iii

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 ¡¡¡ ¡*package¿

```
\ExplSyntaxOn

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Global Data Structures
% A counter to generate unique path names.
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\int_new:N \g__nsk_connect_counter_int

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% A sequence storing bridging info for each path:
% Each item is a comma list:
%   pathName, arrowStyle, color, bridgingStyle, bridgingPath,
%   bridgingSpan, bridgingGap
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\seq_new:N \g__nsk_bridging_info_seq


% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Storing One Connect Item
% - (path name, arrow style, color, bridging style, bridging path,
%  bridging span, bridging gap)
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\cs_new_protected_nopar:Npn \__nsk_connect_store_item:nnnnnnnn #1#2#3#4#5#6#7#8
{
 % We'll build a single token list item, then push it into
 % the bridging sequence.
 \tl_clear_new:N \l__nsk_storeitem_tl

 % Insert #1..#7 by expansion (x), forming a comma list:
 \exp_args:NNx \tl_put_right:Nn \l__nsk_storeitem_tl
 { #1, #2, { #3 }, #4, #5, #6, #7, #8 }

 % Now append that item to the bridging sequence:
 \seq_gput_right:No \g__nsk_bridging_info_seq \l__nsk_storeitem_tl

```

```
 % Debug lines (currently commented out):
 \seq_show:N \g__nsk_bridging_info_seq
 \tl_show:N  \l__nsk_storeitem_tl
}



% build/tracker ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
% A boolean to track bridging = true/false
\bool_new:N \l__nsk_connect_bridging_bool

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Main nsk / connect Primitive
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\keys_define:nn { nsk / connect }
{
 % ref-id ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 id .tl_set:N = \l_nsk_connect_id_tl,
 id .initial:n = { },
 id .default:n = { },

 % end-points ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 from .tl_set:N           = \l_nsk_connect_from_tl,
 from .value_required:n   = true,

 to .tl_set:N             = \l_nsk_connect_to_tl,
 to .value_required:n     = true,

 route .tl_set:N          = \l_nsk_connect_route_tl,
 route .initial:n         = { -- },

 % bridging ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 bridging .choice:,
 bridging / true  .code:n = { \bool_set_true:N  \l__nsk_connect_bridging_bool },
 bridging / false .code:n = { \bool_set_false:N \l__nsk_connect_bridging_bool },
 bridging .initial:n      = { false },

 bridging-style .choice:,
 bridging-style / over  .code:n  = { \tl_set:Nn \l_nsk_connect_bridging_style_tl { over } },
 bridging-style / under .code:n  = { \tl_set:Nn \l_nsk_connect_bridging_style_tl { under } },
 bridging-style .initial:n       = { under },
 bridging-style .default:n       = { under },

 bridging-path .tl_set:N         = \l_nsk_connect_bridging_path_tl,
 bridging-path .initial:n        = { arc },
 bridging-path .default:n        = { arc },

 bridging-span .tl_set:N         = \l_nsk_connect_bridging_span_tl,
 bridging-span .initial:n        = { 8pt },
 bridging-span .default:n        = { 8pt },

 bridging-gap .tl_set:N          = \l_nsk_connect_bridging_gap_tl,
 bridging-gap .initial:n         = { 4pt },
```

```
bridging-gap .default:n         = { 4pt },

arrow-style .tl_set:N = \l_nsk_connect_arrow_style_tl,
arrow-style .initial:n = {ultra~thick, line~cap=round},
arrow-style .default:n = {ultra~thick, line~cap=round},

arrow-tip .tl_set:N = \l_nsk_connect_arrow_tip_tl,
arrow-tip .initial:n = {-},
arrow-tip .default:n = {-},

shift-x .dim_set:N = \l_nsk_connect_shift_x_dim,
shift-x .initial:n = {0mm},
shift-x .default:n = {0mm},

shift-y .dim_set:N = \l_nsk_connect_shift_y_dim,
shift-y .initial:n = {0mm},
shift-y .default:n = {0mm},

color .tl_set:N                 = \l_nsk_connect_color_tl,
color .initial:n                = { c__nsk_principal },
color .default:n                = { c__nsk_principal },

% corner-radius ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
corner-radius .dim_set:N = \l__nsk_connect_corner_radius_dim,
corner-radius .initial:n = { 3mm },
corner-radius .default:n = { 3mm },

% bend-routing ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
bend-type .choice:,
bend-type / none   .code:n = { \tl_set:Nn \l__nsk_connect_bend_type_tl {none} },
bend-type / single .code:n = { \tl_set:Nn \l__nsk_connect_bend_type_tl {single} },
bend-type / double  .code:n = { \tl_set:Nn \l__nsk_connect_bend_type_tl {double} },
bend-type .initial:n = { none },

