



# neural-sketch-bridges.sty

Path bridging, spath, fillbetween for neural-sketch

Vincenzo Buono

March 26, 2025

## 1 Introduction

This file provides bridging arcs, connect macros, etc.

implementation

!!! i\*package!

```
1 \ExplSyntaxOn
2
3 % ~~~~~
4 % Global Data Structures
5 % A counter to generate unique path names.
6 % ~~~~~ <<<
7 \int_new:N \g__nsk_connect_counter_int
8
9 % ~~~~~
10 % A sequence storing bridging info for each path:
11 % Each item is a comma list:
12 %   pathName, arrowStyle, color, bridgingStyle, bridgingPath,
13 %   bridgingSpan, bridgingGap
14 % ~~~~~ <<<
15 \seq_new:N \g__nsk_bridging_info_seq
16
17
18 % ~~~~~
19 % Storing One Connect Item
20 % - (path name, arrow style, color, bridging style, bridging path,
21 %   bridging span, bridging gap)
22 % ~~~~~ <<<
23 \cs_new_protected_nopar:Npn \__nsk_connect_store_item:nnnnnnnn #1#2#3#4#5#6#7#8
24 {
25   % We'll build a single token list item, then push it into
26   % the bridging sequence.
27   \tl_clear_new:N \l__nsk_storeitem_tl
28
29   % Insert #1..#7 by expansion (x), forming a comma list:
30   \exp_args:NNx \tl_put_right:Nn \l__nsk_storeitem_tl
31   { #1, #2, { #3 }, #4, #5, #6, #7, #8 }
32
33   % Now append that item to the bridging sequence:
34   \seq_gput_right:No \g__nsk_bridging_info_seq \l__nsk_storeitem_tl
35
```

```

36 % Debug lines (currently commented out):
37 \seq_show:N \g__nsk_bridging_info_seq
38 \tl_show:N \l__nsk_storeitem_tl
39 }
40
41
42
43
44 % build/tracker ~~~~~ <<<
45 % A boolean to track bridging = true/false
46 \bool_new:N \l__nsk_connect_bridging_bool
47
48 % ~~~~~
49 % Main nsk / connect Primitive
50 % ~~~~~ <<<
51 \keys_define:nn { nsk / connect }
52 {
53 % ref-id ~~~~~ <<<
54 id .tl_set:N = \l_nsk_connect_id_tl,
55 id .initial:n = { },
56 id .default:n = { },
57
58 % end-points ~~~~~ <<<
59 from .tl_set:N = \l_nsk_connect_from_tl,
60 from .value_required:n = true,
61
62 to .tl_set:N = \l_nsk_connect_to_tl,
63 to .value_required:n = true,
64
65 route .tl_set:N = \l_nsk_connect_route_tl,
66 route .initial:n = { -- },
67
68 % bridging ~~~~~ <<<
69 bridging .choice:,
70 bridging / true .code:n = { \bool_set_true:N \l__nsk_connect_bridging_bool },
71 bridging / false .code:n = { \bool_set_false:N \l__nsk_connect_bridging_bool },
72 bridging .initial:n = { false },
73
74 bridging-style .choice:,
75 bridging-style / over .code:n = { \tl_set:Nn \l_nsk_connect_bridging_style_tl { over } },
76 bridging-style / under .code:n = { \tl_set:Nn \l_nsk_connect_bridging_style_tl { under } },
77 bridging-style .initial:n = { under },
78 bridging-style .default:n = { under },
79
80 bridging-path .tl_set:N = \l_nsk_connect_bridging_path_tl,
81 bridging-path .initial:n = { arc },
82 bridging-path .default:n = { arc },
83
84 bridging-span .tl_set:N = \l_nsk_connect_bridging_span_tl,
85 bridging-span .initial:n = { 8pt },
86 bridging-span .default:n = { 8pt },
87
88 bridging-gap .tl_set:N = \l_nsk_connect_bridging_gap_tl,
89 bridging-gap .initial:n = { 4pt },

