tolerance_dim {#1}
},
end~ tolerance=14pt,
clip/.style={
  clip
},
background~ clip/.style={
  clip
},
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}
},
ignore~ crossing/.code={%
  \tl_clear_new:c {l__knot_ignore_crossing_#1}
  \tl_set:cn {l__knot_ignore_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
},
debug/.is~ choice,
debug/true/.code={
  \bool_set_true:N \l__knot_debugging_bool
},
debug/false/.code={
  \bool_set_false:N \l__knot_debugging_bool
},
debug/.default=true,
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
  },
}
```

`\knot_debug:n` Debugging

```
\cs_new_nopar:Npn \knot_debug:n #1
{
  \bool_if:NT \l__knot_debugging_bool
  {
    \iow_term:n {===Knot~ debug: #1===}
  }
}

\cs_new_nopar:Npn \knot_show_strand:n #1
{
  \bool_if:NT \l__knot_debugging_bool
  {
    \iow_term:n {===Knot~ debug: #1===}
    \spath_show:v {knot #1}
  }
}

\cs_generate_variant:Nn \knot_debug:n {x}
```

(*End of definition for* `\knot_debug:n`*.*)

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
{
  \knot_debug:n {knot~ apply~ style}
  \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}}%
}
```

(*End of definition for* `\flipcrossings`.)

`\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,spath/set~ name=knot,spath/save=\int_use:N \l__knot_strands_int]
}
```

(*End of definition for* `\strand`.)

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

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

(*End of definition for* `knot`.)

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

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

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

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

```
\cs_new_protected_nopar:Npn \knot_render:
{
  \knot_debug:n {knot~ render}
```

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

```
  \pgfscope
  \pgftransformreset
```

Set the dimension for deciding when to include neighbouring strands

```
  \dim_set:Nn \l__knot_redraw_tolerance_dim {\fp_to_dim:n
    {
      sqrt(2) * max(\l__knot_clip_bg_radius_dim, \l__knot_clip_draw_radius_dim)
    }
  }
```

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 \g__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 \g__knot_intersections_int
```

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

```
    \tl_clear:N \l__knot_node_tl
    \bool_if:NT \l__knot_draft_bool
    {
      \tl_set:Nn \l__knot_node_tl {
        \exp_not:N \node[coordinate,
          pin={[
              node~ contents={\int_use:N \g__knot_intersections_int},
              knot~ diagram/draft/crossing~ label,
              knot~ diagram/draft/crossing~
              \int_use:N \g__knot_intersections_int \c_space_tl label/.try
            ]
            }]
      }
    }
```

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_tmpb_tl
      {
        \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
```

Draw the crossing nodes

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

Close the scope

```
  \endpgfscope
  \knot_debug:x {knot~rendered,
    ~found~\int_use:N  \g__knot_intersections_int \c_space_tl~intersections}
}
```

(*End of definition for* `\knot_render:`.)

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

```
\cs_new_protected_nopar:Npn \knot_draw_strand:n #1
{
  \knot_debug:n {knot~ draw~ strand~ #1}
  \pgfscope
  \group_begin:
  \spath_bake_round:c {knot strand #1}
  \tl_set:Nn \l__knot_tmpa_tl {knot~ diagram/every~ strand/.try,}
  \tl_put_right:Nv \l__knot_tmpa_tl {l__knot_options_strand #1}
  \tl_put_right:Nn \l__knot_tmpa_tl
  {
    ,
    knot~ diagram/only~ when~ rendering/.try,
    only~ when~ rendering/.try,
  }
  \knot_show_strand:n {strand #1}
  \spath_tikz_path:Vv \l__knot_tmpa_tl {knot strand #1}
  \group_end:
  \endpgfscope
}
\cs_generate_variant:Nn \tl_put_right:Nn {Nv}
```

(*End of definition for* `\knot_draw_strand:n`.)

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