% bend direction ~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
bend-direction .choice:,
bend-direction / up    .code:n = { \tl_set:Nn \l__nsk_connect_bend_direction_tl { up } },
bend-direction / down  .code:n = { \tl_set:Nn \l__nsk_connect_bend_direction_tl { down } },
bend-direction / left  .code:n = { \tl_set:Nn \l__nsk_connect_bend_direction_tl { left } },
bend-direction / right .code:n = { \tl_set:Nn \l__nsk_connect_bend_direction_tl { right } },
bend-direction .initial:n = { up }, % e.g. default is "up"

% bend distance ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
bend-distance .dim_set:N = \l__nsk_connect_bend_amount_dim,
bend-distance .initial:n  = {1cm},
bend-distance .default:n  = {1cm},

% shorten ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
shorten-from .dim_set:N = \l__nsk_connect_shorten_from_dim,
shorten-from .initial:n = {0pt},
shorten-from .default:n = {0pt},

shorten-to .dim_set:N = \l__nsk_connect_shorten_to_dim,
shorten-to .initial:n = {0pt},
```

```
 shorten-to .default:n = {0pt},

 % ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 % Bend route
 % convenience key that allows user to write something like
 % bend-route = { up 1cm } or just up 1cm
 % ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 bend-route .code:n =
  {
   % parse the #1 string, e.g. "up 1cm"
   \__nsk_parse_bend_route:n {#1}
  },

}

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Connect Drawing --
%  Immediate draw if bridging is false:
%   \nsk__draw_connect:VVVV
%  #1 => arrow-style, #2 => color, #3 => id
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\cs_new_protected_nopar:Npn \nsk__draw_connect:VVVV #1#2#3#4
{
 % if \l__nsk_connect_bend_bool is true, generate route accordingly:
 \tl_if_eq:NnF \l__nsk_connect_bend_type_tl {none}
 {
  \__nsk_connect_generate_bend_route:
 }

 % \bool_if:NT \l__nsk_connect_bend_bool
 % {
 %  \__nsk_connect_generate_bend_route:
 % }

 % corner-radius ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 \tl_put_left:No \l_nsk_connect_route_tl
 {[rounded~corners=\l__nsk_connect_corner_radius_dim]}

 \expandafter\draw\expandafter[
  #1, #2,
  draw=#3,
  spath/save~global=#4,
  shorten~<=\l__nsk_connect_shorten_from_dim,
  shorten~>=\l__nsk_connect_shorten_to_dim,
 ]
  ([xshift=\l_nsk_connect_shift_x_dim, yshift=\l_nsk_connect_shift_y_dim]\l_nsk_connect_from_
 \l_nsk_connect_route_tl
 ([xshift=\l_nsk_connect_shift_x_dim, yshift=\l_nsk_connect_shift_y_dim]\l_nsk_connect_to_tl)
}

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Save a soft path if bridging is true:
%   \nsk__spath_save:V
% #1 => path name (by value)
```

```
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\cs_new_protected_nopar:Npn \nsk__spath_save:V #1
{
 \iow_term:x {
  =save=>#1
  -from=\tl_use:N \l_nsk_connect_from_tl
  -route=\tl_use:N \l_nsk_connect_route_tl
  -to=\tl_use:N \l_nsk_connect_to_tl
 }

 % Just a simplified path from->to:
 \path[
  spath/save\space global=#1
 ]
 (\tl_use:N \l_nsk_connect_from_tl)
 \tl_use:N \l_nsk_connect_route_tl
 (\tl_use:N \l_nsk_connect_to_tl);
}

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Public Interface
% - (user-facing)
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\NewDocumentCommand \nskConnect { O{} }
{
 \group_begin:

 % (a) parse user keys for "nsk/connect"
 \keys_set:nn { nsk / connect } { #1 }

 % (b) increment the global path counter
 \int_gincr:N \g__nsk_connect_counter_int

 % (c) build a default path name e.g. "path1"
 \tl_set:Nx \l_tmpa_tl { path\int_use:N \g__nsk_connect_counter_int }

 % (d) bridging check
 \bool_if:NTF \l__nsk_connect_bridging_bool
 {
  % bridging == true => store path with spath
  \nsk__spath_save:V \l_tmpa_tl

  % then store bridging data for later
  \__nsk_connect_store_item:nnnnnnnn
  {
   \tl_use:N \l_tmpa_tl
  }
  { \l_nsk_connect_arrow_style_tl      }
  { \l_nsk_connect_arrow_tip_tl      }
  { \l_nsk_connect_color_tl            }
  { \l_nsk_connect_bridging_style_tl   }
  { \l_nsk_connect_bridging_path_tl    }
  { \l_nsk_connect_bridging_span_tl    }
  { \l_nsk_connect_bridging_gap_tl     }
```

```
 }
 {
  % bridging == false => draw it now
  \nsk__draw_connect:VVVV
  \l_nsk_connect_arrow_style_tl
  \l_nsk_connect_arrow_tip_tl
  \l_nsk_connect_color_tl
  \l_nsk_connect_id_tl
 }

 \group_end:
}

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Finalize:
% Automagically bridging arcs over/under
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Path expansion
% Expand each path vs. each other path, bridging accordingly.