```

```

90 bridging-gap .default:n          = { 4pt },
91
92 arrow-style .tl_set:N = \l_nsk_connect_arrow_style_tl,
93 arrow-style .initial:n = {ultra~thick, line~cap=round},
94 arrow-style .default:n = {ultra~thick, line~cap=round},
95
96 arrow-tip .tl_set:N = \l_nsk_connect_arrow_tip_tl,
97 arrow-tip .initial:n = {-},
98 arrow-tip .default:n = {-},
99
100 shift-x .dim_set:N = \l_nsk_connect_shift_x_dim,
101 shift-x .initial:n = {0mm},
102 shift-x .default:n = {0mm},
103
104 shift-y .dim_set:N = \l_nsk_connect_shift_y_dim,
105 shift-y .initial:n = {0mm},
106 shift-y .default:n = {0mm},
107
108 color .tl_set:N          = \l_nsk_connect_color_tl,
109 color .initial:n         = { c__nsk_principal },
110 color .default:n         = { c__nsk_principal },
111
112 % corner-radius ~~~~~ <<<
113 corner-radius .dim_set:N = \l__nsk_connect_corner_radius_dim,
114 corner-radius .initial:n = { 3mm },
115 corner-radius .default:n = { 3mm },
116
117 % bend-routing ~~~~~ <<<
118 bend-type .choice:,
119 bend-type / none .code:n = { \tl_set:Nn \l__nsk_connect_bend_type_tl {none} },
120 bend-type / single .code:n = { \tl_set:Nn \l__nsk_connect_bend_type_tl {single} },
121 bend-type / double .code:n = { \tl_set:Nn \l__nsk_connect_bend_type_tl {double} },
122 bend-type .initial:n = { none },
123
124 % bend direction ~~~~~ <<<
125 bend-direction .choice:,
126 bend-direction / up .code:n = { \tl_set:Nn \l__nsk_connect_bend_direction_tl { up } },
127 bend-direction / down .code:n = { \tl_set:Nn \l__nsk_connect_bend_direction_tl { down } },
128 bend-direction / left .code:n = { \tl_set:Nn \l__nsk_connect_bend_direction_tl { left } },
129 bend-direction / right .code:n = { \tl_set:Nn \l__nsk_connect_bend_direction_tl { right } },
130 bend-direction .initial:n = { up }, % e.g. default is "up"
131
132 % bend distance ~~~~~ <<<
133 bend-distance .dim_set:N = \l__nsk_connect_bend_amount_dim,
134 bend-distance .initial:n = {1cm},
135 bend-distance .default:n = {1cm},
136
137 % shorten ~~~~~ <<<
138 shorten-from .dim_set:N = \l__nsk_connect_shorten_from_dim,
139 shorten-from .initial:n = {0pt},
140 shorten-from .default:n = {0pt},
141
142 shorten-to .dim_set:N = \l__nsk_connect_shorten_to_dim,
143 shorten-to .initial:n = {0pt},

```

```

144 shorten-to .default:n = {0pt},
145
146 % ~~~~~
147 % Bend route
148 % convenience key that allows user to write something like
149 % bend-route = { up 1cm } or just up 1cm
150 % ~~~~~ <<<
151 bend-route .code:n =
152 {
153   % parse the #1 string, e.g. "up 1cm"
154   \__nsk_parse_bend_route:n {#1}
155 },
156
157 }
158
159 % ~~~~~
160 % Connect Drawing --
161 % Immediate draw if bridging is false:
162 % \nsk__draw_connect:VVVV
163 % #1 => arrow-style, #2 => color, #3 => id
164 % ~~~~~ <<<
165 cs_new_protected_nopar:Npn \nsk__draw_connect:VVVV #1#2#3#4
166 {
167   % if \l__nsk_connect_bend_bool is true, generate route accordingly:
168   \tl_if_eq:NnF \l__nsk_connect_bend_type_tl {none}
169   {
170     \__nsk_connect_generate_bend_route:
171   }
172
173   % \bool_if:NT \l__nsk_connect_bend_bool
174   % {
175   %   \__nsk_connect_generate_bend_route:
176   % }
177
178   % corner-radius ~~~~~ <<<
179   \tl_put_left:No \l__nsk_connect_route_tl
180   {[rounded~corners=\l__nsk_connect_corner_radius_dim]}
181
182   \expandafter\draw\expandafter[
183     #1, #2,
184     draw=#3,
185     spath/save~global=#4,
186     shorten~<=\l__nsk_connect_shorten_from_dim,
187     shorten~>=\l__nsk_connect_shorten_to_dim,
188   ]
189   ([xshift=\l__nsk_connect_shift_x_dim, yshift=\l__nsk_connect_shift_y_dim]\l__nsk_connect_from
190   \l__nsk_connect_route_tl
191   ([xshift=\l__nsk_connect_shift_x_dim, yshift=\l__nsk_connect_shift_y_dim]\l__nsk_connect_to_tl)
192 }
193
194 % ~~~~~
195 % Save a soft path if bridging is true:
196 % \nsk__spath_save:V
197 % #1 => path name (by value)

