

The `spath` package

Andrew Stacey

stacey@math.ntnu.no

v1.0 from 2011/06/03

1 Introduction

2 Implementation

Load the `pgf.oo` module.

```
1 \usepgfmodule{oo}
```

Define some auxilliary macros (should replace these with `etoolbox`

```
\ge@addto@macro
```

```
2 \long\def\ge@addto@macro#1#2{%
3   \begingroup
4   \toks@{\expandafter\expandafter\expandafter{\expandafter#1#2}%
5   \xdef#1{\the\toks@}%
6   \endgroup}
7
8 \def\tikz@clear@foreach{%
9   \let\pgffor@beginhook=\pgfutil@empty
10  \let\pgffor@endhook=\pgfutil@empty
11  \let\pgffor@afterhook=\pgfutil@empty
12 }
13
```

```
\ge@addbefore@macro
```

```
14 \long\def\ge@addbefore@macro#1#2{%
15   \begingroup
16   \toks@{\expandafter\expandafter\expandafter{\expandafter#2#1}%
17   \xdef#1{\the\toks@}%
18   \endgroup}
```

```
\g@addbefore@macro
```

```
19 \long\def\g@addbefore@macro#1#2{%
20   \def\@temp{#2}%
21   \ge@addbefore@macro{#1}{\@temp}}
```

```

22 \tikzset{%
23   use path/.code={%
24     \pgfsyssoftpath@setcurrentpath{#1}%
25   }
26 }

```

This is the `spath` object. It is defined using the `pgf.oo` module. It represents a “soft path”.

```

27 \pgfooclass{spath}{

```

Certain attributes of the soft path are saved to avoid having to recompute them for every operation. The `path` attribute is the soft path itself.

```

28   \attribute path;

```

The `length` is the crudest measure of length: the number of soft path tokens in the path.

```

29   \attribute length;

```

Since some path pieces consist of more than one token, the `length` is longer than what one might call the path length. The `real length` counts only the key tokens.

```

30   \attribute real length;

```

A path can be viewed as being a collection of drawing tokens and moving tokens. We call a collection of subsequent drawing tokens a `component`. The `number of components` counts this. (Actually, it counts the number of `movetos`, which could be more if there are successive `movetos`.)

```

31   \attribute number of components;

```

The `initial point` is where the path starts.

```

32   \attribute initial point;

```

The `final point` is where the path ends.

```

33   \attribute final point;

```

The `first action` is the first drawing action on the path (so not including the initial `moveto`.)

```

34   \attribute first action;

```

The `last action` is the last action on the path (which might be a `moveto`.)

```

35   \attribute last action;

```

To compute most of these attributes, we need to walk along the path. Obviously, we save their values to avoid having to do this walk too often. The `prepared` attribute tells us whether or not we’ve already done this walk.

```

36   \attribute prepared;

```

For the path tapering, we need to know the line width and the width to taper to. The `taper line width` holds the latter.

```

37   \attribute taper line width;

```

We need to keep track of our bounding box

```

38   \attribute min bb;

```

```

39   \attribute max bb;

```

Our last attribute is a “scratch” attribute for saving some macro that relates to the object in some way.

```
40 \attribute scratch pad;
```

Now we have the methods for the `spath` object.

We start with the creator. If given a path, we store it as the `path` attribute.

```
41 \method spath(#1) {%
42   \let\spath@temp=#1\relax
43   \ifx\spath@temp\relax
44   \else
45     \pgfoolet{path}{#1}%
46   \fi
47 }
```

Set path from current path

```
48 \method use current path() {%
49   \pgfsyssoftpath@getcurrentpath{\spath@temp}%
50   \pgfoolet{path}{\spath@temp}%
51   \let\spath@temp=\pgfutil@empty
52 }
```

Next, we have some generic attribute handling methods. These allow us to set, get, and let attributes, and also to show them in the logs.

We start with one that inserts the value of the given attribute in the token stream.

```
53 \method value(#1) {%
54   \pgfoovalueof{#1}%
55 }
```

This sets the attribute (first argument) to the second argument.

```
56 \method set(#1,#2) {%
57   \pgfooset{#1}{#2}%
58 }
```

This lets the attribute (first argument) to the second argument, which must be a macro.

```
59 \method let(#1,#2) {%
60   \pgfoolet{#1}{#2}%
61 }
```

This sets the attribute (first argument) to the second argument if the attribute is not empty.

```
62 \method set if not empty(#1,#2) {%
63   \pgfooget{#1}{\spath@tmp}%
64   \ifx\spath@tmp\pgfutil@empty
65     \pgfooset{#1}{#2}%
66   \fi
67 }
```

This lets the attribute (first argument) to the second argument, which must be a macro, if the attribute is not empty.

```

68 \method let if not empty(#1,#2) {%
69   \pgfooget{#1}{\spath@tmp}%
70   \ifx\spath@tmp\pgfutil@empty
71     \pgfoolet{#1}{#2}%
72   \fi
73 }

```

This gets the attribute (first argument) into the second argument, which must be a macro.

```

74 \method get(#1,#2) {%
75   \pgfooget{#1}{#2}%
76 }

```

This shows the attribute (argument) in the logs, via `\show`. (Should prefix with the attribute name to be more useful.)

```

77 \method show(#1) {%
78   \pgfooget{#1}{\@temp}%
79   \show\@temp
80 }

```

This says what we are

```

81 \method what am I() {%
82   \message{I am an spath object}%
83 }

```

This clones the path to the macro passed to it. The macro must already exist (at the moment, though in thinking about it that's a little daft) but will be overwritten.

```

84 \method clone(#1) {%
85   \let\spath@newpath=#1\relax
86   \ifx\spath@newpath\relax
87     \else
88       \pgfoonew \spath@newpath =new spath()%
89   \tikz@clear@foreach
90   \foreach \attribute in {
91     path,
92     length,
93     real length,
94     number of components,
95     initial point,
96     final point,
97     first action,
98     last action,
99     prepared,
100    taper line width,
101    min bb,
102    max bb%,
103  } {
104    \pgfooget{\attribute}{\spath@temp}%
105    \spath@newpath.let(\attribute,\spath@temp)%
106  }

```

```

107   \let#1=\spath@newpath%
108   \fi
109 }

```

Now we begin the path-walking methods. These methods work by “evaluating” the path under certain conditions. A soft path consists of a series of commands which take two arguments, nominally an x-coordinate and a y-coordinate. By resetting the commands, we can make a soft path rewrite itself in a modified form, or to extract certain information from it. The actual implementation of these uses various auxilliary macros which are defined later.

This method translates the path by #2 in x and #3 in y. If #1 is given then it should be a macro and it will be turned into an object with the translated path, otherwise the current object is modified.

It works by redefining each soft path token to add the given dimensions to its arguments and then rewrite itself back to the path. Having done that, we have to adjust the various other attributes which are location-specific.

```

110 \method translate path(#1,#2,#3) {%
111   \let\spath@newpath=#1\relax
112   \ifx\spath@newpath\relax
113     \let\spath@tmppath=\pgfutil@empty
114     \begingroup
115       \spath@trx=#2\relax
116       \spath@try=#3\relax
117       \spath@translate@init
118       \pgfoovalueof{path}%
119       \pgfoolet{path}{\spath@tmppath}%
120       \let\spath@tmppath=\pgfutil@empty
121   \tikz@clear@foreach
122     \foreach \attribute in {
123       initial point,
124       final point,
125       min bb,
126       max bb%,
127     } {
128       \pgfooget{\attribute}{\spath@temp}%
129       \ifx\spath@temp\pgfutil@empty
130       \else
131         \spath@temp
132         \advance\pgf@x by \spath@trx\relax
133         \advance\pgf@y by \spath@try\relax
134         \edef\spath@temp{\noexpand\pgfpoint{\the\pgf@x}{\the\pgf@y}}%
135         \pgfoolet{\attribute}{\spath@temp}%
136       \fi
137     }
138   \endgroup
139   \let\spath@tmppath=\pgfutil@empty
140 \else
141   \pgfoothis.clone(\spath@newpath)%
142 \begingroup

```

```

143   \spath@newpath.translate path(,#2,#3)%
144   \endgroup
145   \let#1=\spath@newpath
146   \fi
147 }

Transform path according to an affine transformation
148 \method transform path(#1,#2) {%
149   \let\spath@newpath=#1\relax
150   \ifx\spath@newpath\relax
151     \let\spath@tmppath=\pgfutil@empty
152     \begingroup
153     \pgftransformreset
154     \pgfsettransform{#2}%
155     \spath@transform@init
156     \pgfoovalueof{path}%
157     \pgfoolet{path}{\spath@tmppath}%
158     \let\spath@tmppath=\pgfutil@empty
159   \tikz@clear@foreach
160     \foreach \attribute in {
161       initial point,
162       final point,
163       min bb,
164       max bb,
165       prepared%,
166     } {
167       \pgfoolet{\attribute}{\pgfutil@empty}%
168     }%
169   \endgroup
170   \let\spath@tmppath=\pgfutil@empty
171   \else
172     \pgfoothis.clone(\spath@newpath)%
173     \begingroup
174     \spath@newpath.transform path(,#2)%
175     \endgroup
176     \let#1=\spath@newpath
177   \fi
178 }

Scale all coordinates
179 \method scale path(#1,#2,#3) {%
180   \let\spath@newpath=#1\relax
181   \ifx\spath@newpath\relax
182     \let\spath@tmppath=\pgfutil@empty
183     \begingroup
184     \edef\spath@scx{#2}%
185     \edef\spath@scy{#3}%
186     \ifx\spath@scy\pgfutil@empty
187       \edef\spath@scy{#2}%
188     \fi
189     \spath@scale@init

```

```

190     \pgfoovalueof{path}%
191     \pgfoolet{path}{\spath@tmppath}%
192     \let\spath@tmppath=\pgfutil@empty
193 \tikz@clear@foreach
194     \foreach \attribute in {
195         initial point,
196         final point,
197         min bb,
198         max bb%,
199     } {
200         \pgfooget{\attribute}{\spath@temp}%
201         \ifx\spath@temp\pgfutil@empty
202         \else
203             \spath@temp
204             \pgf@x=\spath@scx\pgf@x\relax
205             \pgf@y=\spath@scy\pgf@y\relax
206             \edef\spath@temp{\noexpand\pgfpoint{\the\pgf@x}{\the\pgf@y}}%
207             \pgfoolet{\attribute}{\spath@temp}%
208         \fi
209     }
210 \endgroup
211 \let\spath@tmppath=\pgfutil@empty
212 \else
213     \pgfoothis.clone(\spath@newpath)%
214     \begingroup
215     \spath@newpath.scale path(,#2,#3)%
216     \endgroup
217     \let#1=\spath@newpath
218 \fi
219 }

```

Round all the corners on the path, ignoring any that are already set

```

220 \method round corners(#1,#2,#3) {%
221     \let\spath@newpath=#1\relax
222     \ifx\spath@newpath\relax
223         \let\spath@tmppath=\pgfutil@empty
224         \begingroup
225         \spath@round@init
226         \edef\spath@rndx{#2}%
227         \edef\spath@rndy{#3}%
228         \ifx\spath@rndy\pgfutil@empty
229             \let\spath@rndy\spath@rndx
230         \fi
231         \pgfoovalueof{path}%
232         \pgfoolet{path}{\spath@tmppath}%
233     \endgroup
234     \pgfooget{path}{\spath@tmppath}%
235     \pgfprocessround\spath@tmppath\spath@temppath
236     \pgfoolet{path}{\spath@temppath}%
237 \else

```

```

238 \pgfooget{path}{\spath@tmppath}%
239 \spath@newpath =new spath(\spath@tmppath)%
240 \begingroup
241 \spath@newpath.round corners(,#2)%
242 \endgroup
243 \let#1=\spath@newpath
244 \fi
245 }

```

This shrinks the path towards a particular point by a given number of points. It doesn't scale the path, rather it adds a number of points to each coordinate depending on which quadrant the point is in relative to the given point.

```

246 \method shrink path(#1,#2,#3,#4) {%
247 \let\spath@newpath=#1\relax
248 \ifx\spath@newpath\relax
249 \let\spath@tmppath=\pgfutil@empty
250 \begingroup
251 \spath@trx=#2\relax
252 \spath@try=#3\relax
253 \edef\spath@shrinkby{#4}%
254 \spath@shrink@init
255 \pgfoovalueof{path}%
256 \pgfoolet{path}{\spath@tmppath}%
257 \let\spath@tmppath=\pgfutil@empty
258 \tikz@clear@foreach
259 \foreach \attribute in {
260   initial point,
261   final point,
262   min bb,
263   max bb%,
264 } {
265   \pgfooget{\attribute}{\spath@temp}%
266   \ifx\spath@temp\pgfutil@empty
267   \else
268     \spath@temp
269     \pgfmathsetlength{\pgf@xa}{\pgf@x < \spath@trx ? \pgf@x + \spath@shrinkby : (\pgf@x > \s
270     \pgfmathsetlength{\pgf@ya}{\pgf@y < \spath@try ? \pgf@y + \spath@shrinkby : (\pgf@y > \s
271     \edef\spath@temp{\noexpand\pgfpoint{\the\pgf@xa}{\the\pgf@ya}}%
272     \pgfoolet{\attribute}{\spath@temp}%
273   \fi
274 }
275 \endgroup
276 \let\spath@tmppath=\pgfutil@empty
277 \else
278   \pgfoothis.clone(\spath@newpath)%
279   \begingroup
280   \spath@newpath.shrink path(,#2,#3)%
281   \endgroup
282   \let#1=\spath@newpath
283 \fi

```


284 }

This method prepares the path for a spirograph by applying an incremental rotation to each coordinate. If #1 is given then it should be a macro and it will be turned into an object with the transformed path, otherwise the current object is modified.

It works by redefining each soft path token to apply the given rotation to its arguments and then rewrite itself back to the path. Having done that, we have to adjust the various other attributes which are location-specific.

```

285 \method prepare spirograph(#1,#2,#3,#4) {%
286   \pgfoothis.prepare()%
287   \let\spath@newpath=#1\relax
288   \ifx\spath@newpath\relax
289     \let\spath@tmppath=\pgfutil@empty
290     \begingroup
291       \spath@trx=#2\relax
292       \spath@try=#3\relax
293       \def\spath@n{0}%
294       \pgfooget{length}{\spath@temp}%
295       \pgfmathsetmacro{\spath@gang}{#4/(\spath@temp-1)}%
296       \spath@spirograph@init
297       \pgfoovalueof{path}%
298       \pgfoolet{path}{\spath@tmppath}%
299       \let\spath@tmppath=\pgfutil@empty
300       \pgfooget{final point}{\spath@temp}%
301       \ifx\spath@temp\pgfutil@empty
302         \else
303           \spath@temp
304           \pgf@xa=\pgf@x\relax
305           \pgf@ya=\pgf@y\relax
306           \advance\pgf@xa by -\spath@trx\relax
307           \advance\pgf@ya by -\spath@try\relax
308           \pgfmathsetmacro{\spath@gcos}{cos(#4)}%
309           \pgfmathsetmacro{\spath@gsin}{sin(#4)}%
310           \pgfmathsetlength{\pgf@xb}{\spath@gcos * \pgf@xa - \spath@gsin * \pgf@ya}%
311           \pgfmathsetlength{\pgf@yb}{\spath@gsin * \pgf@xa + \spath@gcos * \pgf@ya}%
312           \advance\pgf@xb by \spath@trx\relax
313           \advance\pgf@yb by \spath@try\relax
314           \edef\spath@temp{\noexpand\pgfpoint{\the\pgf@xb}{\the\pgf@yb}}%
315           \pgfoolet{final point}{\spath@temp}%
316         \fi
317       \endgroup
318       \let\spath@tmppath=\pgfutil@empty
319     \else
320       \pgfoothis.clone(\spath@newpath)%
321       \begingroup
322       \spath@newpath.prepare spirograph(, #2, #3, #4)%
323       \endgroup
324       \let#1=\spath@newpath
325     \fi

```

326 }

This method actually makes a spirograph. The arguments are the point of rotation and the number of iterations.

```

327 \method spirograph(#1,#2,#3,#4) {
328   \pgfoothis.prepare()%
329   \let\spath@newpath=#1\relax
330   \ifx\spath@newpath\relax
331     \begingroup
332     \pgfmithsetmacro{\spath@gn}{360/#4}%
333     \pgfoothis.prepare spirograph(,#2,#3,\spath@gn)%
334     \pgfoothis.clone(\spath@tempa)%
335     \pgfmithsetmacro{\spath@gna}{2*\spath@gn}%
336     \pgfmithsetmacro{\spath@gnb}{360-\spath@gn}%
337     \tikz@clear@foreach
338     \foreach \k in {\spath@gn,\spath@gna,...,\spath@gnb} {%
339       \spath@tempa.rotate path(,\spath@gn)%
340       \pgfoothis.weld(,\spath@tempa)%
341     }%
342   \endgroup
343   \else
344     \pgfoothis.clone(\spath@newpath)%
345     \begingroup
346     \spath@newpath.spirograph(,#2,#3,#4)%
347   \endgroup
348   \let#1=\spath@newpath
349   \fi
350 }
```

This method rotates the path by the given angle. If #1 is given then it should be a macro and it will be turned into an object with the transformed path, otherwise the current object is modified.