% O^2
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\cs_new_protected_nopar:Npn \just_expand:VV #1#2
{
 \seq_map_inline:Nn #2
 {
  \iow_term:x {
   =inner => (#1, ##1)
   =style => \l_nsk_connect_bridging_style_tl
  }

  % local to each iteration
  \tl_set:Nx \a_tl {#1}
  \clist_set:No \args_dcs_clist {##1}
  \tl_set:Nx \b_tl {\clist_item:Nn \args_dcs_clist {1}}
  \tl_set:Nx \b_bridging_style_tl {\clist_item:Nn \args_dcs_clist {2}}

  \clist_show:N \args_dcs_clist

  % skip bridging with itself
  \tl_if_eq:NNF \a_tl \b_tl
  {
   \iow_term:x {
    difpaths: (\tl_use:N \a_tl, \tl_use:N \b_tl)
   }

   % bridging style => over or under
   \tl_if_eq:NNF \l_nsk_connect_bridging_style_tl \b_bridging_style_tl
   {

    \tl_if_eq:NnTF \l_nsk_connect_bridging_style_tl {over}
    {
     \exp_args:NNx \tl_set:Nn \brd_tl
```

```
     { bridge={\tl_use:N \a_tl}{\tl_use:N \b_tl} }

    }
    {
     \exp_args:NNx \tl_set:Nn \brd_tl
     { bridge={\tl_use:N \b_tl}{\tl_use:N \a_tl} }
    }

    \exp_args:Nx \tikzset{ \brd_tl }
   }


  }
 }
}


% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Briding keys utils
% A small init macro for bridging keys
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\cs_new_protected_nopar:Npn \nsk__init_bridging_code:
{
\tikzset{
bridge/.style\space 2\space args={
spath/split\space at\space intersections\space with={##1}{##2},
spath/insert\space gaps\space after\space components={##1}{\pgfkeysvalueof{/tikz/bridging\spa
spath/join\space components\space upright\space with={##1}{\pgfkeysvalueof{/tikz/bridging\spa

spath/split\space at\space intersections\space with={##2}{##1},
spath/insert\space gaps\space after\space components={##2}{\pgfkeysvalueof{/tikz/bridging\spa
},
}

\path[
 spath/save\space global=arc
]
(0,0)
arc[
  radius=1cm,
  start\space angle=180,
  delta\space angle=-180
 ];
}


% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Public Interface
% - (user-facing)
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\NewDocumentCommand \nskDoBridging { O{} }
{
 \group_begin:

```

```
% Initialize bridging keys, and define an arc path globally
\nsk__init_bridging_code:

% For storing all path names
\seq_new:N \l__nsk_tmp_paths_seq

% Collect path names from bridging info
\seq_map_inline:Nn \g__nsk_bridging_info_seq
{
 % {pathid, bridging-style}
 \clist_set:Nn \l_tmpa_clist { ##1 }
 \exp_args:NNx \seq_put_right:No \l__nsk_tmp_paths_seq
 { \clist_item:Nn \l_tmpa_clist {1}, \clist_item:Nn \l_tmpa_clist {4} }
}

% Copy bridging info into a local seq, for iteration
\seq_set_eq:NN \l__nsk_tmp_seq \g__nsk_bridging_info_seq


% For each bridging path, set bridging keys, do bridging with others,
% then finally draw its path.