```

```

198 % ~~~~~ <<<
199 \cs_new_protected_nopar:Npn \nsk__spath_save:V #1
200 {
201   \iow_term:x {
202     =save=>#1
203     -from=\tl_use:N \l_nsk_connect_from_tl
204     -route=\tl_use:N \l_nsk_connect_route_tl
205     -to=\tl_use:N \l_nsk_connect_to_tl
206   }
207
208   % Just a simplified path from->to:
209   \path[
210     spath/save\space global=#1
211   ]
212   (\tl_use:N \l_nsk_connect_from_tl)
213   \tl_use:N \l_nsk_connect_route_tl
214   (\tl_use:N \l_nsk_connect_to_tl);
215 }
216
217 % ~~~~~
218 % Public Interface
219 % - (user-facing)
220 % ~~~~~ <<<
221 \NewDocumentCommand \nskConnect { 0{ } }
222 {
223   \group_begin:
224
225   % (a) parse user keys for "nsk/connect"
226   \keys_set:nn { nsk / connect } { #1 }
227
228   % (b) increment the global path counter
229   \int_gincr:N \g__nsk_connect_counter_int
230
231   % (c) build a default path name e.g. "path1"
232   \tl_set:Nx \l_tmpa_tl { path\int_use:N \g__nsk_connect_counter_int }
233
234   % (d) bridging check
235   \bool_if:NTF \l_nsk_connect_bridging_bool
236   {
237     % bridging == true => store path with spath
238     \nsk__spath_save:V \l_tmpa_tl
239
240     % then store bridging data for later
241     \__nsk_connect_store_item:nnnnnnnn
242     {
243       \tl_use:N \l_tmpa_tl
244     }
245     { \l_nsk_connect_arrow_style_tl      }
246     { \l_nsk_connect_arrow_tip_tl       }
247     { \l_nsk_connect_color_tl           }
248     { \l_nsk_connect_bridging_style_tl  }
249     { \l_nsk_connect_bridging_path_tl   }
250     { \l_nsk_connect_bridging_span_tl   }
251     { \l_nsk_connect_bridging_gap_tl    }

```

```

252 }
253 {
254 % bridging == false => draw it now
255 \nsk_draw_connect:VVVV
256 \l_nsk_connect_arrow_style_tl
257 \l_nsk_connect_arrow_tip_tl
258 \l_nsk_connect_color_tl
259 \l_nsk_connect_id_tl
260 }
261
262 \group_end:
263 }
264
265 % ~~~~~
266 % Finalize:
267 % Automagically bridging arcs over/under
268 % ~~~~~ <<<
269
270 % ~~~~~
271 % Path expansion
272 % Expand each path vs. each other path, bridging accordingly.
273 % 0^2
274 % ~~~~~ <<<
275 \cs_new_protected_nopar:Npn \just_expand:VV #1#2
276 {
277 \seq_map_inline:Nn #2
278 {
279 \iow_term:x {
280 =inner => (#1, ##1)
281 =style => \l_nsk_connect_bridging_style_tl
282 }
283
284 % local to each iteration
285 \tl_set:Nx \a_tl {#1}
286 \clist_set:No \args_dcs_clist {##1}
287 \tl_set:Nx \b_tl {\clist_item:Nn \args_dcs_clist {1}}
288 \tl_set:Nx \b_bridging_style_tl {\clist_item:Nn \args_dcs_clist {2}}
289
290 \clist_show:N \args_dcs_clist
291
292 % skip bridging with itself
293 \tl_if_eq:NNF \a_tl \b_tl
294 {
295 \iow_term:x {
296 difpaths: (\tl_use:N \a_tl, \tl_use:N \b_tl)
297 }
298
299 % bridging style => over or under
300 \tl_if_eq:NNF \l_nsk_connect_bridging_style_tl \b_bridging_style_tl
301 {
302
303 \tl_if_eq:NnTF \l_nsk_connect_bridging_style_tl {over}
304 {
305 \exp_args:NNx \tl_set:Nn \brd_tl