It works by redefining each soft path token to apply the given rotation to its arguments and then rewrite itself back to the path. Having done that, we have to adjust the various other attributes which are location-specific.

```

351 \method rotate path(#1,#2) {%
352   \let\spath@newpath=#1\relax
353   \ifx\spath@newpath\relax
354     \let\spath@tmppath=\pgfutil@empty
355     \begingroup
356     \pgfmithsetmacro{\spath@gcos}{cos(#2)}%
357     \pgfmithsetmacro{\spath@gsin}{sin(#2)}%
358     \spath@rotate@init
359     \pgfoovalueof{path}%
360     \pgfoolet{path}{\spath@tmppath}%
361     \let\spath@tmppath=\pgfutil@empty
362     \pgfooget{initial point}{\spath@temp}%
363     \ifx\spath@temp\pgfutil@empty
364       \else
365         \spath@temp
```

```

366     \pgf@xa=\pgf@x\relax
367     \pgf@ya=\pgf@y\relax
368     \pgfmathsetlength{\pgf@xb}{\spath@gcoss * \pgf@xa - \spath@gsin * \pgf@ya}%
369     \pgfmathsetlength{\pgf@yb}{\spath@gsin * \pgf@xa + \spath@gcoss * \pgf@ya}%
370     \edef\spath@temp{\noexpand\pgfpoint{\the\pgf@xb}{\the\pgf@yb}}%
371     \pgfoolet{initial point}{\spath@temp}%
372   \fi
373   \pgfoolet{final point}{\spath@temp}%
374   \ifx\spath@temp\pgfutil@empty
375   \else
376     \spath@temp
377     \pgf@xa=\pgf@x\relax
378     \pgf@ya=\pgf@y\relax
379     \pgfmathsetlength{\pgf@xb}{\spath@gcoss * \pgf@xa - \spath@gsin * \pgf@ya}%
380     \pgfmathsetlength{\pgf@yb}{\spath@gsin * \pgf@xa + \spath@gcoss * \pgf@ya}%
381     \edef\spath@temp{\noexpand\pgfpoint{\the\pgf@xb}{\the\pgf@yb}}%
382     \pgfoolet{final point}{\spath@temp}%
383   \fi
384   \endgroup
385   \let\spath@tmppath=\pgfutil@empty
386   \else
387     \pgfoothis.clone(\spath@newpath)%
388     \begingroup
389     \spath@newpath.rotate path(, #2)%
390     \endgroup
391     \let#1=\spath@newpath
392   \fi
393 }

```

Get the length of the path (number of pieces)

```

394 \method length() {%
395   \pgfoolet{length}{\spath@temp}%
396   \ifx\spath@temp\pgfutil@empty
397     \setcounter{spath@length}{0}%
398   \begingroup
399     \spath@length@init
400     \pgfoovalueof{path}%
401   \endgroup
402   \edef\spath@temp{\the\value{spath@length}}%
403   \pgfoolet{length}{\spath@temp}%
404   \fi
405   \spath@temp
406 }

```

Get the real length of the path (number of components that draw)

```

407 \method real length() {%
408   \pgfoolet{real length}{\spath@temp}%
409   \ifx\spath@temp\pgfutil@empty
410     \setcounter{spath@length}{0}%
411   \begingroup
412     \spath@reallength@init

```

```

413     \pgfoovalueof{path}%
414     \endgroup
415     \edef\spath@temp{\the\value{spath@length}}%
416     \pgfoolet{real length}{\spath@temp}%
417     \fi
418     \spath@temp
419 }

Get the number of components of the path
420 \method number of components() {%
421     \pgfooget{number of components}{\spath@temp}%
422     \ifx\spath@temp\pgfutil@empty
423     \setcounter{spath@length}{0}%
424     \begingroup
425     \spath@components@init
426     \pgfoovalueof{path}%
427     \endgroup
428     \edef\spath@temp{\the\value{spath@length}}%
429     \pgfoolet{real length}{\spath@temp}%
430     \fi
431     \spath@temp
432 }

Get the initial point of the path
433 \method initial point() {%
434     \pgfooget{initial point}{\spath@temp}%
435     \ifx\spath@temp\pgfutil@empty
436     \begingroup
437     \spath@start@init
438     \pgfoovalueof{path}%
439     \edef\spath@temp{\noexpand\pgfpoint{\spath@sx}{\spath@sy}}%
440     \pgfoolet{initial point}{\spath@temp}%
441     \endgroup
442     \pgfooget{initial point}{\spath@temp}%
443     \fi
444     \spath@temp
445 }

Get the final point of the path
446 \method final point() {%
447     \pgfooget{final point}{\spath@temp}%
448     \ifx\spath@temp\pgfutil@empty
449     \begingroup
450     \spath@end@init
451     \pgfoovalueof{path}%
452     \edef\spath@temp{\noexpand\pgfpoint{\spath@ex}{\spath@ey}}%
453     \pgfoolet{final point}{\spath@temp}%
454     \endgroup
455     \pgfooget{final point}{\spath@temp}%
456     \fi
457     \spath@temp

```

```

458 }
    Get the bounding box of the path
459 \method bounding box() {%
460     \pgfooget{min bb}{\spath@temp}%
461     \ifx\spath@temp\pgfutil@empty
462     \else
463     \pgfooget{min bb}{\spath@temp}%
464     \fi
465     \ifx\spath@temp\pgfutil@empty
466     \begingroup
467     \spath@boundingbox@init
468     \pgfoovalueof{path}%
469     \edef\spath@temp{\noexpand\pgfpoint{\spath@mix}{\spath@miy}}%
470     \pgfoolet{min bb}{\spath@temp}%
471     \edef\spath@temp{\noexpand\pgfpoint{\spath@max}{\spath@may}}%
472     \pgfoolet{max bb}{\spath@temp}%
473     \endgroup
474     \fi
475 }
    Get the bounding box of the path: minimum corner
476 \method min bb() {%
477     \pgfooget{min bb}{\spath@temp}%
478     \ifx\spath@temp\pgfutil@empty
479     \pgfoothis.bounding box()%
480     \pgfooget{min bb}{\spath@temp}%
481     \fi
482     \spath@temp
483 }
    Get the bounding box of the path: maximum corner
484 \method max bb() {%
485     \pgfooget{max bb}{\spath@temp}%
486     \ifx\spath@temp\pgfutil@empty
487     \pgfoothis.bounding box()%
488     \pgfooget{max bb}{\spath@temp}%
489     \fi
490     \spath@temp
491 }
    Get the mid point of the path (according to the bounding box)
492 \method mid point() {%
493     \pgfoothis.min bb()%
494     \pgf@xa=.5\pgf@x
495     \pgf@ya=.5\pgf@y
496     \pgfoothis.max bb()%
497     \pgf@x=.5\pgf@x
498     \pgf@y=.5\pgf@y
499     \advance\pgf@xa by \pgf@x
500     \advance\pgf@ya by \pgf@y

```

```

501 \edef\spath@temp{\noexpand\pgfpoint{\the\pgf@xa}{\the\pgf@ya}}%
502 \spath@temp
503 }

```

Reverse the path. If #1 is given then it should be a macro and it will be an object with the reversed path, otherwise the current object is modified.

```

504 \method reverse path(#1) {%
505 \let\spath@newpath=#1\relax
506 \ifx\spath@newpath\relax
507 \let\spath@tmppath=\pgfutil@empty
508 \pgfoothis.prepare()%
509 \begingroup
510 \spath@reverse@init
511 \pgfoovalueof{path}%
512 \g@addbefore@macro\spath@tmppath\pgfsyssoftpath@movetotoken
513 \endgroup
514 \pgfoolet{path}{\spath@tmppath}%
515 \pgfooget{initial point}{\spath@temp}%
516 \pgfooget{final point}{\spath@tempa}%
517 \pgfoolet{final point}{\spath@temp}%
518 \pgfoolet{initial point}{\spath@tempa}%
519 \pgfooget{first action}{\spath@temp}%
520 \pgfooget{last action}{\spath@tempa}%
521 \pgfoolet{last action}{\spath@temp}%
522 \pgfoolet{first action}{\spath@tempa}%
523 \else
524 \pgfoothis.clone(\spath@newpath)%
525 \begingroup
526 \spath@newpath.reverse()%
527 \endgroup
528 \let#1=\spath@newpath
529 \fi
530 }

```

Prepare the path, filling out all the attributes. As many of the attributes require “walking” the path to figure them out, by doing them all in one go we can avoid too much duplication of effort.

```

531 \method prepare() {%
532 \pgfooget{prepared}{\spath@temp}%
533 \ifx\spath@temp\pgfutil@empty
534 \let\spath@tmppath=\pgfutil@empty
535 \begingroup
536 \setcounter{spath@reallength}{0}%
537 \setcounter{spath@components}{0}%
538 \setcounter{spath@length}{0}%
539 \spath@prepare@init
540 \let\spath@first=\pgfutil@empty
541 \pgfoovalueof{path}%
542 \edef\spath@temp{\the\value{spath@components}}%
543 \pgfoolet{number of components}{\spath@temp}%

```

```

544 \edef\spath@temp{\the\value{spath@length}}%
545 \pgfoolet{length}{\spath@temp}%
546 \edef\spath@temp{\the\value{spath@reallength}}%
547 \pgfoolet{real length}{\spath@temp}%
548 \edef\spath@temp{\noexpand\pgfpoint{\spath@sx}{\spath@sy}}%
549 \pgfoolet{initial point}{\spath@temp}%
550 \edef\spath@temp{\noexpand\pgfpoint{\spath@ex}{\spath@ey}}%
551 \pgfoolet{final point}{\spath@temp}%
552 \edef\spath@temp{\noexpand\pgfpoint{\spath@mix}{\spath@miy}}%
553 \pgfoolet{min bb}{\spath@temp}%
554 \edef\spath@temp{\noexpand\pgfpoint{\spath@max}{\spath@may}}%
555 \pgfoolet{max bb}{\spath@temp}%
556 \pgfoolet{first action}{\spath@first}%
557 \pgfoolet{last action}{\spath@last}%
558 \pgfoolet{prepared}{1}%
559 \endgroup
560 \fi
561 }

```

Now we have some other path manipulation methods that don't involve walking a path.

For tapering a path, it is important that the path has at least 3 components so that the first and last can be tapered. This method ensures that that is so by splitting the path if its real length is less than 3. If the real length is 1, it replaces the component by three. If the real length is 2, it splits both of them in half.

```

562 \method at least three() {%
563 \pgfoothis.prepare()%
564 \pgfooget{real length}{\spath@temp}%
565 \pgfooget{path}{\spath@tmppath}%
566 \ifnum\spath@temp=1\relax
567 \pgfooget{first action}{\spath@temp}%
568 \ifx\spath@temp\spath@lineto
569 \expandafter\spath@split@single@lineto\spath@tmppath\relax
570 \pgfoolet{path}{\spath@tmppath}%
571 \pgfooset{length}{4}%
572 \else
573 \expandafter\spath@split@single@curveto\spath@tmppath\relax
574 \pgfoolet{path}{\spath@tmppath}%
575 \pgfooset{length}{10}%
576 \fi
577 \pgfooset{real length}{3}%
578 \else
579 \ifnum\spath@temp=2\relax
580 \pgfooget{first action}{\spath@temp}%
581 \def\spath@tempa{0}%
582 \ifx\spath@temp\spath@lineto
583 \expandafter\spath@split@first@lineto\spath@tmppath\relax
584 \def\spath@tempa{3}%
585 \else
586 \expandafter\spath@split@first@curveto\spath@tmppath\relax

```

```

587     \def\spath@tempa{7}%
588     \fi
589     \pgfooget{last action}{\spath@temp}%
590     \ifx\spath@temp\spath@lineto
591         \expandafter\spath@split@second@lineto\spath@split@path@end\relax
592         \pgfmathsetmacro{\spath@tempa}{\spath@tempa + 2}
593     \else
594         \expandafter\spath@split@second@curveto\spath@split@path@end\relax
595         \pgfmathsetmacro{\spath@tempa}{\spath@tempa + 6}
596     \fi
597     \let\spath@tmppath=\spath@split@path@start
598     \ge@addto@macro\spath@tmppath\spath@split@path@end
599     \pgfoolet{path}{\spath@tmppath}%
600     \pgfoolet{length}{\spath@tempa}%
601     \pgfoolet{real length}{4}%
602     \fi
603 \fi
604 }

```

This is the tapering method. It actually only tapers the first (real) component of the path and throws away the rest. It should therefore be called after a path has been split (and after the path has been made to have at least 3 components).

```

605 \method taper out() {%
606     \pgfoothis.prepare()%
607     \pgfooget{first action}{\spath@temp}%
608     \pgfooget{path}{\spath@tmppath}%
609     \pgfooget{taper line width}{\taper@line@width}%
610     \ifx\taper@line@width\pgfutil@empty
611         \pgfmathsetmacro{\taper@line@width}{.5*\pgflinewidth}%
612     \fi
613     \ifx\spath@temp\spath@lineto
614         \expandafter\spath@taper@lineto@out\spath@tmppath\relax
615     \pgfoolet{path}{\spath@tapered@path}%
616     \else
617         \ifx\spath@temp\spath@curveto
618             \expandafter\spath@taper@curveto@out\spath@tmppath\relax
619         \pgfoolet{path}{\spath@tapered@path}%
620     \fi
621     \fi
622 }

```

The next few methods split a path by various conditions. The actual splitting routine is the same for all, the next few methods work by setting up the conditions necessary to determine the place to split the path.

The arguments are the same for each: macros to store the first and second parts of the path, and the number at which to split (negative counts from the end).