\seq_map_inline:Nn \l__nsk_tmp_seq
{
 \clist_set:Nn \l_tmpa_clist { ##1 }

 \tl_set:Nx \l__nsk_pathname_tl
 { \clist_item:Nn \l_tmpa_clist {1} }
 \tl_set:Nx \l__nsk_arrowstyle_tl
 { \clist_item:Nn \l_tmpa_clist {2} }
 \tl_set:Nx \l__nsk_color_tl
 { \clist_item:Nn \l_tmpa_clist {3} }

 \tl_set:Nx \l_nsk_connect_bridging_style_tl
 { \clist_item:Nn \l_tmpa_clist {4} }

 \tl_set:Nx \l__nsk_bridgingpath_tl
 { \clist_item:Nn \l_tmpa_clist {5} }
 \tl_set:Nx \l__nsk_bridgingspan_tl
 { \clist_item:Nn \l_tmpa_clist {6} }
 \tl_set:Nx \l__nsk_bridginggap_tl
 { \clist_item:Nn \l_tmpa_clist {7} }

 % Set bridging path, span, gap
 \tikzset{
  bridging\space path=\tl_use:N \l__nsk_bridgingpath_tl,
  bridging\space span=\tl_use:N \l__nsk_bridgingspan_tl,
  bridging\space gap=\tl_use:N \l__nsk_bridginggap_tl,
 }

 % do bridging over all other path names
 \just_expand:VV
 {\l__nsk_pathname_tl}
 {\l__nsk_tmp_paths_seq}

```

```
  % finally draw the path with color
  \use:x {
   \exp_not:N \draw[
    \l__nsk_arrowstyle_tl,
    spath/save~global=\l_nsk_connect_id_tl,
    draw=\l__nsk_color_tl,
    spath/use=\l__nsk_pathname_tl
   ];
  }
 }

 % Clear bridging items so not redrawn
 \seq_clear:N \g__nsk_bridging_info_seq

 \group_end:
}

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Route Bend
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\cs_new_protected_nopar:Npn \__nsk_parse_bend_route:n #1
{
 % Clear and split the input on spaces. E.g. "double up=10mm" -> "double", "up=10mm"
 \seq_clear_new:N \l__nsk_tmp_seq
 \seq_set_split:Nnn \l__nsk_tmp_seq { ~ } { #1 }

 % Attempt to read the first token => single/double
 \seq_if_empty:NF \l__nsk_tmp_seq
 {
  \seq_pop_left:NN \l__nsk_tmp_seq \l_tmpa_tl
  % That should be "single" or "double"
  \tl_set:Nx \l__nsk_connect_bend_type_tl \l_tmpa_tl
 }

 % Attempt to read the second token => "up=10mm", "down=1cm", or just "right", etc.
 \seq_if_empty:NF \l__nsk_tmp_seq
 {
  \seq_pop_left:NN \l__nsk_tmp_seq \l_tmpb_tl
  % Now check if there is an "=" inside the second token
  \tl_if_in:VnTF \l_tmpb_tl { = }
  {
   % split e.g. "up=10mm" => "up" and "10mm"
   \seq_clear_new:N \l__nsk_tmp_seqb
   \seq_set_split:Nnx \l__nsk_tmp_seqb { = } { \l_tmpb_tl }


   % the first part is direction, second is dimension
   \seq_pop_left:NN \l__nsk_tmp_seqb \l_tmpc_tl % e.g. "up"
   \seq_pop_left:NN \l__nsk_tmp_seqb \l_tmpd_tl % e.g. "10mm"

   \tl_set:Nx \l__nsk_connect_bend_direction_tl \l_tmpc_tl
   \dim_set:Nn \l__nsk_connect_bend_amount_dim {\l_tmpd_tl}
  }
  {
```

```
   % no "=" => the token is direction alone, e.g. "down"
   \tl_set_eq:NN \l__nsk_connect_bend_direction_tl \l_tmpb_tl
  }
 }
}


% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Temporary dims for midpoint
% computation
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\dim_new:N \l_cx_from_dim
\dim_new:N \l_cy_from_dim
\dim_new:N \l_cx_to_dim
\dim_new:N \l_cy_to_dim

% This sets \l_nsk_connect_route_tl to the standard ({ { distance -|) or (|- distance { {)
\cs_new_protected_nopar:Npn \__nsk_connect_generate_bend_route:
{
 \tl_clear_new:N \l_nsk_connect_route_tl

 % Copy bend-direction to a safe scratch var:
 \tl_set_eq:NN \l_tmpa_tl \l__nsk_connect_bend_direction_tl

 % Start building route
 \tl_set:Nn \l_nsk_connect_route_tl { -- ++(0, }

 % intermediary point ~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 \path coordinate(from_pt) at (\l_nsk_connect_from_tl);
 \path coordinate(to_pt) at (\l_nsk_connect_to_tl);

 \pgfextractx{\l_cx_from_dim}{\pgfpointanchor{from_pt}{center}}
 \pgfextracty{\l_cy_from_dim}{\pgfpointanchor{from_pt}{center}}

 \pgfextractx{\l_cx_to_dim}{\pgfpointanchor{to_pt}{center}}
 \pgfextracty{\l_cy_to_dim}{\pgfpointanchor{to_pt}{center}}


 \str_case:Vn { \l__nsk_connect_bend_type_tl }
 {

  {single} {
    \str_case:Vn { \l_tmpa_tl }
    {
     {up}
      {
       % (ax, by)
       \tl_set:Nx \midpoint {
        (\dim_use:N \l_cx_from_dim,\dim_use:N \l_cy_to_dim)