```

```

306     { bridge={\tl_use:N \a_tl}{\tl_use:N \b_tl} }
307   }
308 }
309 {
310   \exp_args:NNx \tl_set:Nn \brd_tl
311   { bridge={\tl_use:N \b_tl}{\tl_use:N \a_tl} }
312 }
313
314   \exp_args:Nx \tikzset{ \brd_tl }
315 }
316
317
318 }
319 }
320 }
321
322
323 % ~~~~~~
324 % Briding keys utils
325 % A small init macro for bridging keys
326 % ~~~~~~ <<<
327 \cs_new_protected_nopar:Npn \nsk__init_bridging_code:
328 {
329   \tikzset{
330     bridge/.style\space 2\space args={
331       spath/split\space at\space intersections\space with={##1}{##2},
332       spath/insert\space gaps\space after\space components={##1}{\pgfkeysvalueof{/tikz/bridging\space
333       spath/join\space components\space upright\space with={##1}{\pgfkeysvalueof{/tikz/bridging\space
334     }
335     spath/split\space at\space intersections\space with={##2}{##1},
336     spath/insert\space gaps\space after\space components={##2}{\pgfkeysvalueof{/tikz/bridging\space
337   },
338 }
339
340 \path[
341   spath/save\space global=arc
342 ]
343 (0,0)
344 arc[
345   radius=1cm,
346   start\space angle=180,
347   delta\space angle=-180
348 ];
349 }
350
351
352 % ~~~~~~
353 % Public Interface
354 % - (user-facing)
355 % ~~~~~~ <<<
356 \NewDocumentCommand \nskDoBridging { 0{ } }
357 {
358   \group_begin:
359

```



```

360 % Initialize bridging keys, and define an arc path globally
361 \nsk__init_bridging_code:
362
363 % For storing all path names
364 \seq_new:N \l__nsk_tmp_paths_seq
365
366 % Collect path names from bridging info
367 \seq_map_inline:Nn \g__nsk_bridging_info_seq
368 {
369   % {pathid, bridging-style}
370   \clist_set:Nn \l_tmpa_clist { ##1 }
371   \exp_args:NNx \seq_put_right:No \l__nsk_tmp_paths_seq
372   { \clist_item:Nn \l_tmpa_clist {1}, \clist_item:Nn \l_tmpa_clist {4} }
373 }
374
375 % Copy bridging info into a local seq, for iteration
376 \seq_set_eq:NN \l__nsk_tmp_seq \g__nsk_bridging_info_seq
377
378
379 % For each bridging path, set bridging keys, do bridging with others,
380 % then finally draw its path.
381 \seq_map_inline:Nn \l__nsk_tmp_seq
382 {
383   \clist_set:Nn \l_tmpa_clist { ##1 }
384
385   \tl_set:Nx \l__nsk_pathname_tl
386   { \clist_item:Nn \l_tmpa_clist {1} }
387   \tl_set:Nx \l__nsk_arrowstyle_tl
388   { \clist_item:Nn \l_tmpa_clist {2} }
389   \tl_set:Nx \l__nsk_color_tl
390   { \clist_item:Nn \l_tmpa_clist {3} }
391
392   \tl_set:Nx \l__nsk_connect_bridging_style_tl
393   { \clist_item:Nn \l_tmpa_clist {4} }
394
395   \tl_set:Nx \l__nsk_bridgingpath_tl
396   { \clist_item:Nn \l_tmpa_clist {5} }
397   \tl_set:Nx \l__nsk_bridgingspan_tl
398   { \clist_item:Nn \l_tmpa_clist {6} }
399   \tl_set:Nx \l__nsk_bridginggap_tl
400   { \clist_item:Nn \l_tmpa_clist {7} }
401
402   % Set bridging path, span, gap
403   \tikzset{
404     bridging\space path=\tl_use:N \l__nsk_bridgingpath_tl,
405     bridging\space span=\tl_use:N \l__nsk_bridgingspan_tl,
406     bridging\space gap=\tl_use:N \l__nsk_bridginggap_tl,
407   }
408
409   % do bridging over all other path names
410   \just_expand:VV
411   {\l__nsk_pathname_tl}
412   {\l__nsk_tmp_paths_seq}
413