The first splits the path by number of tokens.

```

623 \method split path by length(#1,#2,#3) {%
624     \pgfoothis.prepare()%

```



```

625 \def\spath@test{\the\value{spath@length}}%
626 \pgfooget{length}{\spath@length}%
627 \pgfoothis.split(#1,#2,#3)%
628 }

```

This splits the path by number of drawing tokens.

```

629 \method split path by real length(#1,#2,#3) {%
630 \pgfoothis.prepare()%
631 \def\spath@test{\the\value{spath@reallength}}%
632 \pgfooget{real length}{\spath@length}%
633 \pgfoothis.split(#1,#2,#3)%
634 }

```

This splits the path by the number of components (movetos).

```

635 \method split path by component(#1,#2,#3) {%
636 \pgfoothis.prepare()%
637 \def\spath@test{\the\value{spath@components}}%
638 \pgfooget{number of components}{\spath@length}%
639 \pgfoothis.split(#1,#2,#3)%
640 }

```

This is the actual splitting code. It's another path-walker, counting as it goes and splitting when we get to the right place. Negative numbers mean working from the end. After the split, we have to ensure that the second part has a suitable moveto since all paths start with a move.

```

641 \method split(#1,#2,#3) {%
642 \pgfmathsetmacro{\spath@splitat}{#3 < 0 ? \spath@length + #3: #3}%
643 \pgfooget{path}{\spath@tmppath}%
644 \let\spath@tmppatha=\pgfutil@empty
645 \setcounter{spath@length}{0}%
646 \setcounter{spath@reallength}{0}%
647 \setcounter{spath@components}{-1}%
648 \expandafter\spath@gobble\spath@tmppath\relax
649 \pgfoonew #1 =new spath(\spath@tmppatha)%
650 \pgfoonew #2 =new spath(\spath@tmppath)%
651 #1.let(last action,\spath@last)%
652 #2.let(first action,\spath@first)%
653 \edef\spath@temp{\noexpand\pgfpoint{\spath@ex}{\spath@ey}}%
654 #1.let(final point,\spath@temp)%
655 \edef\spath@temp{\noexpand\pgfpoint{\spath@sx}{\spath@sy}}%
656 #2.let(initial point,\spath@temp)%
657 \pgfooget{first action}{\spath@temp}%
658 #1.let(first action,\spath@temp)%
659 \pgfooget{last action}{\spath@temp}%
660 #2.let(last action,\spath@temp)%
661 \pgfooget{initial point}{\spath@temp}%
662 #1.let(initial point,\spath@temp)%
663 \pgfooget{final point}{\spath@temp}%
664 #2.let(final point,\spath@temp)%
665 #1.let(length,\spath@splitat)%
666 \edef\spath@temp{\the\value{spath@reallength}}

```

```

667     #1.let(real length,\spath@temp)%
668     \pgfmathsetmacro{\spath@temp}{\pgfoovalueof{real length} - \spath@temp}
669     #2.let(real length,\spath@temp)%
670     \edef\spath@temp{\the\value{\spath@components}}
671     #1.let(number of components,\spath@temp)%
672     \pgfmathsetmacro{\spath@temp}{\pgfoovalueof{number of components} - \spath@temp}
673     #2.let(number of components,\spath@temp)%
674     \pgfooget{taper line width}{\spath@temp}%
675     #1.let(taper line width,\spath@temp)%
676     #2.let(taper line width,\spath@temp)%
677     \pgfmathsetmacro{\spath@splitat}{\spath@length - \spath@splitat}%
678     #2.let(length,\spath@splitat)%
679     #1.set(prepared,1)%
680     #2.set(prepared,1)%
681 }
682 % \end{macrocode}
683 % This reprocesses the path (it's another walker).
684 % The purpose here is that we can set new conditions (via TikZ or PGF options) which might affect
685 % An obvious example is to turn on the \Verb+rounded corners+ option.
686 % \begin{macrocode}
687 \method reprocess path() {%
688     \begingroup
689     \spath@reprocess@init
690     \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}%
691     \pgfoovalueof{path}%
692     \pgfsyssoftpath@getcurrentpath{\spath@tmppath}%
693     \pgfoolet{path}{\spath@tmppath}%
694     \endgroup
695     \let\spath@tmppath\pgfutil@empty
696 }

```

Now we come to a family of methods that actually use the path. The first sets the path as the current path. This means, for example, that the next TikZ command will start work on this path rather than an empty path.

```

697 \method set as current path() {%
698     \pgfooget{path}{\spath@tmppath}%
699     \pgfsyssoftpath@setcurrentpath{\spath@tmppath}%
700 }

```

This replaces the path in the object by the current soft path in the TikZ/PGF system. As this replaces the path, all other attributes should be thrown away.

```

701 \method get from current path() {%
702     \pgfsyssoftpath@getcurrentpath{\spath@tmppath}%
703     \pgfooset{path}{\spath@tmppath}%
704     \pgfoothis.set(prepared,0)%
705 }

```

This uses the path with the `\pgfusepath` command, so is a low-level access to the path.

```

706 \method use path(#1) {%
707     \pgfooget{path}{\spath@tmppath}%

```

```

708 \pgfoothis.min bb()%
709 \pgf@protocolsizes\pgf@x\pgf@y
710 \pgfoothis.max bb()%
711 \pgf@protocolsizes\pgf@x\pgf@y
712 \pgfsyssoftpath@setcurrentpath{\spath@tmppath}%
713 \pgfsyssoftpath@flushcurrentpath
714 \pgfusepath{#1}%
715 }

```

This uses the path via a TikZ-like command, so can use TikZ style options (passed as the argument).

```

716 \method use path with tikz(#1) {%
717   \path[#1] \pgfextra{%
718     \begingroup
719     \spath@reprocess@init
720     \pgfsyssoftpath@setcurrentpath{\pgfutil@empty}%
721     \pgfoovalueof{path}%
722     \endgroup};
723 }

```

This produces a string representation of the path (mainly for preloading paths)

```

724 \method to string() {%
725   \begingroup
726   \spath@string@init
727   \pgfoovalueof{path}%
728   \endgroup
729 }

```

Now we get some methods which act on two paths and join them in some fashion.

The first simply concatenates the paths. As each path starts with a `moveto`, the two paths are still somewhat separate.

```

730 \method concatenate(#1,#2) {%
731   \let\spath@newpath=#1\relax
732   \ifx\spath@newpath\relax
733     \let\spath@other=#2\relax
734     \ifx\spath@other\relax
735       \else
736         \spath@other.get(path,\spath@tmppath)%
737         \pgfooget{path}{\spath@tmppath}%
738         \ge@addto@macro\spath@tmppath\spath@tmppath
739         \pgfoolet{path}{\spath@tmppath}%
740         \spath@other.get(final point,\spath@temp)%
741         \pgfoolet{final point}{\spath@temp}%
742         \spath@other.get(last action,\spath@temp)%
743         \pgfoolet{last action}{\spath@temp}%
744         \pgfooget{min bb}{\spath@temp}%
745         \spath@temp
746         \pgf@xa=\pgf@x
747         \pgf@ya=\pgf@y
748         \spath@other.get(min bb,\spath@temp)%
749         \spath@temp

```

```

750 \pgfmathsetmacro{\spath@mix}{min(\pgf@x,\pgf@xa)}%
751 \pgfmathsetmacro{\spath@miy}{min(\pgf@y,\pgf@ya)}%
752 \edef\spath@temp{\noexpand\pgfpoint{\spath@mix}{\spath@miy}}%
753 \pgffoolet{min bb}{\spath@temp}%
754 \pgfooget{max bb}{\spath@temp}%
755 \spath@temp
756 \pgf@xa=\pgf@x
757 \pgf@ya=\pgf@y
758 \spath@other.get(max bb,\spath@temp)%
759 \spath@temp
760 \pgfmathsetmacro{\spath@max}{max(\pgf@x,\pgf@xa)}%
761 \pgfmathsetmacro{\spath@may}{max(\pgf@y,\pgf@ya)}%
762 \edef\spath@temp{\noexpand\pgfpoint{\spath@mix}{\spath@miy}}%
763 \pgffoolet{max bb}{\spath@temp}%
764 \tikz@clear@foreach
765 \foreach \attribute in {
766     length,
767     real length,
768     number of components%
769 } {
770     \spath@other.get(\attribute,\spath@temp)%
771     \pgfooget{\attribute}{\spath@tempa}%
772     \ifx\spath@temp\pgfutil@empty
773     \else
774     \ifx\spath@tempa\pgfutil@empty
775     \else
776     \pgfmathsetmacro{\spath@tempa}{\spath@temp + \spath@tempa}%
777     \pgffoolet{\attribute}{\spath@tempa}%
778     \fi
779     \fi
780 }%
781 \fi
782 \else
783 \pgfoothis.clone(\spath@newpath)%
784 \begingroup
785 \spath@newpath.concatenate(,#2)%
786 \endgroup
787 \let#1=\spath@newpath
788 \fi
789 }

```

This concatenates the paths but with a lineto instead of a moveto

```

790 \method concatenate with lineto(#1,#2) {%
791     \let\spath@newpath=#1\relax
792     \ifx\spath@newpath\relax
793     \let\spath@other=#2\relax
794     \ifx\spath@other\relax
795     \else
796     \spath@other.get(path,\spath@tmppath)%
797     \expandafter\spath@movetoline\spath@tmppath\relax

```

```

798 \pgfooget{path}{\spath@tmppatha}%
799 \ge@addto@macro\spath@tmppatha\spath@tmppath
800 \pgfoolet{path}{\spath@tmppatha}%
801 \spath@other.get(final point,\spath@temp)%
802 \pgfoolet{final point}{\spath@temp}%
803 \spath@other.get(last action,\spath@temp)%
804 \pgfoolet{last action}{\spath@temp}%
805 \pgfooget{min bb}{\spath@temp}%
806 \spath@temp
807 \pgf@xa=\pgf@x
808 \pgf@ya=\pgf@y
809 \spath@other.get(min bb,\spath@temp)%
810 \spath@temp
811 \pgfmathsetmacro{\spath@mix}{min(\pgf@x,\pgf@xa)}%
812 \pgfmathsetmacro{\spath@miy}{min(\pgf@y,\pgf@ya)}%
813 \edef\spath@temp{\noexpand\pgfpoint{\spath@mix}{\spath@miy}}%
814 \pgfoolet{min bb}{\spath@temp}%
815 \pgfooget{max bb}{\spath@temp}%
816 \spath@temp
817 \pgf@xa=\pgf@x
818 \pgf@ya=\pgf@y
819 \spath@other.get(max bb,\spath@temp)%
820 \spath@temp
821 \pgfmathsetmacro{\spath@max}{max(\pgf@x,\pgf@xa)}%
822 \pgfmathsetmacro{\spath@may}{max(\pgf@y,\pgf@ya)}%
823 \edef\spath@temp{\noexpand\pgfpoint{\spath@mix}{\spath@miy}}%
824 \pgfoolet{max bb}{\spath@temp}%
825 \tikz@clear@foreach
826 \foreach \attribute in {
827     length,
828     real length,
829     number of components%
830 } {
831     \spath@other.get(\attribute,\spath@temp)%
832     \pgfooget{\attribute}{\spath@tempa}%
833     \ifx\spath@temp\pgfutil@empty
834     \else
835     \ifx\spath@tempa\pgfutil@empty
836     \else
837     \pgfmathsetmacro{\spath@tempa}{\spath@temp + \spath@tempa}%
838     \pgfoolet{\attribute}{\spath@tempa}%
839     \fi
840     \fi
841 }%
842 \fi
843 \else
844 \pgfoothis.clone(\spath@newpath)%
845 \begingroup
846 \spath@newpath.concatenate with lineto(,#2)%
847 \endgroup

```

```

848     \let#1=\spath@newpath
849     \fi
850 }

This concatenates the paths but removes the moveto
851 \method concatenate without moveto(#1,#2) {%
852     \let\spath@newpath=#1\relax
853     \ifx\spath@newpath\relax
854         \let\spath@other=#2\relax
855         \ifx\spath@other\relax
856             \else
857                 \spath@other.get(path,\spath@tmppath)%
858                 \expandafter\spath@trimfirst\spath@tmppath\relax
859                 \pgfooget{path}{\spath@tmppath}%
860                 \ge@addto@macro\spath@tmppath\spath@tmppath
861                 \pgfoolet{path}{\spath@tmppath}%
862                 \spath@other.get(final point,\spath@temp)%
863                 \pgfoolet{final point}{\spath@temp}%
864                 \spath@other.get(last action,\spath@temp)%
865                 \pgfoolet{last action}{\spath@temp}%
866                 \pgfooget{min bb}{\spath@temp}%
867                 \spath@temp
868                 \pgf@xa=\pgf@x
869                 \pgf@ya=\pgf@y
870                 \spath@other.get(min bb,\spath@temp)%
871                 \spath@temp
872                 \pgfmathsetmacro{\spath@mix}{min(\pgf@x,\pgf@xa)}%
873                 \pgfmathsetmacro{\spath@miy}{min(\pgf@y,\pgf@ya)}%
874                 \edef\spath@temp{\noexpand\pgfpoint{\spath@mix}{\spath@miy}}%
875                 \pgfoolet{min bb}{\spath@temp}%
876                 \pgfooget{max bb}{\spath@temp}%
877                 \spath@temp
878                 \pgf@xa=\pgf@x
879                 \pgf@ya=\pgf@y
880                 \spath@other.get(max bb,\spath@temp)%
881                 \spath@temp
882                 \pgfmathsetmacro{\spath@max}{max(\pgf@x,\pgf@xa)}%
883                 \pgfmathsetmacro{\spath@may}{max(\pgf@y,\pgf@ya)}%
884                 \edef\spath@temp{\noexpand\pgfpoint{\spath@mix}{\spath@miy}}%
885                 \pgfoolet{max bb}{\spath@temp}%
886 \tikz@clear@foreach
887     \foreach \attribute in {
888         length,
889         real length,
890         number of components%
891     } {
892         \spath@other.get(\attribute,\spath@temp)%
893         \pgfooget{\attribute}{\spath@temp}%
894         \ifx\spath@temp\pgfutil@empty
895             \else

```

```

896     \ifx\spath@tempa\pgfutil@empty
897     \else
898     \pgfmithsetmacro{\spath@tempa}{\spath@temp + \spath@tempa}%
899     \pgffoolet{\attribute}{\spath@tempa}%
900     \fi
901     \fi
902   }%
903   \pgffooget{length}{\spath@tempa}%
904   \pgfmithsetmacro{\spath@tempa}{\spath@tempa - 1}%
905   \pgffoolet{length}{\spath@tempa}%
906   \fi
907 \else
908   \pgffoothis.clone(\spath@newpath)%
909   \begingroup
910   \spath@newpath.concatenate without moveto(,#2)%
911   \endgroup
912   \let#1=\spath@newpath
913 \fi
914 }

```

Welding is like concatenation except that the paths are brought together and the intervening moveto is removed

```

915 \method weld(#1,#2) {%
916   \let\spath@newpath=#1\relax
917   \ifx\spath@newpath\relax
918   \let\spath@other=#2\relax
919   \ifx\spath@other\relax
920   \else
921     \pgffoothis.final point()%
922     \pgf@xa=\pgf@x
923     \pgf@ya=\pgf@y
924     \spath@other.initial point()%
925     \advance\pgf@xa by -\pgf@x
926     \advance\pgf@ya by -\pgf@y
927     \spath@other.translate path(\spath@tempo,\the\pgf@xa,\the\pgf@ya)%
928 %
929     \spath@tempo.get(path,\spath@tmppath)%
930     \expandafter\spath@trimfirst\spath@tmppath\relax
931     \pgffooget{path}{\spath@tmppath}%
932     \ge@addto@macro\spath@tmppath\spath@tmppath
933     \pgffoolet{path}{\spath@tmppath}%
934 %
935     \spath@tempo.get(final point,\spath@temp)%
936     \pgffoolet{final point}{\spath@temp}%
937     \spath@tempo.get(last action,\spath@temp)%
938     \pgffoolet{last action}{\spath@temp}%
939     \pgffooget{min bb}{\spath@temp}%
940     \spath@temp
941     \pgf@xa=\pgf@x
942     \pgf@ya=\pgf@y

```

```

943     \spath@other.get(min bb,\spath@temp)%
944     \spath@temp
945     \pgfmathsetmacro{\spath@mix}{min(\pgf@x,\pgf@xa)}%
946     \pgfmathsetmacro{\spath@miy}{min(\pgf@y,\pgf@ya)}%
947     \edef\spath@temp{\noexpand\pgfpoint{\spath@mix}{\spath@miy}}%
948     \pgfoolet{min bb}{\spath@temp}%
949     \spath@temp
950     \pgf@xa=\pgf@x
951     \pgf@ya=\pgf@y
952     \spath@other.get(max bb,\spath@temp)%
953     \spath@temp
954     \pgfmathsetmacro{\spath@max}{max(\pgf@x,\pgf@xa)}%
955     \pgfmathsetmacro{\spath@may}{max(\pgf@y,\pgf@ya)}%
956     \edef\spath@temp{\noexpand\pgfpoint{\spath@mix}{\spath@miy}}%
957     \pgfoolet{max bb}{\spath@temp}%
958 \tikz@clear@foreach
959     \foreach \attribute in {
960         length,
961         real length,
962         number of components%
963     } {
964         \spath@tempo.get(\attribute,\spath@temp)%
965         \pgfoolet{\attribute}{\spath@tempa}%
966         \ifx\spath@temp\pgfutil@empty
967             \else
968             \ifx\spath@tempa\pgfutil@empty
969                 \else
970                 \pgfmathsetmacro{\spath@tempa}{\spath@temp + \spath@tempa}%
971                 \pgfoolet{\attribute}{\spath@tempa}%
972             \fi
973         \fi
974     }%
975 \fi
976 \else
977     \pgfthis.clone(\spath@newpath)%
978     \begingroup
979     \spath@newpath.weld(#2)%
980 \endgroup
981 \let#1=\spath@newpath
982 \fi
983 }

```

This library isn't great with closed paths, but we can certainly close up an existing path.

```

984 \method close() {%
985     \pgfoolet{path}{\spath@tmppath}%
986     \g@addto@macro\spath@tmppath{\pgfsyssoftpath@closepathtoken{Opt}{Opt}}%
987     \pgfoolet{path}{\spath@tmppath}%
988     \pgfoolet{length}{\spath@temp}%
989     \pgfmathtruncatemacro{\spath@temp}{\spath@temp + 1}%