```
\cs_new_protected_nopar:Npn \knot_draw_labels:n #1
{
  \knot_debug:n {knot~ draw~ labels}
  \bool_if:NT \l__knot_draft_bool
  {
    \spath_finalpoint:Nv \l__knot_tmpb_tl {knot strand #1}
    \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_initialpoint:Nv \l__knot_tmpb_tl {knot strand #1}
    \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_remove_empty_components:c {knot strand #1}
    \spath_initialpoint:Nv \l__knot_tmpa_tl {knot strand #1}
    \tl_put_left:NV \l__knot_tmpa_tl \c_spath_moveto_tl
    \spath_segments_to_seq:Nv \l__knot_segments_seq {knot strand #1}
    \seq_map_function:NN \l__knot_segments_seq \knot_split_self_intersects:N
    \tl_set_eq:cN {knot strand #1} \l__knot_tmpa_tl
  }
}
```

(*End of definition for* `\knot_draw_labels:n`.)

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

```
\cs_new_protected_nopar:Npn \knot_split_self_intersects:N #1
{
  \knot_debug:n {knot~ split~ self~ intersects}
  \tl_set:Nx \l__knot_tmpc_tl {\tl_item:nn {#1} {4}}
  \token_case_meaning:NnF \l__knot_tmpc_tl
  {
    \c_spath_curvetoa_tl
    {
      \fp_set:Nn \l__knot_tmpa_fp
      {
        (\tl_item:nn {#1} {3} - 3 * \tl_item:nn {#1} {6}
        + 3 * \tl_item:nn {#1} {9} - \tl_item:nn {#1} {12})
        *
        (3 * \tl_item:nn {#1} {8} - 3 * \tl_item:nn {#1} {11})
        -
        (\tl_item:nn {#1} {2} - 3 * \tl_item:nn {#1} {5}
        + 3 * \tl_item:nn {#1} {8} - \tl_item:nn {#1} {11})
        *
        (3 * \tl_item:nn {#1} {9} - 3 * \tl_item:nn {#1} {12})
      }
      \fp_set:Nn \l__knot_tmpb_fp
      {
        (\tl_item:nn {#1} {2} - 3 * \tl_item:nn {#1} {5}
        + 3 * \tl_item:nn {#1} {8} - \tl_item:nn {#1} {11})
        *
        (3 * \tl_item:nn {#1} {6} - 6 * \tl_item:nn {#1} {9}
        + 3 * \tl_item:nn {#1} {12})
        -
        (\tl_item:nn {#1} {3} - 3 * \tl_item:nn {#1} {6}
        + 3 * \tl_item:nn {#1} {9} - \tl_item:nn {#1} {12})
        *
        (3 * \tl_item:nn {#1} {5} - 6 * \tl_item:nn {#1} {8}
        + 3 * \tl_item:nn {#1} {11})
      }
      \fp_compare:nTF
      {
        \l__knot_tmpb_fp != 0
      }
      {
        \fp_set:Nn \l__knot_tmpa_fp {.5 * \l__knot_tmpa_fp / \l__knot_tmpb_fp}
        \bool_if:nTF
        {
          \fp_compare_p:n {0 < \l__knot_tmpa_fp}
          &&
          \fp_compare_p:n {\l__knot_tmpa_fp < 1}
        }
        {
          \spath_split_curve:NNnV
          \l__knot_tmpc_tl
          \l__knot_tmpd_tl
          {#1}
          \l__knot_tmpa_fp
          \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
          \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
          \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
          \tl_set:Nx \l__knot_tmpd_tl {\tl_tail:N \l__knot_tmpd_tl}
          \tl_set:Nx \l__knot_tmpd_tl {\tl_tail:N \l__knot_tmpd_tl}
          \tl_set:Nx \l__knot_tmpd_tl {\tl_tail:N \l__knot_tmpd_tl}
          \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpc_tl
          \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpd_tl
        }
        {
          \tl_set:Nn \l__knot_tmpc_tl {#1}
          \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
          \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
          \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
          \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpc_tl
        }
      }
      {
        \tl_set:Nn \l__knot_tmpc_tl {#1}
        \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
        \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
        \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
        \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpc_tl
      }
    }
    \c_spath_lineto_tl
    {
      \tl_set:Nn \l__knot_tmpc_tl {#1}
      \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
      \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
```