```

```

414 % finally draw the path with color
415 \use:x {
416   \exp_not:N \draw[
417     \l__nsk_arrowstyle_tl,
418     spath/save~global=\l__nsk_connect_id_tl,
419     draw=\l__nsk_color_tl,
420     spath/use=\l__nsk_pathname_tl
421   ];
422 }
423 }
424
425 % Clear bridging items so not redrawn
426 \seq_clear:N \g__nsk_bridging_info_seq
427
428 \group_end:
429 }
430
431 % ~~~~~~
432 % Route Bend
433 % ~~~~~~ <<<
434 \cs_new_protected_nopar:Npn \__nsk_parse_bend_route:n #1
435 {
436   % Clear and split the input on spaces. E.g. "double up=10mm" -> "double", "up=10mm"
437   \seq_clear_new:N \l__nsk_tmp_seq
438   \seq_set_split:Nnn \l__nsk_tmp_seq { ~ } { #1 }
439
440   % Attempt to read the first token => single/double
441   \seq_if_empty:NF \l__nsk_tmp_seq
442   {
443     \seq_pop_left:NN \l__nsk_tmp_seq \l_tmpa_tl
444     % That should be "single" or "double"
445     \tl_set:Nx \l__nsk_connect_bend_type_tl \l_tmpa_tl
446   }
447
448   % Attempt to read the second token => "up=10mm", "down=1cm", or just "right", etc.
449   \seq_if_empty:NF \l__nsk_tmp_seq
450   {
451     \seq_pop_left:NN \l__nsk_tmp_seq \l_tmpb_tl
452     % Now check if there is an "=" inside the second token
453     \tl_if_in:VnTF \l_tmpb_tl { = }
454     {
455       % split e.g. "up=10mm" => "up" and "10mm"
456       \seq_clear_new:N \l__nsk_tmp_seqb
457       \seq_set_split:Nnx \l__nsk_tmp_seqb { = } { \l_tmpb_tl }
458
459       % the first part is direction, second is dimension
460       \seq_pop_left:NN \l__nsk_tmp_seqb \l_tmpe_tl % e.g. "up"
461       \seq_pop_left:NN \l__nsk_tmp_seqb \l_tmpe_tl % e.g. "10mm"
462
463       \tl_set:Nx \l__nsk_connect_bend_direction_tl \l_tmpe_tl
464       \dim_set:Nn \l__nsk_connect_bend_amount_dim {\l_tmpe_tl}
465     }
466   }
467 {

```

```

468 % no "=" => the token is direction alone, e.g. "down"
469 \tl_set_eq:NN \l__nsk_connect_bend_direction_tl \l_tmpb_tl
470 }
471 }
472 }
473
474
475 % ~~~~~
476 % Temporary dims for midpoint
477 % computation
478 % ~~~~~ <<<
479 \dim_new:N \l_cx_from_dim
480 \dim_new:N \l_cy_from_dim
481 \dim_new:N \l_cx_to_dim
482 \dim_new:N \l_cy_to_dim
483
484 % This sets \l_nsk_connect_route_tl to the standard ({ { distance -|} or (|- distance { {)
485 \cs_new_protected_nopar:Npn \__nsk_connect_generate_bend_route:
486 {
487   \tl_clear_new:N \l_nsk_connect_route_tl
488
489   % Copy bend-direction to a safe scratch var:
490   \tl_set_eq:NN \l_tmpa_tl \l__nsk_connect_bend_direction_tl
491
492   % Start building route
493   \tl_set:Nn \l_nsk_connect_route_tl { -- ++(0, }
494
495   % intermediary point ~~~~~ <<<
496   \path coordinate(from_pt) at (\l_nsk_connect_from_tl);
497   \path coordinate(to_pt) at (\l_nsk_connect_to_tl);
498
499   \pgfextractx{\l_cx_from_dim}{\pgfpointanchor{from_pt}{center}}
500   \pgfextracty{\l_cy_from_dim}{\pgfpointanchor{from_pt}{center}}
501
502   \pgfextractx{\l_cx_to_dim}{\pgfpointanchor{to_pt}{center}}
503   \pgfextracty{\l_cy_to_dim}{\pgfpointanchor{to_pt}{center}}
504
505
506   \str_case:Vn { \l__nsk_connect_bend_type_tl }
507   {
508
509     {single} {
510       \str_case:Vn { \l_tmpa_tl }
511       {
512         {up}
513         {
514           % (ax, by)
515           \tl_set:Nx \midpoint {
516             (\dim_use:N \l_cx_from_dim,\dim_use:N \l_cy_to_dim)
517           }
518
519           \tl_set:Nx \l_nsk_connect_route_tl
520           { -- \midpoint -- }
521         }