```



```

990 \pgfoolet{length}{\spath@temp}%
991 \pgfooget{real length}{\spath@temp}%
992 \pgfmathtruncatemacro{\spath@temp}{\spath@temp + 1}%
993 \pgfoolet{real length}{\spath@temp}%
994 \pgfooget{number of components}{\spath@temp}%
995 \pgfmathtruncatemacro{\spath@temp}{\spath@temp + 1}%
996 \pgfoolet{number of components}{\spath@temp}%
997 \pgfooget{initial point}{\spath@temp}%
998 \pgfoolet{final point}{\spath@temp}%
999 \pgfooset{last action}{closepath}%
1000 \pgfooset{prepared}{0}%
1001 }

```

This method steps through the path and splits curvetos that might possibly self-intersect.

```

1002 \method split self intersecting pieces() {%
1003 \beginpgfgroup
1004 \let\spath@tmppath=\pgfutil@empty
1005 \spath@selfintersect@init
1006 \pgfoovalueof{path}%
1007 \pgfoolet{path}{\spath@tmppath}%
1008 \endpgfgroup
1009 }

```

Explode the path into an array of drawing components

```

1010 \method explode(#1) {%
1011 \pgfooget{path}{\spath@tmp}%
1012 \spatexplode#1\spath@tmp
1013 }

1014 \method at intersections(#1) {%
1015 \pgfooget{path}{\spath@tmppath}%
1016 #1.get(path,\spath@tmppath)%
1017 \ifx\spath@tmppath\pgfutil@empty
1018 \else
1019 \pgfoothis.get handle(\spath@firstpath)%
1020 \let\spath@secondpath=#1%
1021 \pgfintersectionofpaths{\pgfsetpath\spath@tmppath}{\pgfsetpath\spath@tmppath}%
1022 \ifnum\pgfintersectionsolutions>0\relax
1023 \pgfsyssoftpath@setcurrentpath\pgfutil@empty
1024 \tikz@clear@foreach
1025 \foreach \spath@k in {1,...,\pgfintersectionsolutions} {
1026 \pgfpointintersectionsolution{\spath@k}%
1027 \edef\spath@ix{\the\pgf@x}%
1028 \edef\spath@iy{\the\pgf@y}%
1029 \pgfoothis.initial point()
1030 \pgfmathtruncatemacro{\spath@first@nearstart}{(abs(\pgf@x - \spath@ix) + abs(\pgf@y - \spath@iy))}
1031 \pgfoothis.final point()
1032 \pgfmathtruncatemacro{\spath@first@nearend}{(abs(\pgf@x - \spath@ix) + abs(\pgf@y - \spath@iy))}
1033 \pgfmathtruncatemacro{\spath@first@nearends}{\spath@first@nearstart || \spath@first@nearend}
1034 \spath@secondpath.initial point()

```

```

1035     \pgfmathtruncatemacro{\spath@second@nearstart}{(abs(\pgf@x - \spath@ix) + abs(\pgf@y - \spa
1036     \spath@secondpath.final point()
1037     \pgfmathtruncatemacro{\spath@second@nearend}{(abs(\pgf@x - \spath@ix) + abs(\pgf@y - \spath
1038     \pgfmathtruncatemacro{\spath@second@nearends}{\spath@second@nearstart || \spath@second@near
1039     \pgfmathtruncatemacro{\spath@nearends}{\spath@first@nearends || \spath@second@nearends}%
1040     \spath@execute@at@intersections
1041 }
1042 \fi
1043 \fi
1044 }

```

Here endeth the `spath` object.

```

1045 }

```

The `spath` component is designed to be part of an array of `spath` objects. Each `spath` component consists of an `spath` object and a link to a previous and a next `spath` component (either of which could be empty).

```

1046 \pgfoclass{spath component}{

```

Our attributes are our `spath`, and the next and previous components.

```

1047 \attribute path;
1048 \attribute next component;
1049 \attribute previous component;

```

This is the creator. If called with an argument, that is assumed to be the previous component in the array.

```

1050 \method spath component(#1) {%
1051     \def\spath@temp{#1}%
1052     \ifx\spath@temp\pgfutil@empty
1053     \else
1054     \pgfoolet{previous component}{#1}%
1055     \fi
1056 }

```

Generic attribute handling, see `spath` class for details

```

1057 \method value(#1) {%
1058     \pgfoovalueof{#1}%
1059 }
1060 \method set(#1,#2) {%
1061     \pgfooset{#1}{#2}%
1062 }
1063 \method let(#1,#2) {%
1064     \pgfoolet{#1}{#2}%
1065 }
1066 \method get(#1,#2) {%
1067     \pgfooget{#1}{#2}%
1068 }
1069 \method show(#1) {%
1070     \pgfooget{#1}{\@temp}%
1071     \show\@temp
1072 }

```

This says what we are

```
1073 \method what am I() {%
1074     \message{I am an spath component object}%
1075 }
```

This sets the `path` attribute (as an instance of the `spath` class) from a saved path.

```
1076 \method set path(#1) {%
1077     \pgfoonew \spath@tempa =new spath(#1)%
1078     \pgfoolet{path}{\spath@tempa}%
1079 }
```

This recurses along the array applying an action to every object. Specifically, it applies the action to the `path` attribute and then calls the next component with the same arguments.

```
1080 \method apply to paths(#1,#2) {%
1081     \pgfooget{path}{\spath@tmppath}%
1082     \ifx\spath@tmppath\pgfutil@empty
1083     \else
1084         \spath@tmppath.#1(#2)%
1085     \fi
1086     \pgfooget{next component}{\spath@temp}%
1087     \let\spath@next=\pgfutil@empty
1088     \ifx\spath@temp\pgfutil@empty
1089     \else
1090         \def\spath@next{\spath@temp.apply to paths(#1,{#2})}%
1091     \fi
1092     \spath@next
1093 }
1094 \method apply to previous paths(#1,#2) {%
1095     \pgfooget{path}{\spath@tmppath}%
1096     \ifx\spath@tmppath\pgfutil@empty
1097     \else
1098         \spath@tmppath.#1(#2)%
1099     \fi
1100     \pgfooget{previous component}{\spath@temp}%
1101     \let\spath@next=\pgfutil@empty
1102     \ifx\spath@temp\pgfutil@empty
1103     \else
1104         \def\spath@next{\spath@temp.apply to previous paths(#1,{#2})}%
1105     \fi
1106     \spath@next%
1107 }
1108 \method apply for paths(#1) {%
1109     #1%
1110     \pgfooget{next component}{\spath@temp}%
1111     \let\spath@next=\pgfutil@empty
1112     \ifx\spath@temp\pgfutil@empty
1113     \else
1114         \def\spath@next{\spath@temp.apply for paths({#1})}%
1115     \fi
```

```

1116     \spath@next%
1117 }
1118 \method apply for previous paths(#1) {%
1119     #1%
1120     \pgfooget{previous component}{\spath@temp}%
1121     \let\spath@next=\pgfutil@empty
1122     \ifx\spath@temp\pgfutil@empty
1123     \else
1124     \def\spath@next{\spath@temp.apply for previous paths({#1})}%
1125     \fi
1126     \spath@next%
1127 }

```

We look for intersections between paths. If #1 is given, we use that as one of the `spath` objects to intersect with and look for intersections with that starting with us and proceeding to all following components. If #1 is not given, we take our `spath` object as the one to use and look for intersections for all subsequent components. We carry the previous and next components around with us in case the intersection happens to be too close to an endpoint and we want to use the other components as well.

```

1128 \method at intersections(#1,#2,#3) {%
1129     \let\spath@next=\pgfutil@empty
1130     \let\spath@tmp#1\relax%
1131     \let\spath@secondpath@prev#2\relax
1132     \let\spath@secondpath@next#3\relax
1133     \ifx\spath@tmp\relax
1134     \pgfooget{path}{\spath@tmp}%
1135     \pgfooget{previous component}{\spath@temp}%
1136     \ifx\spath@temp\pgfutil@empty
1137     \else
1138     \spath@temp.get(path,\spath@secondpath@prev)%
1139     \fi
1140     \pgfooget{next component}{\spath@temp}%
1141     \ifx\spath@temp\pgfutil@empty
1142     \else
1143     \spath@temp.get(path,\spath@secondpath@next)%
1144     \def\spath@next{%
1145         \begingroup
1146         \spath@temp.at intersections(\spath@tmp,\spath@secondpath@prev,\spath@secondpath@next)%
1147         \endgroup
1148         \spath@temp.at intersections(,,)%
1149     }
1150     \fi
1151     \else
1152     \begingroup
1153     \pgfooget{previous component}{\spath@temp}%
1154     \ifx\spath@temp\pgfutil@empty
1155     \else
1156     \spath@temp.get(path,\spath@firstpath@prev)%
1157     \fi

```

```

1158 \pgfooget{next component}{\spath@temp}%
1159 \ifx\spath@temp\pgfutil@empty
1160 \else
1161 \spath@temp.get(path,\spath@firstpath@next)%
1162 \fi
1163 \pgfooget{path}{\spath@temp}%
1164 \ifx\spath@temp\pgfutil@empty
1165 \else
1166 \spath@temp.at intersections(\spath@tmp)%
1167 \fi
1168 \endgroup
1169 \pgfooget{next component}{\spath@temp}%
1170 \ifx\spath@temp\pgfutil@empty
1171 \else
1172 \def\spath@next{%
1173 \spath@temp.at intersections(\spath@tmp,\spath@secondpath@prev,\spath@secondpath@next)%
1174 }
1175 \fi
1176 \fi
1177 \spath@next
1178 }

```

That's all, folks. At least for the `spath` component

```
1179 }
```

The reason that the above array structure was created was to be able to work on a soft path “component by component”. The following macros take a soft path and split it at movetos.

```

1180 \def\spathsplitlet#1#2{%
1181 \ifx#1\relax
1182 \pgfoonew #1 =new spath component({})%
1183 \else
1184 \message{\string#1\space already defined}
1185 \fi
1186 \let\spath@this@component=#1\relax
1187 \expandafter\spath@split@#2\relax
1188 }

1189 \def\spath@split@pgfsyssoftpath@movetotoken#1\relax{%
1190 \spath@split@@#1\pgfsyssoftpath@movetotoken\relax
1191 }

1192 \def\spath@split@@#1\pgfsyssoftpath@movetotoken#2\relax{%
1193 \def\spath@tmppath{\pgfsyssoftpath@movetotoken#1}%
1194 \spath@this@component.set path(\spath@tmppath)%
1195 \def\spath@tmppath{#2}%
1196 \ifx\spath@tmppath\pgfutil@empty
1197 \else
1198 \pgfoonew \spath@next@component =new spath component(\spath@this@component)%
1199 \spath@this@component.let(next component,\spath@next@component)%
1200 \let\spath@this@component=\spath@next@component
1201 \spath@split@@#2\relax

```

```

1202 \fi
1203 }

```

The following macros take a soft path and split it into drawing components.

```

1204 \def\spathexplode#1#2{%
1205   \ifdefined#1
1206   \else
1207   \pgfoonew #1 =new spath component({})%
1208   \fi
1209   \let\spath@this@component=#1\relax
1210   \expandafter\spath@explode@#2\relax
1211 }

1212 \def\spath@explode@#1{%
1213   \let\spath@explode@next=\pgfutil@empty
1214   \ifx#1\pgfsyssoftpath@movetotoken
1215   \let\spath@explode@next=\spath@explode@savexy
1216   \else
1217   \ifx#1\pgfsyssoftpath@linetotoken
1218   \let\spath@explode@next=\spath@explode@lineto
1219   \else
1220   \ifx#1\pgfsyssoftpath@curvetosupportatoken
1221   \let\spath@explode@next=\spath@explode@curveto
1222   \else
1223   \ifx#1\pgfsyssoftpath@closepathtoken
1224   \let\spath@explode@next=\spath@explode@lineto
1225   \else
1226   \ifx#1\pgfsyssoftpath@recttoken
1227   \let\spath@explode@next=\spath@explode@rect
1228   \fi
1229   \fi
1230   \fi
1231   \fi
1232   \fi
1233   \spath@explode@next
1234 }

1235 \def\spath@explode@savexy#1#2{%
1236   \def\spath@ex{#1}%
1237   \def\spath@ey{#2}%
1238   \spath@explode@%
1239 }

1240 \def\spath@explode@lineto#1#2{%
1241   \edef\spath@tmppath{\noexpand\pgfsyssoftpath@movetotoken{\spath@ex}{\spath@ey}\noexpand\pgfsy
1242   \spath@this@component.set path(\spath@tmppath)%
1243   \pgfoonew \spath@next@component =new spath component(\spath@this@component)%
1244   \spath@this@component.let(next component,\spath@next@component)%
1245   \let\spath@this@component=\spath@next@component
1246   \def\spath@ex{#1}%
1247   \def\spath@ey{#2}%
1248   \spath@explode@%

```

```

1249 }

1250 \def\spath@explode@curveto#1#2\pgfsyssoftpath@curvetosupportbtoken#3#4\pgfsyssoftpath@curvetoto
1251 \edef\spath@tmppath{\noexpand\pgfsyssoftpath@movetotoken{\spath@ex}{\spath@ey}\noexpand\pgfsy
1252 \spath@this@component.set path(\spath@tmppath)%
1253 \pgfoonew \spath@next@component =new spath component(\spath@this@component)%
1254 \spath@this@component.let(next component,\spath@next@component)%
1255 \let\spath@this@component=\spath@next@component
1256 \def\spath@ex{#5}%
1257 \def\spath@ey{#6}%
1258 \spath@explode@%
1259 }

1260 % These are all our helper macros.
1261 % The first batch are for defining the ‘walker’ methods of the \Verb+spath+ class.
1262 % For each of the methods we have to define some initialiser code which redefines the soft path
1263 % As the ‘something appropriate’ is often independent of the actual token macro, we set up a
1264 % These set the default actions which can be overridden afterwards.
1265 %
1266 % We start with translation.
1267 % Each soft path token translates its arguments and then appends itself, with the translated ar
1268 % \begin{macrocode}
1269 \def\spath@define@translate#1\relax{%
1270 \expandafter\gdef\csname spath@tr@#1\endcsname##1##2{%
1271 \pgf@xa=##1\relax
1272 \pgf@ya=##2\relax
1273 \advance\pgf@xa by \spath@trx
1274 \advance\pgf@ya by \spath@try
1275 \edef\spath@tmp{\expandafter\noexpand\csname pgfsyssoftpath@#1token\endcsname{\the\pgf@xa}{\
1276 \ge@addto@macro\spath@tmppath\spath@tmp
1277 }
1278 }

Apply a transformation to the path
1279 \def\spath@define@transform#1\relax{%
1280 \expandafter\gdef\csname spath@trans@#1\endcsname##1##2{%
1281 \pgf@xa=##1\relax
1282 \pgf@ya=##2\relax
1283 \pgf@process{\pgfpointransformed{\pgfqpoint{\pgf@xa}{\pgf@ya}}}%
1284 \edef\spath@tmp{\expandafter\noexpand\csname pgfsyssoftpath@#1token\endcsname{\the\pgf@x}{\
1285 \ge@addto@macro\spath@tmppath\spath@tmp
1286 }
1287 }

Scale all coordinates
1288 \def\spath@define@scale#1\relax{%
1289 \expandafter\gdef\csname spath@sc@#1\endcsname##1##2{%
1290 \pgf@xa=##1\relax
1291 \pgf@ya=##2\relax
1292 \pgf@xa=\spath@scx\pgf@xa\relax
1293 \pgf@ya=\spath@scy\pgf@ya\relax
1294 \edef\spath@tmp{\expandafter\noexpand\csname pgfsyssoftpath@#1token\endcsname{\the\pgf@xa}{\

```

```

1295     \ge@addto@macro\spath@tmppath\spath@tmp
1296   }
1297 }

```

Define the rounding macros, most do nothing

```

1298 \def\spath@define@round#1\relax{%
1299   \expandafter\gdef\csname spath@rnd@#1\endcsname##1##2{%
1300     \edef\spath@tmp{\expandafter\noexpand\csname pgfsyssoftpath@#1token\endcsname{##1}{##2}}
1301     \ge@addto@macro\spath@tmppath\spath@tmp
1302   }
1303 }