```
      \tl_set:Nx \l__knot_tmpc_tl {\tl_tail:N \l__knot_tmpc_tl}
      \tl_put_right:NV \l__knot_tmpa_tl \l__knot_tmpc_tl
    }
  }
  {
    \tl_put_right:Nn \l__knot_tmpa_tl {#1}
  }
}
```

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

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

```
\cs_new_protected_nopar:Npn \knot_intersections:nn #1#2
{
  \knot_debug:x {knot~ intersections~ between~
    \l__knot_prefix_tl \c_space_tl #1~ and~ #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}
  \tl_set_eq:Nc \l__knot_tmpc_tl {knot \tl_use:N \l__knot_tmpa_tl}
  \tl_set_eq:Nc \l__knot_tmpd_tl {knot \tl_use:N \l__knot_tmpb_tl}

  \bool_if:nTF {
    \l__knot_save_bool
    &&
    \tl_if_exist_p:c {
      knot~ intersections~
      \tl_use:N \l__knot_name_tl -
      \tl_use:N \l__knot_tmpa_tl -
      \tl_use:N \l__knot_tmpb_tl
    }
  }
  {
    \tl_use:c
    {
      knot~ intersections~ \tl_use:N \l__knot_name_tl -
      \tl_use:N \l__knot_tmpa_tl -
      \tl_use:N \l__knot_tmpb_tl
    }
  }
  {
    \pgfintersectionofpaths{\pgfsetpath\l__knot_tmpc_tl}{\pgfsetpath\l__knot_tmpd_tl}

  }

  \knot_debug:x {found~\pgfintersectionsolutions\c_space_tl~ intersections}
  \int_compare:nT {\pgfintersectionsolutions > 0}
  {
    \int_step_function:nnnN
    {1}
    {1}
    {\pgfintersectionsolutions}
    \knot_do_intersection:n
  }

  \knot_save_intersections:VV \l__knot_tmpa_tl \l__knot_tmpb_tl
  \group_end:
}
```

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

\knot_save_intersections:nn

```
\cs_new_protected_nopar:Npn \knot_save_intersections:nn #1#2
{
  \knot_debug:n {knot~ save~ intersections}
  \bool_if:NT \l__knot_save_bool
  {
    \tl_clear:N \l__knot_aux_tl
    \tl_put_right:Nn \l__knot_aux_tl
    {
      \def\pgfintersectionsolutions
    }
    \tl_put_right:Nx \l__knot_aux_tl
    {
      {\int_eval:n {\pgfintersectionsolutions}}
    }
    \int_compare:nT {\pgfintersectionsolutions > 0}
    {
      \int_step_inline:nnnn {1} {1} {\pgfintersectionsolutions}
      {
        \pgfpointintersectionsolution{##1}
        \dim_set:Nn \l__knot_tmpa_dim {\pgf@x}
        \dim_set:Nn \l__knot_tmpb_dim {\pgf@y}
        \tl_put_right:Nn \l__knot_aux_tl
        {
          \expandafter\def\csname pgfpoint@intersect@solution@##1\endcsname
        }
        \tl_put_right:Nx \l__knot_aux_tl
        {
          {
            \exp_not:N \pgf@x
            =
            \dim_use:N \l__knot_tmpa_dim
            \exp_not:N \relax
            \exp_not:N \pgf@y
            =
            \dim_use:N \l__knot_tmpb_dim
            \exp_not:N \relax
          }
        }
      }
      \tl_set:Nn \l__knot_auxa_tl {\expandafter \gdef \csname knot~ intersections~}
      \tl_put_right:Nx \l__knot_auxa_tl {\tl_use:N \l__knot_name_tl - #1 - #2}
      \tl_put_right:Nn \l__knot_auxa_tl {\endcsname}
      \tl_put_right:Nx \l__knot_auxa_tl {{\tl_to_str:N \l__knot_aux_tl}}
      \protected@write\@auxout{}{\tl_to_str:N \l__knot_auxa_tl}
    }
  }
}
\cs_generate_variant:Nn \knot_save_intersections:nn {VV}
```

(*End of definition for* `\knot_save_intersections:nn`*.*)

`\knot_do_intersection:n`

This handles a specific intersection.