       }

       \tl_set:Nx \l_nsk_connect_route_tl
       { -- \midpoint -- }
      }
```

```
      {down}
      {
       % (bx, ay)
       \tl_set:Nx \midpoint {
        (\dim_use:N \l_cx_to_dim,\dim_use:N \l_cy_from_dim)
       }
       \tl_set:Nx \l_nsk_connect_route_tl
       { -- \midpoint -- }
      }
     % left and right don't make sense for single
    }
   }
   {double}
   {
    % Now do a case on \l_tmpa_tl rather than raw \l__nsk_connect_bend_direction_tl
    \str_case:Vn { \l_tmpa_tl }
    {
     {up}
      { \tl_put_right:No \l_nsk_connect_route_tl
       { \dim_use:N \l__nsk_connect_bend_amount_dim )}
       \tl_put_right:No \l_nsk_connect_route_tl {
        -|
       }
      }
      {down}
      { \tl_put_right:No \l_nsk_connect_route_tl
       { -\dim_use:N \l__nsk_connect_bend_amount_dim ) -| } }
      {left}
      {
       \tl_set:Nn \l_nsk_connect_route_tl { -- ++( }
       \tl_put_right:No \l_nsk_connect_route_tl
       { -\dim_use:N \l__nsk_connect_bend_amount_dim , 0 )  }
       \tl_put_right:No \l_nsk_connect_route_tl {
        [rounded~corners=\l__nsk_connect_corner_radius_dim]
        |-
       }
      }
      {right}
      {
       \tl_set:Nn \l_nsk_connect_route_tl { -- ++( }
       \tl_put_right:No \l_nsk_connect_route_tl
       { \dim_use:N \l__nsk_connect_bend_amount_dim , 0 ) |- }
      }
    }
    {
     % fallback => treat as "up"
     \tl_put_right:No \l_nsk_connect_route_tl
     { \dim_use:N \l__nsk_connect_bend_amount_dim ) -| }
    }
   }
 }

}
```

```
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Main nsk / FillBetween Primitive
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\keys_define:nn { nsk / fillbetween }
{
 % end-points ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 from .tl_set:N = \l_nsk_fillbetween_from_tl,
 from .value_required:n  = true,

 to   .tl_set:N = \l_nsk_fillbetween_to_tl,
 to   .value_required:n   = true,

 % styles ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 fill .tl_set:N = \l_nsk_fillbetween_fill_tl,
 fill .initial:n = {nskStrongBlue!20},

 draw .tl_set:N = \l_nsk_fillbetween_draw_tl,
 draw .initial:n = {nskStrongBlue!50},

 opacity .fp_set:N = \l_nsk_fillbetween_opacity_fp,
 opacity .initial:n = {0.4},

 % edge-types ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 edge .choice:,
 edge / straight .code:n = { \tl_set:Nn \l_nsk_fillbetween_edge_tl { straight } },
 edge / curved  .code:n  = { \tl_set:Nn \l_nsk_fillbetween_edge_tl { curved } },
 edge .initial:n = {straight},

 % orientation ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 orientation .choice:,
 orientation / vertical   .code:n = { \tl_set:Nn \l_nsk_fillbetween_orient_tl { vertical } },
 orientation / horizontal .code:n = { \tl_set:Nn \l_nsk_fillbetween_orient_tl { horizontal }
 orientation .initial:n   = { vertical },

 % overrides ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 % If you want an explicit \out=xx, in=yy" override, you could add:
 out .tl_set:N = \l_nsk_fillbetween_out_tl,
 out .initial:n = {},

 in .tl_set:N  = \l_nsk_fillbetween_in_tl,
 in .initial:n = {},

 % corner-offsets ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 corner-offset .dim_set:N = \l_nsk_fillbetween_corner_offset_dim,
 corner-offset .initial:V = \g__nsk_style_block_border_radius_dim,
 corner-offset .default:V = \g__nsk_style_block_border_radius_dim,

}

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Internal Drawing Utils
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\dim_new:N \l__nsk_offset_west_dim
\dim_new:N \l__nsk_offset_east_dim
```

```
\dim_new:N \l__nsk_offset_north_dim
\dim_new:N \l__nsk_offset_south_dim

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% ~Internal FillBetween Draw
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\cs_new_protected_nopar:Npn \__nsk_fillbetween_draw:
{
 % 2) Prepare local shift offsets for each corner direction:
 %    By default, do no shift.