```

Each soft path token shrinks its arguments towards the given point and then appends itself to the temporary path.

```

1304 \def\spath@define@shrink#1\relax{%
1305   \expandafter\gdef\csname spath@sh@#1\endcsname##1##2{%
1306     \pgfmathsetlength{\pgf@xa}{##1 < \spath@trx ? ##1 + \spath@shrinkby : (##1 > \spath@trx ? #
1307     \pgfmathsetlength{\pgf@ya}{##2 < \spath@try ? ##2 + \spath@shrinkby : (##2 > \spath@try ? #
1308     \edef\spath@tmp{\expandafter\noexpand\csname pgfsyssoftpath@#1token\endcsname{\the\pgf@xa}{
1309     \ge@addto@macro\spath@tmppath\spath@tmp
1310   }
1311 }

```

Each soft path token rotates its arguments about the given origin and then appends itself to the temporary path. Then it increments the rotation counter.

```

1312 \def\spath@define@spirograph#1\relax{%
1313   \expandafter\gdef\csname spath@gch@#1\endcsname##1##2{%
1314     \pgf@xa=##1\relax
1315     \pgf@ya=##2\relax
1316     \advance\pgf@xa by -\spath@trx
1317     \advance\pgf@ya by -\spath@try
1318     \pgfmathsetmacro\spath@gcos{cos(\spath@n * \spath@gang)}
1319     \pgfmathsetmacro\spath@gsin{sin(\spath@n * \spath@gang)}
1320     \pgfmathsetmacro\spath@n{\spath@n + 1}%
1321     \global\let\spath@n=\spath@n
1322     \pgfmathsetlength\pgf@xb{\spath@gcos * \pgf@xa - \spath@gsin * \pgf@ya}
1323     \pgfmathsetlength\pgf@yb{\spath@gsin * \pgf@xa + \spath@gcos * \pgf@ya}
1324     \advance\pgf@xb by \spath@trx
1325     \advance\pgf@yb by \spath@try
1326     \edef\spath@tmp{\expandafter\noexpand\csname pgfsyssoftpath@#1token\endcsname{\the\pgf@xb}{
1327     \ge@addto@macro\spath@tmppath\spath@tmp
1328   }
1329 }

```

Each soft path token rotates its arguments and then appends itself to the temporary path.

```

1330 \def\spath@define@rotate#1\relax{%
1331   \expandafter\gdef\csname spath@rot@#1\endcsname##1##2{%
1332     \pgf@xa=##1\relax
1333     \pgf@ya=##2\relax
1334     \pgfmathsetlength\pgf@xb{\spath@gcos * \pgf@xa - \spath@gsin * \pgf@ya}

```



```

1335 \pgfmathsetlength\pgf@yb{\spath@gsin * \pgf@xa + \spath@gcos * \pgf@ya}
1336 \edef\spath@tmp{\expandafter\noexpand\csname pgfsyssoftpath@#1token\endcsname{\the\pgf@xb}{\the\pgf@yb}}
1337 \ge@addto@macro\spath@tmppath\spath@tmp
1338 }
1339 }

```

This is for the length, we simply increment the length counter.

```

1340 \def\spath@define@length#1\relax{%
1341 \expandafter\gdef\csname spath@len@#1\endcsname##1##2{%
1342 \stepcounter{spath@length}
1343 }
1344 }

```

This is for the number of components, by default we do nothing. The special code for the `moveto` will be set up later.

```

1345 \def\spath@define@components#1\relax{%
1346 \expandafter\gdef\csname spath@comp@#1\endcsname##1##2{%
1347 }
1348 }

```

This is for the real length, by default we do nothing. The special code for the `lineto` and `curveto` will be set up later.

```

1349 \def\spath@define@reallength#1\relax{%
1350 \expandafter\gdef\csname spath@rlen@#1\endcsname##1##2{%
1351 }
1352 }

```

This is for reversing the path. By default, we place our coordinates *before* the current token. The special code for the `curveto` will be set up later.

```

1353 \def\spath@define@reverse#1\relax{%
1354 \expandafter\gdef\csname spath@rev@#1\endcsname##1##2{%
1355 \edef\spath@tmp{{##1}{##2}}\expandafter\noexpand\csname pgfsyssoftpath@#1token\endcsname{\the\spath@tmp}
1356 \ge@addbefore@macro\spath@tmppath\spath@tmp
1357 }
1358 }

```

This is for recording the start of a path, so we save our coordinates and then reinitialise so that the rest of the path does nothing.

```

1359 \def\spath@define@start#1\relax{%
1360 \expandafter\gdef\csname spath@start@#1\endcsname##1##2{%
1361 \edef\spath@sx{##1}
1362 \edef\spath@sy{##2}
1363 \spath@start@reinit
1364 }
1365 }

```

This is for recording the end of a path, so we save our coordinates. Each component overwrites what the previous one saved so we're left with the final coordinates.

```

1366 \def\spath@define@end#1\relax{%
1367 \expandafter\gdef\csname spath@end@#1\endcsname##1##2{%
1368 \edef\spath@ex{##1}
1369 \edef\spath@ey{##2}

```

```

1370 }
1371 }

```

This is for splitting a soft path into components. The default action is to add
ourselves to the current temporary path.

```

1372 \def\spath@define@array#1\relax{%
1373   \expandafter\gdef\csname spath@array@#1\endcsname##1##2{%
1374     \edef\spath@tmp{\expandafter\noexpand\csname          pgfsyssoftpath@#1token\endcsname{##1}{##2}}%
1375     \ge@addto@macro\spath@tmppath\spath@tmp
1376   }
1377 }

```

This is for reprocessing the path.

```

1378 \def\spath@define@reprocess#1\relax{%
1379   \expandafter\gdef\csname spath@rep@#1\endcsname##1##2{%
1380     \csname pgfpath#1\endcsname{\pgfpoint{##1}{##2}}%
1381   }%
1382 }

```

This is for splitting self-intersecting pieces. Most pieces just record the endpoints.
We only actually split `curvetos`.

```

1383 \def\spath@define@selfintersect#1{%
1384   \expandafter\gdef\csname spath@selfintersect@#1\endcsname##1##2{%
1385     \edef\spath@ex{##1}%
1386     \edef\spath@ey{##2}%
1387     \edef\spath@tmp{\expandafter\noexpand\csname pgfsyssoftpath@#1token\endcsname{##1}{##2}}%
1388     \ge@addto@macro\spath@tmppath\spath@tmp
1389   }
1390 }

```

This is for computing the bounding box

```

1391 \def\spath@define@boundingbox#1{%
1392   \expandafter\gdef\csname spath@boundingbox@#1\endcsname##1##2{%
1393     \pgfmathsetmacro\spath@mix{\min(\spath@mix,##1)}%
1394     \pgfmathsetmacro\spath@miy{\min(\spath@miy,##2)}%
1395     \pgfmathsetmacro\spath@max{\max(\spath@max,##1)}%
1396     \pgfmathsetmacro\spath@may{\max(\spath@may,##2)}%
1397   }%
1398 }%

```

This is for converting the path into a string

```

1399 \def\spath@define@string#1\relax{%
1400   \expandafter\gdef\csname spath@str@#1\endcsname##1##2{%
1401     \pgfmathsetmacro\spath@sx{##1/1cm}%
1402     \pgfmathsetmacro\spath@sy{##2/1cm}%
1403     \expandafter\string\csname pgfpath#1\endcsname\{\string\pgfpointxy\{\spath@sx\}\{\spath@sy\}
1404   }%
1405 }

```

The next part of the code sets up the initialiser code for each of the routines. We
start with empty initialiser code.

```

1406 \let\spath@spirograph@init=\pgfutil@empty

```

```

1407 \let\spath@rotate@init=\pgfutil@empty
1408 \let\spath@round@init=\pgfutil@empty
1409 \let\spath@translate@init=\pgfutil@empty
1410 \let\spath@transform@init=\pgfutil@empty
1411 \let\spath@scale@init=\pgfutil@empty
1412 \let\spath@shrink@init=\pgfutil@empty
1413 \let\spath@length@init=\pgfutil@empty
1414 \let\spath@components@init=\pgfutil@empty
1415 \let\spath@reallength@init=\pgfutil@empty
1416 \let\spath@reverse@init=\pgfutil@empty
1417 \let\spath@start@init=\pgfutil@empty
1418 \let\spath@start@reinit=\pgfutil@empty
1419 \let\spath@end@init=\pgfutil@empty
1420 \let\spath@array@init=\pgfutil@empty
1421 \let\spath@prepare@init=\pgfutil@empty
1422 \let\spath@reprocess@init=\pgfutil@empty
1423 \let\spath@string@init=\pgfutil@empty
1424 \let\spath@selfintersect@init=\pgfutil@empty
1425 \def\spath@boundingbox@init{%
1426   \def\spath@mix{16000pt}%
1427   \def\spath@miy{16000pt}%
1428   \def\spath@max{-16000pt}%
1429   \def\spath@may{-16000pt}%
1430 }

```

We now loop over the possible soft path tokens and define the different types of action each will expand to in the different circumstances. These are the default actions, as defined by the above macros. Specific actions can be redefined afterwards.

```

1431 \tikz@clear@foreach
1432 \foreach \spath@cpt in {
1433   moveto,
1434   lineto,
1435   curvetosupporta,
1436   curvetosupportb,
1437   curveto,
1438   rectcorner,
1439   rectsize,
1440   closepath,
1441   specialround%
1442 } {

```

Save token names for comparison

```

1443 \expandafter\xdef\csname spath@\spath@cpt\endcsname{\expandafter\string\csname pgfsyssoftpath@

```

Define the translation macros.

```

1444 \expandafter\spath@define@translate\spath@cpt\relax

```

Add the “redefinition” code to the initialiser

```

1445 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1446

```

```

1447 \ge@addto@macro\spath@translate@init\spath@tmp
    Define the transformation macros.
1448 \expandafter\spath@define@transform\spath@cpt\relax
    Add the “redefinition” code to the initialiser
1449 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1450
1451 \ge@addto@macro\spath@transform@init\spath@tmp
    Define the scaling macros.
1452 \expandafter\spath@define@scale\spath@cpt\relax
    Add the “redefinition” code to the initialiser
1453 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1454
1455 \ge@addto@macro\spath@scale@init\spath@tmp
    Define the rounding macros.
1456 \expandafter\spath@define@round\spath@cpt\relax
    Add the “redefinition” code to the initialiser
1457 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1458
1459 \ge@addto@macro\spath@round@init\spath@tmp
    Define the shrinking macros.
1460 \expandafter\spath@define@shrink\spath@cpt\relax
    Add the “redefinition” code to the initialiser
1461 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1462
1463 \ge@addto@macro\spath@shrink@init\spath@tmp
    Define the spirograph macros.
1464 \expandafter\spath@define@spirograph\spath@cpt\relax
    Add the “redefinition” code to the initialiser
1465 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1466
1467 \ge@addto@macro\spath@spirograph@init\spath@tmp
    Define the rotation macros.
1468 \expandafter\spath@define@rotate\spath@cpt\relax
    Add the “redefinition” code to the initialiser
1469 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1470
1471 \ge@addto@macro\spath@rotate@init\spath@tmp
    Now do the same for counting the total length.
1472 \expandafter\spath@define@length\spath@cpt\relax
1473
1474 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1475
1476 \ge@addto@macro\spath@length@init\spath@tmp

```

Now do the same for counting the components.

```
1477 \expandafter\spath@define@components\spath@cpt\relax
1478
1479 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1480
1481 \ge@addto@macro\spath@components@init\spath@tmp
```

Now do the same for counting the real length.

```
1482 \expandafter\spath@define@reallength\spath@cpt\relax
1483
1484 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1485
1486 \ge@addto@macro\spath@reallength@init\spath@tmp
```

This is for reversing a path.

```
1487 \expandafter\spath@define@reverse\spath@cpt\relax
1488
1489 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1490
1491 \ge@addto@macro\spath@reverse@init\spath@tmp
```

This is for the initial coordinates

```
1492 \expandafter\spath@define@start\spath@cpt\relax
1493
1494 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1495
1496 \ge@addto@macro\spath@start@init\spath@tmp
```

Once we have the initial coordinates, we reinitialise our path tokens so that they just gobble their code.

```
1497 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1498
1499 \ge@addto@macro\spath@start@reinit\spath@tmp
```

This is for the final coordinates.

```
1500 \expandafter\spath@define@end\spath@cpt\relax
1501
1502 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1503
1504 \ge@addto@macro\spath@end@init\spath@tmp
```

Split in to array

```
1505 \expandafter\spath@define@array\spath@cpt\relax
1506
1507 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1508
1509 \ge@addto@macro\spath@array@init\spath@tmp
```

Prepare a path, figuring out all the data. Actions are too complicated to specify a template so just create initialisation code

```
1510 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1511
1512 \ge@addto@macro\spath@prepare@init\spath@tmp
```

Reprocess a path

```
1513 \expandafter\spath@define@reprocess\spath@cpt\relax
1514
1515 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1516
1517 \ge@addto@macro\spath@reprocess@init\spath@tmp
```

Split self-intersecting pieces

```
1518 \expandafter\spath@define@selfintersect\expandafter{\spath@cpt}
1519
1520 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1521
1522 \ge@addto@macro\spath@selfintersect@init\spath@tmp
```

Bounding box

```
1523 \expandafter\spath@define@boundingbox\expandafter{\spath@cpt}
1524
1525 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1526
1527 \ge@addto@macro\spath@boundingbox@init\spath@tmp
```

To string

```
1528 \expandafter\spath@define@string\spath@cpt\relax
1529
1530 \edef\spath@tmp{\noexpand\let\expandafter\noexpand\csname pgfsyssoftpath@\spath@cpt token\endcs
1531
1532 \ge@addto@macro\spath@string@init\spath@tmp
```

Phew!

```
1533 }
```

Now we add the corrections for the above: where an action deviates from the “default” we need to redefine it.

Correction for counting components: count movetos

```
1534 \def\spath@comp@moveto#1#2{%
1535   \stepcounter{spath@length}%
1536 }
```

Correction for counting real length: count linetos and curvetos

```
1537 \def\spath@rlen@lineto#1#2{%
1538   \stepcounter{spath@length}%
1539 }
1540 \def\spath@rlen@curveto#1#2{%
1541   \stepcounter{spath@length}%
1542 }
```

Correction for reversing initial moveto

```
1543 \def\spath@rev@moveto#1#2{
1544   \ifx\spath@tmppath\pgfutil@empty
1545     \edef\spath@tmp{{#1}{#2}}%
1546   \else
1547     \edef\spath@tmp{{#1}{#2}\noexpand\pgfsyssoftpath@movetotoken}%
1548   \fi
1549 }
```

```

1548 \fi
1549 \ge@addbefore@macro\spath@tmppath\spath@tmp
1550 }

Correction for reversing curvetos
1551 \def\spath@rev@curvetosupporta#1#2{%
1552 \edef\spath@tmp{{#1}-{#2}\noexpand\pgfsyssoftpath@curvetotoken}%
1553 \ge@addbefore@macro\spath@tmppath\spath@tmp
1554 }
1555 \def\spath@rev@curvetosupportb#1#2{%
1556 \edef\spath@tmp{{#1}-{#2}\noexpand\pgfsyssoftpath@curvetosupportbtoken}%
1557 \ge@addbefore@macro\spath@tmppath\spath@tmp
1558 }
1559 \def\spath@rev@curveto#1#2{%
1560 \edef\spath@tmp{{#1}-{#2}\noexpand\pgfsyssoftpath@curvetosupportatoken}%
1561 \ge@addbefore@macro\spath@tmppath\spath@tmp
1562 }

Correction for reversing closepaths
1563 \def\spath@rev@closepath#1#2{%
1564 \edef\spath@tmp{\noexpand\pgfsyssoftpath@closepathtoken{#1}-{#2}}%
1565 \ge@addto@macro\spath@tmppath\spath@tmp
1566 }

Correction for reprocessing path: curvetos
1567 \def\spath@rep@curvetosupporta#1#2\pgfsyssoftpath@curvetosupportbtoken#3#4\pgfsyssoftpath@curve
1568 \pgfpathcurveto{\pgfpoint{#1}-{#2}}{\pgfpoint{#3}-{#4}}{\pgfpoint{#5}-{#6}}%
1569 }%

Correction for converting path to string: curvetos
1570 \def\spath@str@curvetosupporta#1#2\pgfsyssoftpath@curvetosupportbtoken#3#4\pgfsyssoftpath@curve
1571 \pgfmathsetmacro\spath@sx{#1/1cm}%
1572 \pgfmathsetmacro\spath@sy{#2/1cm}%
1573 \string\pgfpathcurveto{\string\pgfpointxy{\spath@sx}\{\spath@sy\}\}%
1574 \pgfmathsetmacro\spath@sx{#3/1cm}%
1575 \pgfmathsetmacro\spath@sy{#4/1cm}%
1576 \{\string\pgfpointxy{\spath@sx}\{\spath@sy\}\}%
1577 \pgfmathsetmacro\spath@sx{#5/1cm}%
1578 \pgfmathsetmacro\spath@sy{#6/1cm}%
1579 \{\string\pgfpointxy{\spath@sx}\{\spath@sy\}\}\}%
1580 }%

Closepath
1581 \def\spath@str@closepath#1#2{%
1582 \string\pgfclosepath\}%

Correction for splitting in to an array
1583 \def\spath@array@moveto#1#2{%
1584 \ifx\spath@tmppath\pgfutil@empty
1585 \else
1586 \expandafter\global\expandafter\let\csname spath@array@\spath@path@name @\the\value{spath@ar
1587 \def\spath@tmppath{\pgfsyssoftpath@movetotoken{#1}-{#2}}%