```
\cs_new_protected_nopar:Npn \knot_do_intersection:n #1
{
  \knot_debug:n {knot~ do~ intersection~ #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}
  \knot_debug:x {intersection~at~
    (\dim_use:N \l__knot_tmpa_dim,\dim_use:N \l__knot_tmpb_dim)}
```

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__knot_tmpc_tl {knot previous}
    \tl_put_right:NV \l__knot_tmpc_tl \l__knot_tmpa_tl
    \tl_set:Nv \l__knot_tmpc_tl \l__knot_tmpc_tl
    \tl_if_eq:NNT \l__knot_tmpc_tl \l__knot_tmpb_tl
    {
      \knot_test_endpoint:NVnT \l__knot_tolerance_dim \l__knot_tmpb_tl {final point}
      {
        \bool_set_true:N \l__knot_skip_bool
      }
    }

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

The user 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:NVnT \l__knot_tolerance_dim \l__knot_tmpa_tl {initial point}
    {
      \bool_set_true:N \l__knot_skip_bool
    }
    \knot_test_endpoint:NVnT \l__knot_tolerance_dim \l__knot_tmpa_tl {final point}
    {
      \bool_set_true:N \l__knot_skip_bool
    }
    \knot_test_endpoint:NVnT \l__knot_tolerance_dim \l__knot_tmpb_tl {initial point}
    {
      \bool_set_true:N \l__knot_skip_bool
    }
    \knot_test_endpoint:NVnT \l__knot_tolerance_dim \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 \g__knot_intersections_int
    \knot_debug:x {Processing~intersection~\int_use:N \g__knot_intersections_int}
```

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

```
    \bool_if:nF
    {
      \tl_if_exist_p:c {l__knot_ignore_crossing_ \int_use:N
        \g__knot_intersections_int}
      &&
      ! \tl_if_empty_p:c {l__knot_ignore_crossing_ \int_use:N
        \g__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
          \g__knot_intersections_int}
        &&
        ! \tl_if_empty_p:c {l__knot_crossing_ \int_use:N
          \g__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:NVnT
        \l__knot_redraw_tolerance_dim \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:NVnT
        \l__knot_redraw_tolerance_dim \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
        }
        {
          \tl_clear_new:c {knot \tl_use:N \l__knot_prefix_tl -1}
          \tl_set_eq:cc
          {knot \tl_use:N \l__knot_prefix_tl -1}
          {knot \tl_use:N \l__knot_tmpg_tl}

          \tl_clear_new:c {l__knot_options_ \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_use:N \l__knot_tmpg_tl}

          \bool_if:nT
          {
            \l__knot_prepend_prev_bool
            &&
            \tl_if_exist_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
            &&
            !\tl_if_empty_p:c {knot previous \tl_use:N \l__knot_tmpg_tl}
          }
          {
            \knot_debug:x {Prepending~
              \tl_use:c {knot previous \tl_use:N \l__knot_tmpg_tl}}
            \spath_prepend_no_move:cv
            {knot \tl_use:N \l__knot_prefix_tl -1}
            {knot \tl_use:c {knot previous \tl_use:N \l__knot_tmpg_tl}}
```

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

```
            \bool_if:nT
            {
              \l__knot_splits_bool
              &&
              \tl_if_exist_p:c {knot previous
                \tl_use:c {knot previous \tl_use:N \l__knot_tmpg_tl}
              }
              &&
              !\tl_if_empty_p:c {knot previous
                \tl_use:c {knot previous \tl_use:N \l__knot_tmpg_tl}
              }
            }
            {
              \knot_test_endpoint:NvnT
              \l__knot_redraw_tolerance_dim
              {knot previous \tl_use:N \l__knot_tmpg_tl}
              {initial point}
              {
                \knot_debug:x {Prepending~
                  \tl_use:c {knot previous
                    \tl_use:c {knot previous \tl_use:N \l__knot_tmpg_tl}
                  }
                }
                \spath_prepend_no_move:cv
                {knot \tl_use:N \l__knot_prefix_tl -1}
                {knot \tl_use:c
                  {knot previous \tl_use:c
                    {knot previous \tl_use:N \l__knot_tmpg_tl}
                  }
                }
                \tl_set_eq:Nc \l__knot_tmpa_tl
                {knot \tl_use:N \l__knot_prefix_tl -1}
              }
            }
          }