```

```

522     {down}
523     {
524         % (bx, ay)
525         \tl_set:Nx \midpoint {
526             (\dim_use:N \l_cx_to_dim,\dim_use:N \l_cy_from_dim)
527         }
528         \tl_set:Nx \l_nsk_connect_route_tl
529         { -- \midpoint -- }
530     }
531     % left and right don't make sense for single
532 }
533 }
534 {double}
535 {
536     % Now do a case on \l_tmpa_tl rather than raw \l__nsk_connect_bend_direction_tl
537     \str_case:Vn { \l_tmpa_tl }
538     {
539         {up}
540         { \tl_put_right:No \l_nsk_connect_route_tl
541             { \dim_use:N \l__nsk_connect_bend_amount_dim }}
542         \tl_put_right:No \l_nsk_connect_route_tl {
543             -|
544         }
545     }
546     {down}
547     { \tl_put_right:No \l_nsk_connect_route_tl
548         { -\dim_use:N \l__nsk_connect_bend_amount_dim ) -| } }
549     {left}
550     {
551         \tl_set:Nn \l_nsk_connect_route_tl { -- ++( }
552         \tl_put_right:No \l_nsk_connect_route_tl
553         { -\dim_use:N \l__nsk_connect_bend_amount_dim , 0 ) }
554         \tl_put_right:No \l_nsk_connect_route_tl {
555             [rounded~corners=\l__nsk_connect_corner_radius_dim]
556             |-
557         }
558     }
559     {right}
560     {
561         \tl_set:Nn \l_nsk_connect_route_tl { -- ++( }
562         \tl_put_right:No \l_nsk_connect_route_tl
563         { \dim_use:N \l__nsk_connect_bend_amount_dim , 0 ) |- }
564     }
565 }
566 {
567     % fallback => treat as "up"
568     \tl_put_right:No \l_nsk_connect_route_tl
569     { \dim_use:N \l__nsk_connect_bend_amount_dim ) -| }
570 }
571 }
572 }
573 }
574 }
575

```

```

576 % ~~~~~
577 % Main nsk / FillBetween Primitive
578 % ~~~~~ <<<
579 \keys_define:nn { nsk / fillbetween }
580 {
581   % end-points ~~~~~ <<<
582   from .tl_set:N = \l_nsk_fillbetween_from_tl,
583   from .value_required:n = true,
584
585   to .tl_set:N = \l_nsk_fillbetween_to_tl,
586   to .value_required:n = true,
587
588   % styles ~~~~~ <<<
589   fill .tl_set:N = \l_nsk_fillbetween_fill_tl,
590   fill .initial:n = {nskStrongBlue!20},
591
592   draw .tl_set:N = \l_nsk_fillbetween_draw_tl,
593   draw .initial:n = {nskStrongBlue!50},
594
595   opacity .fp_set:N = \l_nsk_fillbetween_opacity_fp,
596   opacity .initial:n = {0.4},
597
598   % edge-types ~~~~~ <<<
599   edge .choice:,
600   edge / straight .code:n = { \tl_set:Nn \l_nsk_fillbetween_edge_tl { straight } },
601   edge / curved .code:n = { \tl_set:Nn \l_nsk_fillbetween_edge_tl { curved } },
602   edge .initial:n = {straight},
603
604   % orientation ~~~~~ <<<
605   orientation .choice:,
606   orientation / vertical .code:n = { \tl_set:Nn \l_nsk_fillbetween_orient_tl { vertical } },
607   orientation / horizontal .code:n = { \tl_set:Nn \l_nsk_fillbetween_orient_tl { horizontal } },
608   orientation .initial:n = { vertical },
609
610   % overrides ~~~~~ <<<
611   % If you want an explicit \out=xx, in=yy" override, you could add:
612   out .tl_set:N = \l_nsk_fillbetween_out_tl,
613   out .initial:n = {},
614
615   in .tl_set:N = \l_nsk_fillbetween_in_tl,
616   in .initial:n = {},
617
618   % corner-offsets ~~~~~ <<<
619   corner-offset .dim_set:N = \l_nsk_fillbetween_corner_offset_dim,
620   corner-offset .initial:V = \g__nsk_style_block_border_radius_dim,
621   corner-offset .default:V = \g__nsk_style_block_border_radius_dim,
622
623 }
624
625 % ~~~~~
626 % Internal Drawing Utils
627 % ~~~~~ <<<
628 \dim_new:N \l__nsk_offset_west_dim
629 \dim_new:N \l__nsk_offset_east_dim