```

```

1588 \fi
1589 \stepcounter{spath@array}%
1590 }

```

Correction for rounding corners: lineto and curvetosupporta need to add the extra token

```

1591 \def\spath@rnd@lineto#1#2{%
1592 \edef\spath@tmp{%
1593 \noexpand\pgfsyssoftpath@specialroundtoken{\spath@rndx}{\spath@rndy}%
1594 \noexpand\pgfsyssoftpath@linetotoken{#1}{#2}}%
1595 \ge@addto@macro\spath@tmppath\spath@tmp
1596 }
1597 \def\spath@rnd@closepath#1#2{%
1598 \edef\spath@tmp{%
1599 \noexpand\pgfsyssoftpath@specialroundtoken{\spath@rndx}{\spath@rndy}%
1600 \noexpand\pgfsyssoftpath@closepathtoken{#1}{#2}}%
1601 \ge@addto@macro\spath@tmppath\spath@tmp
1602 }
1603 \def\spath@rnd@curvetosupportatoken#1#2{%
1604 \edef\spath@tmp{%
1605 \noexpand\pgfsyssoftpath@specialroundtoken{\spath@rndx}{\spath@rndy}%
1606 \noexpand\pgfsyssoftpath@curvetosupportatoken{#1}{#2}}%
1607 \ge@addto@macro\spath@tmppath\spath@tmp
1608 }

```

Whereas existing special round tokens should take the maximum of their rounding and the specified rounding, and also ensure that the next corner is not *doubly* rounded.

```

1609 \def\spath@rnd@specialround#1#2#3#4#5{%
1610 \pgfmathsetmacro{\spath@srndx}{max(#1,\spath@rndx)}%
1611 \pgfmathsetmacro{\spath@srndy}{max(#1,\spath@rndy)}%
1612 \edef\spath@tmp{%
1613 \noexpand\pgfsyssoftpath@specialroundtoken{\spath@srndx pt}{\spath@srndy pt}%
1614 \noexpand#3{#4}{#5}}%
1615 \ge@addto@macro\spath@tmppath\spath@tmp
1616 }

```

Correction for special rounds: almost nothing should touch these

```

1617 \def\spath@tr@specialround#1#2{%
1618 \edef\spath@tmp{\noexpand\pgfsyssoftpath@specialroundtoken{#1}{#2}}%
1619 \ge@addto@macro\spath@tmppath\spath@tmp
1620 }
1621 \def\spath@rev@specialround#1#2{%
1622 \edef\spath@tmp{\noexpand\pgfsyssoftpath@specialroundtoken{#1}{#2}}%
1623 \ge@addto@macro\spath@tmppath\spath@tmp
1624 }
1625 \def\spath@sh@specialround#1#2{%
1626 \edef\spath@tmp{\noexpand\pgfsyssoftpath@specialroundtoken{#1}{#2}}%
1627 \ge@addto@macro\spath@tmppath\spath@tmp
1628 }
1629 \def\spath@rot@specialround#1#2{%

```



```

1630 \edef\spath@tmp{\noexpand\pgfsyssoftpath@specialroundtoken{#1}{#2}}%
1631 \ge@addto@macro\spath@tmppath\spath@tmp
1632 }

```

The “prepare” routine is too complicated to have defaults, so we need to set up them all here. At the moment, this reverses the path as well. This is now implemented in a different way so should be removed.

```

1633 \def\spath@prepare@moveto#1#2{%
1634 \ifx\spath@tmppath\pgfutil@empty
1635 \g@addbefore@macro\spath@tmppath{{#1}{#2}}%
1636 \edef\spath@sx{#1}%
1637 \edef\spath@sy{#2}%
1638 \edef\spath@mix{#1}%
1639 \edef\spath@miy{#2}%
1640 \edef\spath@max{#1}%
1641 \edef\spath@may{#2}%
1642 \else
1643 \g@addbefore@macro\spath@tmppath{{#1}{#2}\pgfsyssoftpath@movetotoken}%
1644 \pgfmathsetmacro{\spath@mix}{min(\spath@mix,#1)}%
1645 \pgfmathsetmacro{\spath@max}{max(\spath@max,#1)}%
1646 \pgfmathsetmacro{\spath@miy}{min(\spath@miy,#2)}%
1647 \pgfmathsetmacro{\spath@may}{max(\spath@may,#2)}%
1648 \fi
1649 \edef\spath@ex{#1}%
1650 \edef\spath@ey{#2}%
1651 \stepcounter{spath@length}%
1652 \stepcounter{spath@components}%
1653 \let\spath@last\spath@moveto
1654 }
1655
1656 \def\spath@prepare@lineto#1#2{%
1657 \g@addbefore@macro\spath@tmppath{{#1}{#2}\pgfsyssoftpath@linetotoken}%
1658 \edef\spath@ex{#1}%
1659 \edef\spath@ey{#2}%
1660 \pgfmathsetmacro{\spath@mix}{min(\spath@mix,#1)}%
1661 \pgfmathsetmacro{\spath@max}{max(\spath@max,#1)}%
1662 \pgfmathsetmacro{\spath@miy}{min(\spath@miy,#2)}%
1663 \pgfmathsetmacro{\spath@may}{max(\spath@may,#2)}%
1664 \stepcounter{spath@length}%
1665 \stepcounter{spath@reallength}%
1666 \let\spath@last\spath@lineto
1667 \ifx\spath@first\pgfutil@empty
1668 \let\spath@first\spath@lineto
1669 \fi
1670 }
1671
1672 \def\spath@prepare@curveto#1#2{%
1673 \g@addbefore@macro\spath@tmppath{{#1}{#2}\pgfsyssoftpath@curvetosupportatoken}%
1674 \edef\spath@ex{#1}%
1675 \edef\spath@ey{#2}%

```

```

1676 \pgfmathsetmacro{\spath@mix}{min(\spath@mix,#1)}%
1677 \pgfmathsetmacro{\spath@max}{max(\spath@max,#1)}%
1678 \pgfmathsetmacro{\spath@miy}{min(\spath@miy,#2)}%
1679 \pgfmathsetmacro{\spath@may}{max(\spath@may,#2)}%
1680 \stepcounter{spath@length}%
1681 \stepcounter{spath@reallength}%
1682 \let\spath@last\spath@curveto
1683 \ifx\spath@first\pgfutil@empty
1684 \let\spath@first\spath@curveto
1685 \fi
1686 }
1687
1688 \def\spath@prepare@curvetosupporta#1#2{%
1689 \g@addbefore@macro\spath@tmppath{{#1}{#2}\pgfsyssoftpath@curvetotoken}%
1690 \stepcounter{spath@length}%
1691 \pgfmathsetmacro{\spath@mix}{min(\spath@mix,#1)}%
1692 \pgfmathsetmacro{\spath@max}{max(\spath@max,#1)}%
1693 \pgfmathsetmacro{\spath@miy}{min(\spath@miy,#2)}%
1694 \pgfmathsetmacro{\spath@may}{max(\spath@may,#2)}%
1695 }
1696
1697 \def\spath@prepare@curvetosupportb#1#2{%
1698 \g@addbefore@macro\spath@tmppath{{#1}{#2}\pgfsyssoftpath@curvetosupportbtoken}%
1699 \stepcounter{spath@length}%
1700 \pgfmathsetmacro{\spath@mix}{min(\spath@mix,#1)}%
1701 \pgfmathsetmacro{\spath@max}{max(\spath@max,#1)}%
1702 \pgfmathsetmacro{\spath@miy}{min(\spath@miy,#2)}%
1703 \pgfmathsetmacro{\spath@may}{max(\spath@may,#2)}%
1704 }
1705
1706 \def\spath@prepare@closepath#1#2{%
1707 \g@addbefore@macro\spath@tmppath{{#1}{#2}\pgfsyssoftpath@closepathtoken}%
1708 \stepcounter{spath@length}%
1709 }
1710
1711 \def\spath@prepare@specialround#1#2{%
1712 \g@addbefore@macro\spath@tmppath{{#1}{#2}\pgfsyssoftpath@specialroundtoken}%
1713 \stepcounter{spath@length}%
1714 }

```

At the moment, this doesn't handle the closepath variants very well (if at all).
This needs implementing.

This splits a `curveto` if there is a possibility that it self-intersects.

```

1715 \def\spath@selfintersect@curvetosupporta#1#2\pgfsyssoftpath@curvetosupportbtoken#3#4\pgfsyssoftt
1716 \pgfmathsetmacro{\spath@ax}{\spath@ex/28}
1717 \pgfmathsetmacro{\spath@bx}{#1/28}
1718 \pgfmathsetmacro{\spath@cx}{#3/28}
1719 \pgfmathsetmacro{\spath@dx}{#5/28}
1720 \pgfmathsetmacro{\spath@ay}{\spath@ey/28}
1721 \pgfmathsetmacro{\spath@by}{#2/28}

```

```

1722 \pgfmathsetmacro{\spath@cy}{#4/28}
1723 \pgfmathsetmacro{\spath@dy}{#6/28}
1724 \pgfmathsetmacro{\spath@enum}{%
1725   ( (\spath@ay - 3 * \spath@by + 3 * \spath@cy - \spath@dy) * (3 *
1726 \spath@cx - 3 * \spath@dx)
1727   - (\spath@ax - 3 * \spath@bx + 3 * \spath@cx - \spath@dx) * (3 *
1728 \spath@cy - 3 * \spath@dy) )}
1729 \pgfmathsetmacro{\spath@denum}{%
1730   ( (\spath@ax - 3 * \spath@bx + 3 * \spath@cx - \spath@dx) * (3 *
1731 \spath@by - 6 * \spath@cy + 3 * \spath@dy)
1732   - (\spath@ay - 3 * \spath@by + 3 * \spath@cy - \spath@dy) * (3 *
1733 \spath@bx - 6 * \spath@cx + 3 * \spath@dx) )}
1734 \pgfmathtruncatemacro{\spath@split}{\spath@enum > 0 ? (\spath@enum <
1735 2 * \spath@denum) : (\spath@enum > 2 * \spath@denum)}
1736 \ifnum\spath@split=1\relax
1737 \pgfmathsetmacro{\spath@splitt}{.5*(\spath@enum)/(\spath@denum)}%
1738 \pgfsyssoftpath@setcurrentpath{\spath@tmppath}%
1739 \pgfpathcurvebetweentimecontinue{0}{\spath@splitt}{\pgfqpoint{\spath@ex}{\spath@ey}}{\pgfqpoint{\spath@ex}{\spath@ey}}%
1740 \pgfpathcurvebetweentimecontinue{\spath@splitt}{1}{\pgfqpoint{\spath@ex}{\spath@ey}}{\pgfqpoint{\spath@ex}{\spath@ey}}%
1741 \pgfsyssoftpath@getcurrentpath{\spath@tmppath}%
1742 \else
1743 \g@addto@macro\spath@tmppath{%
1744 \pgfsyssoftpath@curvetosupportatoken{#1}{#2}%
1745 \pgfsyssoftpath@curvetosupportbtoken{#3}{#4}%
1746 \pgfsyssoftpath@curvetotoken{#5}{#6}%
1747 }%
1748 \fi
1749 \edef\spath@ex{#5}%
1750 \edef\spath@ey{#6}%
1751 }

```

Now we have some helper macros. This gets rid of the initial `moveto`, leaving the rest in `\spath@tmppath`.

```

1752 \def\spath@trimfirst#1#2#3#4\relax{%
1753   \edef\spath@this@action{\string#1}%
1754   \ifx\spath@this@action\spath@moveto
1755     \def\spath@tmppath{#4}%
1756   \else
1757     \def\spath@tmppath{#1{#2}{#3}{#4}%
1758   \fi
1759 }

```

This replaces the initial `moveto` with a `lineto`, leaving the whole path in `\spath@tmppath`.

```

1760 \def\spath@movetoline#1#2#3#4\relax{%
1761   \edef\spath@this@action{\string#1}%
1762   \ifx\spath@this@action\spath@moveto
1763     \def\spath@tmppath{\pgfsyssoftpath@linetotoken{#2}{#3}{#4}%
1764   \else
1765     \def\spath@tmppath{#1{#2}{#3}{#4}%

```

```

1766 \fi
1767 }

```

This is our gobbling macro for splitting a path according to some criterion (length, real length, or number of components).

```

1768 \def\spath@gobble#1#2#3{%
1769   \stepcounter{spath@length}%
1770   \edef\spath@this@action{\string#1}%
1771   \ifx\spath@this@action\spath@lineto
1772     \stepcounter{spath@reallength}%
1773   \fi
1774   \ifx\spath@this@action\spath@curveto
1775     \stepcounter{spath@reallength}%
1776   \fi
1777   \ifx\spath@this@action\spath@moveto
1778     \stepcounter{spath@components}%
1779   \fi
1780   \g@addto@macro\spath@tmppath{#1{#2}{#3}}%
1781   \ifnum\spath@test=\spath@splitat\relax
1782     \def\spath@tmppath{\pgfsyssoftpath@movetotoken{#2}{#3}}%
1783     \edef\spath@last{\string#1}%
1784     \ifx\spath@last\spath@curvetosupporta
1785       \let\spath@last=\spath@curveto
1786     \fi
1787     \edef\spath@ex{#2}%
1788     \edef\spath@ey{#3}%
1789     \let\spath@next=\spath@lastgobble
1790   \else
1791     \let\spath@next=\spath@gobble
1792   \fi
1793   \pgfutil@ifnextchar\relax{%
1794     \ifx\spath@next\spath@gobble
1795       \def\spath@tmppath{\pgfsyssoftpath@movetotoken{#2}{#3}}%
1796       \edef\spath@last{\string#1}%
1797       \ifx\spath@last\spath@curvetosupporta
1798         \let\spath@last=\spath@curveto
1799       \fi
1800       \edef\spath@ex{#2}%
1801       \edef\spath@ey{#3}%
1802     \fi
1803     \edef\spath@sx{#2}%
1804     \edef\spath@sy{#3}%
1805     \let\spath@first\spath@moveto
1806     }\let\spath@next}%
1807 }
1808 \def\spath@lastgobble#1#2#3#4\relax{%
1809   \g@addto@macro\spath@tmppath{#1{#2}{#3}#4}%
1810   \edef\spath@first{\string#1}%
1811   \ifx\spath@first\spath@curvetosupporta
1812     \let\spath@first=\spath@curveto

```

```

1813 \fi
1814 \edef\spath@sx{#2}%
1815 \edef\spath@sy{#3}%
1816 }

```

We need a few dimensions and counters to keep track of things.

```

1817 \newdimen\spath@trx
1818 \newdimen\spath@try
1819 \newcounter{spath@length}
1820 \newcounter{spath@reallength}
1821 \newcounter{spath@components}
1822 \newcounter{spath@array}

```

`\spath@taper@lineto@out` This macro sets things up for tapering a `lineto`.

```

1823 \def\spath@taper@lineto@out#1#2#3#4#5#6#7\relax{%
    #1 is \pgfsyssoftpath@movetotoken #2 is x-coord of starting point #3 is y-
    coord of starting point #4 is \pgfsyssoftpath@linetotoken #5 is x-coord of
    ending point #6 is y-coord of ending point #7 shouldn't have anything in
1824 \pgfmathsetmacro{\spath@sx}{.7 * #2 + .3 * #5}
1825 \pgfmathsetmacro{\spath@sy}{.7 * #3 + .3 * #6}
1826 \pgfmathsetmacro{\spath@ex}{.3 * #2 + .7 * #5}
1827 \pgfmathsetmacro{\spath@ey}{.3 * #3 + .7 * #6}
1828 \edef\spath@tmp{\noexpand\spath@taper@path{#2}{#3}{\spath@sx pt}{\spath@sy pt}{\spath@ex pt}{\
1829 \spath@tmp
1830 }