```

Now the same for appending.

```
    \bool_if:nT
    {
      \l__knot_append_next_bool
      &&
      \tl_if_exist_p:c {knot next \tl_use:N \l__knot_tmpg_tl}
      &&
      !\tl_if_empty_p:c {knot next \tl_use:N \l__knot_tmpg_tl}
    }
    {
      \knot_debug:x {Appending~
        \tl_use:c {knot next \tl_use:N \l__knot_tmpg_tl}}
      \spath_append_no_move:cv
      {knot \tl_use:N \l__knot_prefix_tl -1}
      {knot \tl_use:c {knot next \tl_use:N \l__knot_tmpg_tl}}
      \bool_if:nT
      {
        \l__knot_splits_bool
        &&
        \tl_if_exist_p:c {knot next \tl_use:c { knot next \tl_use:N
          \l__knot_tmpg_tl}}
        &&
        !\tl_if_empty_p:c {knot next
          \tl_use:c { knot next \tl_use:N \l__knot_tmpg_tl}
        }
      }
      {
        \knot_debug:x {Testing~ whether~ to~ append~
          {knot next \tl_use:c { knot next \tl_use:N \l__knot_tmpg_tl}}
        }
        \knot_test_endpoint:NvnT
        \l__knot_redraw_tolerance_dim
        {knot next \tl_use:N \l__knot_tmpg_tl}
        {final point}
        {
          \knot_debug:x {Appending~
            {knot next \tl_use:c { knot next \tl_use:N \l__knot_tmpg_tl}}
          }
          \spath_append_no_move:cv
          {knot \tl_use:N \l__knot_prefix_tl -1}
          {knot \tl_use:c
            {knot next \tl_use:c
              {knot next \tl_use:N \l__knot_tmpg_tl}
            }
          }
        }
      }
    }
    \tl_set:Nn \l__knot_tmpg_tl {\tl_use:N \l__knot_prefix_tl -1}
  }
}
```

Now we render the crossing.

```
      \pgfscope
      \group_begin:
      \tikzset{
        knot~ diagram/every~ intersection/.try,
        every~ intersection/.try,
        knot~ diagram/intersection~ \int_use:N \g__knot_intersections_int/.try
      }
      \knot_draw_crossing:VVV \l__knot_tmpg_tl \l__knot_tmpa_dim \l__knot_tmpb_dim
      \coordinate
      (\l__knot_name_tl \c_space_tl \int_use:N \g__knot_intersections_int)
      at (\dim_use:N \l__knot_tmpa_dim, \dim_use:N \l__knot_tmpb_dim);
      \group_end:
      \endpgfscope
```

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

```
    }
```

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

```
    \tl_if_empty:NF \l__knot_node_tl
    {
      \seq_gpush:Nx
      \g__knot_nodes_seq
      {
        \l__knot_node_tl
        at
        (\dim_use:N \l__knot_tmpa_dim, \dim_use:N \l__knot_tmpb_dim) {};
      }
    }
  }
}

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

(*End of definition for* `\knot_do_intersection:n`.)

`\knot_test_endpoint:N`

Test whether the point is near the intersection point.

```
\prg_new_conditional:Npnn \knot_test_endpoint:NN #1#2 {p,T,F,TF}
{
  \dim_compare:nTF
  {
    \dim_abs:n { \l__knot_tmpa_dim - \tl_item:Nn #2 {1}}
    +
    \dim_abs:n { \l__knot_tmpb_dim - \tl_item:Nn #2 {2}}
    <
    #1
  }
  {
    \prg_return_true:
  }
  {
    \prg_return_false:
  }
}
```

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

`\knot_test_endpoint:nn`

Wrapper around the above.