```

```

630 \dim_new:N \l__nsk_offset_north_dim
631 \dim_new:N \l__nsk_offset_south_dim
632
633 % ~~~~~~
634 % ~Internal FillBetween Draw
635 % ~~~~~~ <<<
636 \cs_new_protected_nopar:Npn \__nsk_fillbetween_draw:
637 {
638   % 2) Prepare local shift offsets for each corner direction:
639   %   By default, do no shift.
640   \dim_set:Nn \l__nsk_offset_west_dim { 0pt }
641   \dim_set:Nn \l__nsk_offset_east_dim { 0pt }
642   \dim_set:Nn \l__nsk_offset_north_dim { 0pt }
643   \dim_set:Nn \l__nsk_offset_south_dim { 0pt }
644
645   % If orientation=vertical
646   \str_if_eq:VnT { \l__nsk_fillbetween_orient_tl } { vertical }
647   {
648     \dim_set:Nn \l__nsk_offset_south_dim { \dim_use:N \l__nsk_fillbetween_corner_offset_dim }
649     \dim_set:Nn \l__nsk_offset_north_dim { -\dim_use:N \l__nsk_fillbetween_corner_offset_dim }
650   }
651
652   % If orientation=horizontal
653   \str_if_eq:VnT { \l__nsk_fillbetween_orient_tl } { horizontal }
654   {
655     \dim_set_eq:NN \l__nsk_offset_west_dim \l__nsk_fillbetween_corner_offset_dim
656     \dim_set:Nn \l__nsk_offset_east_dim { -\dim_use:N \l__nsk_fillbetween_corner_offset_dim }
657   }
658   % 1) Determine out/in angles if user wants curved edges but didn't provide them
659   \tl_if_eq:NnF \l__nsk_fillbetween_edge_tl \c_tl_empty_tl
660   {
661     \tl_if_eq:NnT \l__nsk_fillbetween_edge_tl {curved}
662     {
663       % if "curved" but user didn't specify \l__nsk_fillbetween_out_tl or in_tl
664       % we can set them based on orientation
665       \tl_if_blank:VTF \l__nsk_fillbetween_out_tl
666       {
667         \str_case:Vn { \l__nsk_fillbetween_orient_tl }
668         {
669           {vertical} { \tl_set:Nn \l__nsk_fillbetween_out_tl { 90 } }
670           {horizontal} { \tl_set:Nn \l__nsk_fillbetween_out_tl { 180 } }
671         }{}
672       }{}
673
674       \tl_if_blank:VTF \l__nsk_fillbetween_in_tl
675       {
676         \str_case:Vn { \l__nsk_fillbetween_orient_tl }
677         {
678           {vertical} { \tl_set:Nn \l__nsk_fillbetween_in_tl { -90 } }
679           {horizontal} { \tl_set:Nn \l__nsk_fillbetween_in_tl { 0 } }
680         }{}
681       }{}
682     }
683   }

```

```

684 % draw ~~~~~ <<<
685 \begin{pgfonlayer}{interim}
686 \draw[
687   spath/save~global=\tl_use:N \l_nsk_connect_id_tl,
688   fill=\l_nsk_fillbetween_fill_tl,
689   draw=\l_nsk_fillbetween_draw_tl,
690   opacity=\fp_use:N \l_nsk_fillbetween_opacity_fp
691 ]
692 let
693 % -- \From" corners --
694 \p{TL_a} = ($(\l_nsk_fillbetween_from_tl.north\space west)
695 + (\dim_use:N \l__nsk_offset_west_dim,
696 \dim_use:N \l__nsk_offset_north_dim)$),
697
698 \p{TR_a} = ($(\l_nsk_fillbetween_from_tl.north\space east)
699 + (\dim_use:N \l__nsk_offset_east_dim,
700 \dim_use:N \l__nsk_offset_north_dim)$),
701
702 \p{BL_a} = ($(\l_nsk_fillbetween_from_tl.south\space west)
703 + (\dim_use:N \l__nsk_offset_west_dim,
704 \dim_use:N \l__nsk_offset_south_dim)$),
705
706 \p{BR_a} = ($(\l_nsk_fillbetween_from_tl.south\space east)
707 + (\dim_use:N \l__nsk_offset_east_dim,
708 \dim_use:N \l__nsk_offset_south_dim)$),
709
710 % -- \To" corners --
711 \p{TL_b} = ($(\l_nsk_fillbetween_to_tl.north\space west)
712 + (\dim_use:N \l__nsk_offset_west_dim,
713 \dim_use:N \l__nsk_offset_north_dim)$),
714
715 \p{TR_b} = ($(\l_nsk_fillbetween_to_tl.north\space east)
716 + (\dim_use:N \l__nsk_offset_east_dim,
717 \dim_use:N \l__nsk_offset_north_dim)$),
718
719 \p{BL_b} = ($(\l_nsk_fillbetween_to_tl.south\space west)
720 + (\dim_use:N \l__nsk_offset_west_dim,
721 \dim_use:N \l__nsk_offset_south_dim)$),
722
723 \p{BR_b} = ($(\l_nsk_fillbetween_to_tl.south\space east)
724 + (\dim_use:N \l__nsk_offset_east_dim,
725 \dim_use:N \l__nsk_offset_south_dim)$)
726 in
727 \str_case:VnF { \l_nsk_fillbetween_edge_tl }
728 {
729   {straight}
730   {
731     \str_case:Vn { \l_nsk_fillbetween_orient_tl }
732     {
733       {vertical} {
734         (\p{BL_a}) -- (\p{TL_b}) -- (\p{TR_b}) -- (\p{BR_a})
735         -- cycle;
736       }
737       {horizontal} {