```

`\spath@taper@curveto@out` This macro sets things up for tapering a `curveto`.

```

1831 \def\spath@taper@curveto@out\pgfsyssoftpath@movetotoken#1#2\pgfsyssoftpath@curvetosupportatoken
1832 \spath@taper@path{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}
1833 }

```

`\spath@taper@path` This is the actual tapering macro.

```

1834 \def\spath@taper@path#1#2#3#4#5#6#7#8{
    #1 is x-coord of starting point #2 is y-coord of starting point #3 is x-coord of first
    control point #4 is y-coord of first control point #5 is x-coord of second control
    point #6 is y-coord of second control point #7 is x-coord of ending point #8 is
    y-coord of ending point
1835 \edef\spath@sx{#1}
1836 \edef\spath@sy{#2}
1837 \edef\spath@cx{#3}
1838 \edef\spath@cy{#4}
1839 \edef\spath@dx{#5}
1840 \edef\spath@dy{#6}
1841 \edef\spath@ex{#7}
1842 \edef\spath@ey{#8}
1843

```

Orthogonal vector at end

```
1844 \pgfmathsetmacro{\spath@ox}{-\spath@ey + \spath@dy}
1845 \pgfmathsetmacro{\spath@oy}{\spath@ex - \spath@dx}
1846
```

Orthogonal vector at start

```
1847 \pgfmathsetmacro{\spath@sox}{\spath@sy - \spath@cy}
1848 \pgfmathsetmacro{\spath@soy}{-\spath@sx + \spath@cx}
1849
```

Adjust length to half the line width

```
1850 \pgfmathsetmacro{\spath@oox}{.5*\spath@ox * \pgflinewidth / vecLen(\spath@ox,\spath@oy)}
1851 \pgfmathsetmacro{\spath@ooy}{.5*\spath@oy * \pgflinewidth / vecLen(\spath@ox,\spath@oy)}
1852
```

Adjust length to half the thinner line width

```
1853 \pgfmathsetmacro{\spath@soox}{.5*\spath@sox * \taper@line@width / vecLen(\spath@sox,\spath@soy)}
1854 \pgfmathsetmacro{\spath@ssoy}{.5*\spath@ssoy * \taper@line@width / vecLen(\spath@sox,\spath@soy)}
```

Shift the start point, the control points, and the end points

```
1855 \pgfmathsetmacro{\spath@sx}{\spath@sx + \spath@soox}
1856 \pgfmathsetmacro{\spath@sy}{\spath@sy + \spath@ssoy}
1857 \pgfmathsetmacro{\spath@cx}{\spath@cx + \spath@soox}
1858 \pgfmathsetmacro{\spath@cy}{\spath@cy + \spath@ssoy}
1859 \pgfmathsetmacro{\spath@dx}{\spath@dx + \spath@oox}
1860 \pgfmathsetmacro{\spath@dy}{\spath@dy + \spath@ooy}
1861 \pgfmathsetmacro{\spath@ex}{\spath@ex + \spath@oox}
1862 \pgfmathsetmacro{\spath@ey}{\spath@ey + \spath@ooy}
```

Add the first pieces

```
1863 \let\spath@tapered@path=\pgfutil@empty
1864 \edef\spath@to@add{\noexpand\pgfsyssoftpath@movetotoken{\spath@sx pt}{\spath@sy pt}}
1865 \ge@addto@macro\spath@tapered@path\spath@to@add
1866 \edef\spath@to@add{\noexpand\pgfsyssoftpath@curvetosupportatoken{\spath@cx pt}{\spath@cy pt}}
1867 \ge@addto@macro\spath@tapered@path\spath@to@add
1868 \edef\spath@to@add{\noexpand\pgfsyssoftpath@curvetosupportbtoken{\spath@dx pt}{\spath@dy pt}}
1869 \ge@addto@macro\spath@tapered@path\spath@to@add
1870 \edef\spath@to@add{\noexpand\pgfsyssoftpath@curvetotoken{\spath@ex pt}{\spath@ey pt}}
1871 \ge@addto@macro\spath@tapered@path\spath@to@add
```

Make the end roughly round

```
1872 \pgfmathsetmacro{\spath@fx}{\spath@ex + 1.32*\spath@ooy}
1873 \pgfmathsetmacro{\spath@fy}{\spath@ey - 1.32*\spath@oox}
1874 \edef\spath@to@add{\noexpand\pgfsyssoftpath@curvetosupportatoken{\spath@fx pt}{\spath@fy pt}}
1875 \ge@addto@macro\spath@tapered@path\spath@to@add
```

Shift the end points and the control points again

```
1876 \pgfmathsetmacro{\spath@fx}{\spath@fx - 2*\spath@oox}
1877 \pgfmathsetmacro{\spath@fy}{\spath@fy - 2*\spath@ooy}
1878 \pgfmathsetmacro{\spath@dx}{\spath@dx - 2*\spath@oox}
1879 \pgfmathsetmacro{\spath@dy}{\spath@dy - 2*\spath@ooy}
1880 \pgfmathsetmacro{\spath@ex}{\spath@ex - 2*\spath@oox}
```

```

1881 \pgfmathsetmacro{\spath@ey}{\spath@ey - 2*\spath@ooy}
1882 \pgfmathsetmacro{\spath@cx}{\spath@cx - 2*\spath@soox}
1883 \pgfmathsetmacro{\spath@cy}{\spath@cy - 2*\spath@sooy}
1884 \pgfmathsetmacro{\spath@sx}{\spath@sx - 2*\spath@soox}
1885 \pgfmathsetmacro{\spath@sy}{\spath@sy - 2*\spath@sooy}
1886 \edef\spath@to@add{\noexpand\pgfsyssoftpath@curvetosupportbtoken{\spath@fx pt}{\spath@fy pt}}
1887 \ge@addto@macro\spath@tapered@path\spath@to@add
1888 \edef\spath@to@add{\noexpand\pgfsyssoftpath@curvetotoken{\spath@ex pt}{\spath@ey pt}}
1889 \ge@addto@macro\spath@tapered@path\spath@to@add
1890 \edef\spath@to@add{\noexpand\pgfsyssoftpath@curvetosupportatoken{\spath@dx pt}{\spath@dy pt}}
1891 \ge@addto@macro\spath@tapered@path\spath@to@add
1892 \edef\spath@to@add{\noexpand\pgfsyssoftpath@curvetosupportbtoken{\spath@cx pt}{\spath@cy pt}}
1893 \ge@addto@macro\spath@tapered@path\spath@to@add
1894 \edef\spath@to@add{\noexpand\pgfsyssoftpath@curvetotoken{\spath@sx pt}{\spath@sy pt}}
1895 \ge@addto@macro\spath@tapered@path\spath@to@add

```

Make the end roughly round

```

1896 \pgfmathsetmacro{\spath@fx}{\spath@sx - 1.32*\spath@sooy}
1897 \pgfmathsetmacro{\spath@fy}{\spath@sy + 1.32*\spath@soox}
1898 \edef\spath@to@add{\noexpand\pgfsyssoftpath@curvetosupportatoken{\spath@fx pt}{\spath@fy pt}}
1899 \ge@addto@macro\spath@tapered@path\spath@to@add
1900 \pgfmathsetmacro{\spath@fx}{\spath@fx + 2*\spath@soox}
1901 \pgfmathsetmacro{\spath@fy}{\spath@fy + 2*\spath@sooy}
1902 \edef\spath@to@add{\noexpand\pgfsyssoftpath@curvetosupportbtoken{\spath@fx pt}{\spath@fy pt}}
1903 \ge@addto@macro\spath@tapered@path\spath@to@add
1904 \pgfmathsetmacro{\spath@sx}{\spath@sx + 2*\spath@soox}
1905 \pgfmathsetmacro{\spath@sy}{\spath@sy + 2*\spath@sooy}
1906 \edef\spath@to@add{\noexpand\pgfsyssoftpath@curvetotoken{\spath@sx pt}{\spath@sy pt}}
1907 \ge@addto@macro\spath@tapered@path\spath@to@add

```

Close the path, ought to make it rounded

```

1908 % \ge@addto@macro{\spath@tapered@path}{\pgfsyssoftpath@closepathtoken}
1909 }

```

The following have been modified to fit the new routines

\spath@single@split

```

1910 \def\spath@single@split{.3}

```

\spath@split@single@lineto This splits a single lineto into three pieces.

```

1911 \def\spath@split@single@lineto\pgfsyssoftpath@movetotoken#1#2\pgfsyssoftpath@linetotoken#3#4#5\
1912 \pgfmathsetmacro{\spath@sx}{(1-\spath@single@split)*#1 + \spath@single@split * #3}
1913 \pgfmathsetmacro{\spath@sy}{(1-\spath@single@split)*#2 + \spath@single@split * #4}
1914 \pgfmathsetmacro{\spath@ex}{(1-\spath@single@split)*#3 + \spath@single@split * #1}
1915 \pgfmathsetmacro{\spath@ey}{(1-\spath@single@split)*#4 + \spath@single@split * #2}
1916 \edef\spath@tmppath{\noexpand\pgfsyssoftpath@movetotoken{#1}{#2}\noexpand\pgfsyssoftpath@line
1917 }

```

\spath@split@single@curveto This splits a single curveto into three pieces.

```

1918 \def\spath@split@single@curveto\pgfsyssoftpath@movetotoken#1#2\pgfsyssoftpath@curvetosupportatoken#3#4#5\

```

```

1919 \pgfmathsetmacro{\spath@cx}{(1 - \spath@single@split)*#1 + \spath@single@split * #3}
1920 \pgfmathsetmacro{\spath@cy}{(1 - \spath@single@split)*#2 + \spath@single@split * #4}
1921 \pgfmathsetmacro{\spath@dx}{(1 - \spath@single@split)^2 * #1 + 2*\spath@single@split * (1 - \
1922 \pgfmathsetmacro{\spath@dy}{(1 - \spath@single@split)^2 * #2 + 2*\spath@single@split * (1 - \
1923 \pgfmathsetmacro{\spath@ex}{(1 - \spath@single@split)^3 * #1 + 3*\spath@single@split * (1 - \
1924 \pgfmathsetmacro{\spath@ey}{(1 - \spath@single@split)^3 * #2 + 3*\spath@single@split * (1 - \
1925 \edef\spath@tmppath{\noexpand\pgfsyssoftpath@movetotoken{#1}{#2}\noexpand\pgfsyssoftpath@curv

```

Should be some sort of optimisation to do here

```

1926 \pgfmathsetmacro{\spath@cx}{(1 - \spath@single@split)^2*\spath@single@split * #1 + (1 - 3 * \
1927 \pgfmathsetmacro{\spath@cy}{(1 - \spath@single@split)^2*\spath@single@split * #2 + (1 - 3 * \
1928 \pgfmathsetmacro{\spath@dx}{(1 - \spath@single@split)^2*\spath@single@split * #7 + (1 - 3 * \
1929 \pgfmathsetmacro{\spath@dy}{(1 - \spath@single@split)^2*\spath@single@split * #8 + (1 - 3 * \
1930 \pgfmathsetmacro{\spath@ex}{\spath@single@split^3 * #1 + 3 * \spath@single@split^2 * (1 - \sp
1931 \pgfmathsetmacro{\spath@ey}{\spath@single@split^3 * #2 + 3 * \spath@single@split^2 * (1 - \sp
1932 \edef\spath@to@add{\noexpand\pgfsyssoftpath@curvetosupportatoken{\spath@cx pt}{\spath@cy
1933 \ge@addto@macro\spath@tmppath\spath@to@add
1934 \pgfmathsetmacro{\spath@dx}{(1 - \spath@single@split)*#7 + \spath@single@split * #5}
1935 \pgfmathsetmacro{\spath@dy}{(1 - \spath@single@split)*#8 + \spath@single@split * #6}
1936 \pgfmathsetmacro{\spath@cx}{(1 - \spath@single@split)^2 * #7 + 2*\spath@single@split * (1 - \
1937 \pgfmathsetmacro{\spath@cy}{(1 - \spath@single@split)^2 * #8 + 2*\spath@single@split * (1 - \
1938 \edef\spath@to@add{\noexpand\pgfsyssoftpath@curvetosupportatoken{\spath@cx pt}{\spath@cy
1939 \ge@addto@macro\spath@tmppath\spath@to@add
1940 }

```

The next routines split two path tokens into four, they therefore split each one in half.

`\spath@double@split`

```

1941 \def\spath@double@split{.5}

```

`\spath@split@first@lineto` This splits the first token if it is a lineto.

```

1942 \def\spath@split@first@lineto\pgfsyssoftpath@movetotoken#1#2\pgfsyssoftpath@linetotoken#3#4#5\
1943 \pgfmathsetmacro{\spath@sx}{(1-\spath@single@split)*#1 + \spath@single@split * #3}
1944 \pgfmathsetmacro{\spath@sy}{(1-\spath@single@split)*#2 + \spath@single@split * #4}
1945 \edef\spath@split@path@start{\noexpand\pgfsyssoftpath@movetotoken{#1}{#2}\noexpand\pgfsyssoft
1946 \edef\spath@split@path@end{\noexpand\pgfsyssoftpath@movetotoken{#3}{#4}}
1947 \ge@addto@macro\spath@split@path@end{#5}
1948 }

```

`\spath@split@second@lineto` This splits the second token if it is a lineto.

```

1949 \def\spath@split@second@lineto\pgfsyssoftpath@movetotoken#1#2\pgfsyssoftpath@linetotoken#3#4#5\
1950 \pgfmathsetmacro{\spath@sx}{(1-\spath@single@split)*#3 + \spath@single@split * #1}
1951 \pgfmathsetmacro{\spath@sy}{(1-\spath@single@split)*#4 + \spath@single@split * #2}
1952 \edef\spath@split@path@end{\noexpand\pgfsyssoftpath@linetotoken{\spath@sx pt}{\spath@sy pt}\n
1953 }

```

`\spath@split@first@curveto` This splits the first token if it is a curveto.

```

1954 \def\spath@split@first@curveto\pgfsyssoftpath@movetotoken#1#2\pgfsyssoftpath@curvetosupportatok
1955 \pgfmathsetmacro{\spath@cx}{(1 - \spath@double@split)*#1 + \spath@double@split * #3}

```



```

1956 \pgfmathsetmacro{\spath@cy}{(1 - \spath@double@split)*#2 + \spath@double@split * #4}
1957 \pgfmathsetmacro{\spath@dx}{(1 - \spath@double@split)^2 * #1 + 2*\spath@double@split * (1 - \
1958 \pgfmathsetmacro{\spath@dy}{(1 - \spath@double@split)^2 * #2 + 2*\spath@double@split * (1 - \
1959 \pgfmathsetmacro{\spath@ex}{(1 - \spath@double@split)^3 * #1 + 3*\spath@double@split * (1 - \
1960 \pgfmathsetmacro{\spath@ey}{(1 - \spath@double@split)^3 * #2 + 3*\spath@double@split * (1 - \
1961 \edef\spath@split@path@start{\noexpand\pgfsyssoftpath@movetotoken{#1}{#2}\noexpand\pgfsyssoft
1962 \pgfmathsetmacro{\spath@dx}{(1 - \spath@double@split)*#5 + \spath@double@split * #7}
1963 \pgfmathsetmacro{\spath@dy}{(1 - \spath@double@split)*#6 + \spath@double@split * #8}
1964 \pgfmathsetmacro{\spath@cx}{(1 - \spath@double@split)^2 * #3 + 2*\spath@double@split * (1 - \
1965 \pgfmathsetmacro{\spath@cy}{(1 - \spath@double@split)^2 * #4 + 2*\spath@double@split * (1 - \
1966
1967 \edef\spath@to@add{\noexpand\pgfsyssoftpath@curvetosupportatoken{\spath@cx pt}{\spath@cy
1968 \ge@addto@macro\spath@split@path@start\spath@to@add
1969 \edef\spath@split@path@end{\noexpand\pgfsyssoftpath@movetotoken{#7}{#8}}
1970 \ge@addto@macro\spath@split@path@end{#9}
1971 }