```
\prg_new_protected_conditional:Npnn \knot_test_endpoint:Nnn #1#2#3 {T,F,TF}
{
  \use:c {spath_#3:Nv} \l__knot_tmpd_tl {knot #2}
  \knot_test_endpoint:NNTF #1 \l__knot_tmpd_tl
  {
    \prg_return_true:
  }
  {
    \prg_return_false:
  }
}

\cs_generate_variant:Nn \knot_test_endpoint:NnnT {NVnT,NvnT}
\cs_generate_variant:Nn \knot_test_endpoint:NnnF {NVnF,NvnF}
\cs_generate_variant:Nn \knot_test_endpoint:NnnTF {NVnTF,NvnTF}
```

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

`\knot_draw_crossing:nnn`

This is the code that actually renders a crossing.

```
\cs_new_protected_nopar:Npn \knot_draw_crossing:nnn #1#2#3
{
  \knot_debug:n {knot~ draw~ crossing}
  \group_begin:
  \pgfscope
  \path[knot~ diagram/background~ clip] (#2, #3)
  circle[radius=\l__knot_clip_bg_radius_dim];

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

  \endpgfscope

  \pgfscope
  \path[knot~ diagram/clip] (#2, #3)
  circle[radius=\l__knot_clip_draw_radius_dim];

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

  \endpgfscope
  \group_end:
}

\cs_generate_variant:Nn \knot_draw_crossing:nnn {nVV, VVV}

\cs_new_protected_nopar:Npn \knot_draw_crossing:nn #1#2
{
  \tikz@scan@one@point\pgfutil@firstofone #2 \relax
  \knot_draw_crossing:nVV {#1} \pgf@x \pgf@y
}
```

(*End of definition for* `\knot_draw_crossing:nnn`*.*)

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

```
\cs_new_protected_nopar:Npn \knot_split_strands:
{
  \knot_debug:n {knot~ split~ strands}
  \int_gzero:N \g__knot_filaments_int
  \int_step_function:nnnN {1} {1} {\l__knot_strands_int} \knot_split_strand:n
  \int_step_function:nnnN {1} {1} {\g__knot_filaments_int} \knot_compute_nexts:n
}
```

(*End of definition for* `\knot_split_strands:`.)

`\knot_compute_nexts:n` Each filament needs to know its predecessor and successor. We work out the predecessors as we go along, this fills in the successors.

```
\cs_new_protected_nopar:Npn \knot_compute_nexts:n #1
{
  \knot_debug:n {knot~ compute~ nexts}
  \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}
}
```

(*End of definition for* `\knot_compute_nexts:n`.)

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

```
\cs_new_protected_nopar:Npn \knot_split_strand:n #1
{
  \knot_debug:n {knot~ split~ strand}
  \int_set_eq:NN \l__knot_component_start_int \g__knot_filaments_int
  \int_incr:N \l__knot_component_start_int
  \tl_set_eq:Nc \l__knot_tmpa_tl {l__knot_options_strand #1}
  \spath_remove_empty_components:c {knot strand #1}
  \spath_segments_to_seq:Nv \l__knot_segments_seq {knot strand #1}
  \seq_map_function:NN \l__knot_segments_seq \knot_save_filament:N
}
```

(*End of definition for* `\knot_split_strand:n`.)

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

```
\cs_new_protected_nopar:Npn \knot_save_filament:N #1
{
  \knot_debug:n {knot~ save~ filament}
  \tl_set:Nx \l__knot_tmpb_tl {\tl_item:nn {#1} {4}}
  \token_case_meaning:NnF \l__knot_tmpb_tl
  {
    \c_spath_moveto_tl
    {
      \int_compare:nT {\l__knot_component_start_int < \g__knot_filaments_int}
      {
        \int_set_eq:NN \l__knot_component_start_int \g__knot_filaments_int
      }
    }
    \c_spath_lineto_tl
    {
      \int_gincr:N \g__knot_filaments_int
      \tl_clear_new:c {knot filament \int_use:N \g__knot_filaments_int}
      \tl_set:cn {knot filament \int_use:N \g__knot_filaments_int} {#1}

      \tl_clear_new:c {l__knot_options_filament \int_use:N \g__knot_filaments_int}
      \tl_set_eq:cN {l__knot_options_filament \int_use:N \g__knot_filaments_int}
      \l__knot_tmpa_tl