```

```

738         (\p{TR_a}) -- (\p{TL_b}) -- (\p{BL_b}) -- (\p{BR_a})
739         -- cycle;
740     }
741 }{}
742 }
743 {curved}
744 {
745     \str_case:Vn { \l_nsk_fillbetween_orient_tl }
746     {
747         {vertical} {
748             (\p{BL_a}) to[out=\tl_use:N \l_nsk_fillbetween_in_tl, in=\tl_use:N \l_nsk_fillbetween
749             -- (\p{TR_b}) to[out=\tl_use:N \l_nsk_fillbetween_out_tl, in=\tl_use:N \l_nsk_fillbet
750             -- cycle;
751         }
752         {horizontal} {
753             (\p{TR_a}) to[out=\tl_use:N \l_nsk_fillbetween_in_tl, in=\tl_use:N \l_nsk_fillbetween
754             -- (\p{BL_b}) to[out=\tl_use:N \l_nsk_fillbetween_out_tl, in=\tl_use:N \l_nsk_fillbet
755             -- cycle;
756         }
757     }{}
758 }
759 }
760 {
761     % TODO: you may wanna add some defaults here
762 }
763 \end{pgfonlayer}
764 }
765
766 % ~~~~~
767 % Public Interface
768 % - (user-facing)
769 % ~~~~~ <<<
770 \NewDocumentCommand \nskFillBetween { O{} }
771 {
772     \group_begin:
773     % 1. Parse the keys
774     \keys_set:nn {nsk / fillbetween}{#1}
775
776     % 2. Draw
777     \__nsk_fillbetween_draw:
778     \group_end:
779 }
780
781 % ~~~~~
782 % Angled Connect
783 % ~~~~~ <<<
784 \keys_define:nn { nsk/angles }
785 {
786     dir .choice:,
787     dir / left .code:n = { \tl_set:Nn \l__nsk_dir_tl {left} },
788     dir / right .code:n = { \tl_set:Nn \l__nsk_dir_tl {right} },
789     dir / up .code:n = { \tl_set:Nn \l__nsk_dir_tl {up} },
790     dir / down .code:n = { \tl_set:Nn \l__nsk_dir_tl {down} },
791

```



```

792 dir-align .default:n = {down},
793 dir-align .initial:n = {down},
794
795 alpha .fp_set:N = \l__nsk_alpha_fp
796 }
797
798 \NewDocumentCommand{\nskAngledc}{ 0{ } }
799 {
800   \keys_set:nn { nsk/angles } { #1 }
801
802
803   \tikzmath{
804     \da=\fp_use:N \l__nsk_alpha_fp;
805   }
806
807   \str_case:Nn \l__nsk_dir_tl
808   {
809     {down}{
810       \tikzmath{
811         \i = 180;
812         \o = -\da;
813         \ii = 0;
814         \oo = 180 + \da;
815       }
816     }
817     {up}{
818       \tikzmath{
819         \i = 180;
820         \o = \da;
821         \ii = 0;
822         \oo = 180 - \da;
823       }
824     }
825     {right}{
826       \tikzmath{
827         \i = 90;
828         \o = -\da;
829         \ii = -90;
830         \oo = \da;
831       }
832     }
833     {left}{
834       \tikzmath{
835         \i = 90;
836         \o = 180 + \da;
837         \ii = -90;
838         \oo = 180 - \da;
839       }
840     }
841
842   }
843 }
844
845 \ExplSyntaxOff

```

i/package;