```

`\spath@split@second@curveto` This splits the second token if it is a `curveto`.

```

1972 \def\spath@split@second@curveto\pgfsyssoftpath@movetotoken#1#2\pgfsyssoftpath@curvetosupportatoken#3#4#5#6#7#8#9#10#11#12#13#14#15#16#17#18#19#20#21#22#23#24#25#26#27#28#29#30#31#32#33#34#35#36#37#38#39#40#41#42#43#44#45#46#47#48#49#50#51#52#53#54#55#56#57#58#59#60#61#62#63#64#65#66#67#68#69#70#71#72#73#74#75#76#77#78#79#80#81#82#83#84#85#86#87#88#89#90#91#92#93#94#95#96#97#98#99#100#101#102#103#104#105#106#107#108#109#110#111#112#113#114#115#116#117#118#119#120#121#122#123#124#125#126#127#128#129#130#131#132#133#134#135#136#137#138#139#140#141#142#143#144#145#146#147#148#149#150#151#152#153#154#155#156#157#158#159#160#161#162#163#164#165#166#167#168#169#170#171#172#173#174#175#176#177#178#179#180#181#182#183#184#185#186#187#188#189#190#191#192#193#194#195#196#197#198#199#200#201#202#203#204#205#206#207#208#209#210#211#212#213#214#215#216#217#218#219#220#221#222#223#224#225#226#227#228#229#230#231#232#233#234#235#236#237#238#239#240#241#242#243#244#245#246#247#248#249#250#251#252#253#254#255#256#257#258#259#260#261#262#263#264#265#266#267#268#269#270#271#272#273#274#275#276#277#278#279#280#281#282#283#284#285#286#287#288#289#290#291#292#293#294#295#296#297#298#299#300#301#302#303#304#305#306#307#308#309#310#311#312#313#314#315#316#317#318#319#320#321#322#323#324#325#326#327#328#329#330#331#332#333#334#335#336#337#338#339#340#341#342#343#344#345#346#347#348#349#350#351#352#353#354#355#356#357#358#359#360#361#362#363#364#365#366#367#368#369#370#371#372#373#374#375#376#377#378#379#380#381#382#383#384#385#386#387#388#389#390#391#392#393#394#395#396#397#398#399#400#401#402#403#404#405#406#407#408#409#410#411#412#413#414#415#416#417#418#419#420#421#422#423#424#425#426#427#428#429#430#431#432#433#434#435#436#437#438#439#440#441#442#443#444#445#446#447#448#449#450#451#452#453#454#455#456#457#458#459#460#461#462#463#464#465#466#467#468#469#470#471#472#473#474#475#476#477#478#479#480#481#482#483#484#485#486#487#488#489#490#491#492#493#494#495#496#497#498#499#500#501#502#503#504#505#506#507#508#509#510#511#512#513#514#515#516#517#518#519#520#521#522#523#524#525#526#527#528#529#530#531#532#533#534#535#536#537#538#539#540#541#542#543#544#545#546#547#548#549#550#551#552#553#554#555#556#557#558#559#560#561#562#563#564#565#566#567#568#569#570#571#572#573#574#575#576#577#578#579#580#581#582#583#584#585#586#587#588#589#590#591#592#593#594#595#596#597#598#599#600#601#602#603#604#605#606#607#608#609#610#611#612#613#614#615#616#617#618#619#620#621#622#623#624#625#626#627#628#629#630#631#632#633#634#635#636#637#638#639#640#641#642#643#644#645#646#647#648#649#650#651#652#653#654#655#656#657#658#659#660#661#662#663#664#665#666#667#668#669#670#671#672#673#674#675#676#677#678#679#680#681#682#683#684#685#686#687#688#689#690#691#692#693#694#695#696#697#698#699#700#701#702#703#704#705#706#707#708#709#710#711#712#713#714#715#716#717#718#719#720#721#722#723#724#725#726#727#728#729#730#731#732#733#734#735#736#737#738#739#740#741#742#743#744#745#746#747#748#749#750#751#752#753#754#755#756#757#758#759#760#761#762#763#764#765#766#767#768#769#770#771#772#773#774#775#776#777#778#779#780#781#782#783#784#785#786#787#788#789#790#791#792#793#794#795#796#797#798#799#800#801#802#803#804#805#806#807#808#809#810#811#812#813#814#815#816#817#818#819#820#821#822#823#824#825#826#827#828#829#830#831#832#833#834#835#836#837#838#839#840#841#842#843#844#845#846#847#848#849#850#851#852#853#854#855#856#857#858#859#860#861#862#863#864#865#866#867#868#869#870#871#872#873#874#875#876#877#878#879#880#881#882#883#884#885#886#887#888#889#890#891#892#893#894#895#896#897#898#899#900#901#902#903#904#905#906#907#908#909#910#911#912#913#914#915#916#917#918#919#920#921#922#923#924#925#926#927#928#929#930#931#932#933#934#935#936#937#938#939#940#941#942#943#944#945#946#947#948#949#950#951#952#953#954#955#956#957#958#959#960#961#962#963#964#965#966#967#968#969#970#971#972#973#974#975#976#977#978#979#980#981#982#983#984#985#986#987#988#989#990#991#992#993#994#995#996#997#998#999#1000#1001#1002#1003#1004#1005#1006#1007#1008#1009#1010#1011#1012#1013#1014#1015#1016#1017#1018#1019#1020#1021#1022#1023#1024#1025#1026#1027#1028#1029#1030#1031#1032#1033#1034#1035#1036#1037#1038#1039#1040#1041#1042#1043#1044#1045#1046#1047#1048#1049#1050#1051#1052#1053#1054#1055#1056#1057#1058#1059#1060#1061#1062#1063#1064#1065#1066#1067#1068#1069#1070#1071#1072#1073#1074#1075#1076#1077#1078#1079#1080#1081#1082#1083#1084#1085#1086#1087#1088#1089#1090#1091#1092#1093#1094#1095#1096#1097#1098#1099#1100#1101#1102#1103#1104#1105#1106#1107#1108#1109#1110#1111#1112#1113#1114#1115#1116#1117#1118#1119#1120#1121#1122#1123#1124#1125#1126#1127#1128#1129#1130#1131#1132#1133#1134#1135#1136#1137#1138#1139#1140#1141#1142#1143#1144#1145#1146#1147#1148#1149#1150#1151#1152#1153#1154#1155#1156#1157#1158#1159#1160#1161#1162#1163#1164#1165#1166#1167#1168#1169#1170#1171#1172#1173#1174#1175#1176#1177#1178#1179#1180#1181#1182#1183#1184#1185#1186#1187#1188#1189#1190#1191#1192#1193#1194#1195#1196#1197#1198#1199#1200#1201#1202#1203#1204#1205#1206#1207#1208#1209#1210#1211#1212#1213#1214#1215#1216#1217#1218#1219#1220#1221#1222#1223#1224#1225#1226#1227#1228#1229#1230#1231#1232#1233#1234#1235#1236#1237#1238#1239#1240#1241#1242#1243#1244#1245#1246#1247#1248#1249#1250#1251#1252#1253#1254#1255#1256#1257#1258#1259#1260#1261#1262#1263#1264#1265#1266#1267#1268#1269#1270#1271#1272#1273#1274#1275#1276#1277#1278#1279#1280#1281#1282#1283#1284#1285#1286#1287#1288#1289#1290#1291#1292#1293#1294#1295#1296#1297#1298#1299#1300#1301#1302#1303#1304#1305#1306#1307#1308#1309#1310#1311#1312#1313#1314#1315#1316#1317#1318#1319#1320#1321#1322#1323#1324#1325#1326#1327#1328#1329#1330#1331#1332#1333#1334#1335#1336#1337#1338#1339#1340#1341#1342#1343#1344#1345#1346#1347#1348#1349#1350#1351#1352#1353#1354#1355#1356#1357#1358#1359#1360#1361#1362#1363#1364#1365#1366#1367#1368#1369#1370#1371#1372#1373#1374#1375#1376#1377#1378#1379#1380#1381#1382#1383#1384#1385#1386#1387#1388#1389#1390#1391#1392#1393#1394#1395#1396#1397#1398#1399#1400#1401#1402#1403#1404#1405#1406#1407#1408#1409#1410#1411#1412#1413#1414#1415#1416#1417#1418#1419#1420#1421#1422#1423#1424#1425#1426#1427#1428#1429#1430#1431#1432#1433#1434#1435#1436#1437#1438#1439#1440#1441#1442#1443#1444#1445#1446#1447#1448#1449#1450#1451#1452#1453#1454#1455#1456#1457#1458#1459#1460#1461#1462#1463#1464#1465#1466#1467#1468#1469#1470#1471#1472#1473#1474#1475#1476#1477#1478#1479#1480#1481#1482#1483#1484#1485#1486#1487#1488#1489#1490#1491#1492#1493#1494#1495#1496#1497#1498#1499#1500#1501#1502#1503#1504#1505#1506#1507#1508#1509#1510#1511#1512#1513#1514#1515#1516#1517#1518#1519#1520#1521#1522#1523#1524#1525#1526#1527#1528#1529#1530#1531#1532#1533#1534#1535#1536#1537#1538#1539#1540#1541#1542#1543#1544#1545#1546#1547#1548#1549#1550#1551#1552#1553#1554#1555#1556#1557#1558#1559#1560#1561#1562#1563#1564#1565#1566#1567#1568#1569#1570#1571#1572#1573#1574#1575#1576#1577#1578#1579#1580#1581#1582#1583#1584#1585#1586#1587#1588#1589#1590#1591#1592#1593#1594#1595#1596#1597#1598#1599#1600#1601#1602#1603#1604#1605#1606#1607#1608#1609#1610#1611#1612#1613#1614#1615#1616#1617#1618#1619#1620#1621#1622#1623#1624#1625#1626#1627#1628#1629#1630#1631#1632#1633#1634#1635#1636#1637#1638#1639#1640#1641#1642#1643#1644#1645#1646#1647#1648#1649#1650#1651#1652#1653#1654#1655#1656#1657#1658#1659#1660#1661#1662#1663#1664#1665#1666#1667#1668#1669#1670#1671#1672#1673#1674#1675#1676#1677#1678#1679#1680#1681#1682#1683#1684#1685#1686#1687#1688#1689#1690#1691#1692#1693#1694#1695#1696#1697#1698#1699#1700#1701#1702#1703#1704#1705#1706#1707#1708#1709#1710#1711#1712#1713#1714#1715#1716#1717#1718#1719#1720#1721#1722#1723#1724#1725#1726#1727#1728#1729#1730#1731#1732#1733#1734#1735#1736#1737#1738#1739#1740#1741#1742#1743#1744#1745#1746#1747#1748#1749#1750#1751#1752#1753#1754#1755#1756#1757#1758#1759#1760#1761#1762#1763#1764#1765#1766#1767#1768#1769#1770#1771#1772#1773#1774#1775#1776#1777#1778#1779#1780#1781#1782#1783#1784#1785#1786#1787#1788#1789#1790#1791#1792#1793#1794#1795#1796#1797#1798#1799#1800#1801#1802#1803#1804#1805#1806#1807#1808#1809#1810#1811#1812#1813#1814#1815#1816#1817#1818#1819#1820#1821#1822#1823#1824#1825#1826#1827#1828#1829#1830#1831#1832#1833#1834#1835#1836#1837#1838#1839#1840#1841#1842#1843#1844#1845#1846#1847#1848#1849#1850#1851#1852#1853#1854#1855#1856#1857#1858#1859#1860#1861#1862#1863#1864#1865#1866#1867#1868#1869#1870#1871#1872#1873#1874#1875#1876#1877#1878#1879#1880#1881#1882#1883#1884#1885#1886#1887#1888#1889#1890#1891#1892#1893#1894#1895#1896#1897#1898#1899#1900#1901#1902#1903#1904#1905#1906#1907#1908#1909#1910#1911#1912#1913#1914#1915#1916#1917#1918#1919#1920#1921#1922#1923#1924#1925#1926#1927#1928#1929#1930#1931#1932#1933#1934#1935#1936#1937#1938#1939#1940#1941#1942#1943#1944#1945#1946#1947#1948#1949#1950#1951#1952#1953#1954#1955#1956#1957#1958#1959#1960#1961#1962#1963#1964#1965#1966#1967#1968#1969#1970#1971#1972#1973#1974#1975#1976#1977#1978#1979#1980#1981#1982#1983#1984#1985#1986#1987#1988#1989#1990#1991#1992#1993#1994#1995#1996#1997#1998#1999#2000#2001#2002#2003#2004#2005#2006#2007#2008#2009#2010#2011#2012#2013#2014#2015#2016#2017#2018#2019#2020#2021#2022#2023#2024#2025#2026#2027#2028#2029#2030#2031#2032#2033#2034#2035#2036#2037#2038#2039#2040#2041#2042#2043#2044#2045#2046#2047#2048#2049#2050#2051#2052#2053#2054#2055#2056#2057#2058#2059#2060#2061#2062#2063#2064#2065#2066#2067#2068#2069#2070#2071#2072#2073#2074#2075#2076#2077#2078#2079#2080#2081#2082#2083#2084#2085#2086#2087#2088#2089#2090#2091#2092#2093#2094#2095#2096#2097#2098#2099#2100#2101#2102#2103#2104#2105#2106#2107#2108#2109#2110#2111#2112#2113#2114#2115#2116#2117#2118#2119#2120#2121#2122#2123#2124#2125#2126#2127#2128#2129#2130#2131#2132#2133#2134#2135#2136#2137#2138#2139#2140#2141#2142#2143#2144#2145#2146#2147#2148#2149#2150#2151#2152#2153#2154#2155#2156#2157#2158#2159#2160#2161#2162#2163#2164#2165#2166#2167#2168#2169#2170#2171#2172#2173#2174#2175#2176#2177#2178#2179#2180#2181#2182#2183#2184#2185#2186#2187#2188#2189#2190#2191#2192#2193#2194#2195#2196#2197#2198#2199#2200#2201#2202#2203#2204#2205#2206#2207#2208#2209#2210#2211#2212#2213#2214#2215#2216#2217#2218#2219#2220#2221#2222#2223#2224#2225#2226#2227#2228#2229#2230#2231#2232#2233#2234#2235#2236#2237#2238#2239#2240#2241#2242#2243#2244#2245#2246#2247#2248#2249#2250#2251#2252#2253#2254#2255#2256#2257#2258#2259#2260#2261#2262#2263#2264#2265#2266#2267#2268#2269#2270#2271#2272#2273#2274#2275#2276#2277#2278#2279#2280#2281#2282#2283#2284#2285#2286#2287#2288#2289#2290#2291#2292#2293#2294#2295#2296#2297#2298#2299#2300#2301#2302#2303#2304#2305#2306#2307#2308#2309#2310#2311#2312#2313#2314#2315#2316#2317#2318#2319#2320#2321#2322#2323#2324#2325#2326#2327#2328#2329#2330#2331#2332#2333#2334#2335#2336#2337#2338#2339#2340#2341#2342#2343#2344#2345#2346#2347#2348#2349#2350#2351#2352#2353#2354#2355#2356#2357#2358#2359#2360#2361#2362#2363#2364#2365#2366#2367#2368#2369#2370#2371#2372#2373#2374#2375#2376#2377#2378#2379#2380#2381#2382#2383#2384#2385#2386#2387#2388#2389#2390#2391#2392#2393#2394#2395#2396#2397#2398#2399#2400#2401#2402#2403#2404#2405#2406#2407#2408#2409#2410#2411#2412#2413#2414#2415#2416#2417#2418#2419#2420#2421#2422#2423#2424#2425#2426#2427#2428#2429#2430#2431#2432#2433#2434#2435#2436#2437#2438#2439#2440#2441#2442#2443#2444#2445#2446#2447#2448#2449#2450#2451#2452#2453#2454#2455#2456#2457#2458#2459#2460#2461#2462#2463#2464#2465#2466#2467#2468#2469#2470#2471#2472#2473#2474#2475#2476#2477#2478#2479#2480#2481#2482#2483#2484#2485#2486#2487#2488#2489#2490#2491#2492#2493#2494#2495#2496#2497#2498#2499#2500#2501#2502#2503#2504#2505#2506#2507#2508#2509#2510#2511#2512#2513#2514#2515#2516#2517#2518#2519#2520#2521#2522#2523#2524#2525#2526#2527#2528#2529#2530#2531#2532#2533#2534#2535#2536#2537#2538#2539#2540#2541#2542#2543#2544#2545#2546#2547#2548#2549#2550#2551#
```