      \tl_clear_new:c {knot previous filament \int_use:N \g__knot_filaments_int}
      \int_compare:nF {\l__knot_component_start_int == \g__knot_filaments_int}
      {
        \tl_set:cx {knot previous filament \int_use:N \g__knot_filaments_int}
        {filament \int_eval:n {\g__knot_filaments_int - 1}}
      }
    }
    \c_spath_curvetoa_tl
    {
      \int_gincr:N \g__knot_filaments_int
      \tl_clear_new:c {knot filament \int_use:N \g__knot_filaments_int}
      \tl_set:cn {knot filament \int_use:N \g__knot_filaments_int} {#1}
      \tl_clear_new:c {l__knot_options_filament \int_use:N \g__knot_filaments_int}
      \tl_set_eq:cN {l__knot_options_filament \int_use:N \g__knot_filaments_int}
      \l__knot_tmpa_tl

      \tl_clear_new:c {knot previous filament \int_use:N \g__knot_filaments_int}
      \int_compare:nF {\l__knot_component_start_int == \g__knot_filaments_int}
      {
        \tl_set:cx
        {knot previous filament \int_use:N \g__knot_filaments_int}
        {filament \int_eval:n {\g__knot_filaments_int - 1}}
      }
    }
    \c_spath_closepath_tl
    {
      \int_gincr:N \g__knot_filaments_int
      \tl_clear_new:c {knot filament \int_use:N \g__knot_filaments_int}
      \tl_clear:N \l__knot_tmpa_tl
      \tl_put_right:Nx
      {
        \tl_item:nn {#1} {1}\tl_item:nn {#1} {2}\tl_item:nn {#1} {3}
      }
      \tl_put_right:NV \l__knot_tmpa_tl \c_spath_lineto_tl
      \tl_put_right:Nx {\tl_item:nn {#1} {5}\tl_item:nn {#1} {6}}

      \tl_set:cV {knot filament \int_use:N \g__knot_filaments_int} \l__knot_tmpa_tl
      \tl_set_eq:cN {l__knot_options_filament \int_use:N \g__knot_filaments_int}
      \l__knot_tmpa_tl
      \tl_clear_new:c {knot previous filament \int_use:N \g__knot_filaments_int}
      \int_compare:nF {\l__knot_component_start_int == \g__knot_filaments_int}
      {
        \tl_set:cx
        {knot previous filament \int_use:N \g__knot_filaments_int}
        {filament \int_eval:n {\g__knot_filaments_int - 1}}
      }
      \tl_set:cx
      {knot previous filament \int_use:N \l__knot_component_start_int}
      {filament \int_use:N \g__knot_filaments_int}
    }
  }
  {
  }
}
```

(*End of definition for* `\knot_save_filament:N`.)

`\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:nn}
  \tl_put_right:Nx \l__knot_redraws_tl {
    {strand #1} {#2}% {\dim_use:N \pgf@x} {\dim_use:N \pgf@y}
  }
}
```

(*End of definition for* `\redraw`.)

```
\ExplSyntaxOff
```

<@@=>

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

```
\def\pgf@sh__knotknotanchor#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%
  }%
}
```

(*End of definition for* `\pgf@sh__knotknotanchor`.)

`knot␣crossing`

```
\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__knotknotanchor{crossing}{2}
  \pgf@sh__knotknotanchor{crossing}{3}
  \pgf@sh__knotknotanchor{crossing}{4}
  \pgf@sh__knotknotanchor{crossing}{8}
  \pgf@sh__knotknotanchor{crossing}{16}
  \pgf@sh__knotknotanchor{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%
  }
}
```

(*End of definition for* `knot crossing`.)

`knot␣over␣cross`

```
\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}}
 }
}
```

(*End of definition for* `knot over cross`*.*)

`knot␣under␣cross`

```
\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}}
 }
}
```

(*End of definition for* `knot under cross`*.*)

knot␣vert

```
\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}}
 }
}
```

(*End of definition for* `knot vert`*.*)

knot␣horiz

```
\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}}
 }
}
```

(*End of definition for* `knot horiz`*.*)