 \dim_set:Nn \l__nsk_offset_west_dim  { 0pt }
 \dim_set:Nn \l__nsk_offset_east_dim  { 0pt }
 \dim_set:Nn \l__nsk_offset_north_dim { 0pt }
 \dim_set:Nn \l__nsk_offset_south_dim { 0pt }

 % If orientation=vertical
 \str_if_eq:VnT { \l_nsk_fillbetween_orient_tl } { vertical }
 {
  \dim_set:Nn    \l__nsk_offset_south_dim { \dim_use:N \l_nsk_fillbetween_corner_offset_dim }
  \dim_set:Nn    \l__nsk_offset_north_dim { -\dim_use:N \l_nsk_fillbetween_corner_offset_dim
 }

 % If orientation=horizontal
 \str_if_eq:VnT { \l_nsk_fillbetween_orient_tl } { horizontal }
 {
  \dim_set_eq:NN \l__nsk_offset_west_dim  \l_nsk_fillbetween_corner_offset_dim
  \dim_set:Nn    \l__nsk_offset_east_dim { -\dim_use:N \l_nsk_fillbetween_corner_offset_dim }
 }
 % 1) Determine out/in angles if user wants curved edges but didn't provide them
 \tl_if_eq:NNF \l_nsk_fillbetween_edge_tl \c_tl_empty_tl
 {
  \tl_if_eq:NnT \l_nsk_fillbetween_edge_tl {curved}
  {
   % if "curved" but user didn't specify \l_nsk_fillbetween_out_tl or in_tl
   % we can set them based on orientation
   \tl_if_blank:VTF \l_nsk_fillbetween_out_tl
   {
    \str_case:Vn { \l_nsk_fillbetween_orient_tl }
    {
     {vertical}   { \tl_set:Nn \l_nsk_fillbetween_out_tl { 90 } }
      {horizontal} { \tl_set:Nn \l_nsk_fillbetween_out_tl {   180 } }
    }{}
   }{}

   \tl_if_blank:VTF \l_nsk_fillbetween_in_tl
   {
    \str_case:Vn { \l_nsk_fillbetween_orient_tl }
    {
     {vertical}   { \tl_set:Nn \l_nsk_fillbetween_in_tl  { -90 } }
      {horizontal} { \tl_set:Nn \l_nsk_fillbetween_in_tl  { 0 } }
    }{}
   }{}
  }
}
```

```
% draw ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\begin{pgfonlayer}{interim}
 \draw[
  spath/save~global=\tl_use:N \l_nsk_connect_id_tl,
  fill=\l_nsk_fillbetween_fill_tl,
  draw=\l_nsk_fillbetween_draw_tl,
  opacity=\fp_use:N \l_nsk_fillbetween_opacity_fp
 ]
 let
 % -- \From" corners --
 \p{TL_a} = ($(\l_nsk_fillbetween_from_tl.north\space west)
  + (\dim_use:N \l__nsk_offset_west_dim,
  \dim_use:N \l__nsk_offset_north_dim)$),

 \p{TR_a} = ($(\l_nsk_fillbetween_from_tl.north\space east)
  + (\dim_use:N \l__nsk_offset_east_dim,
  \dim_use:N \l__nsk_offset_north_dim)$),

 \p{BL_a} = ($(\l_nsk_fillbetween_from_tl.south\space west)
  + (\dim_use:N \l__nsk_offset_west_dim,
  \dim_use:N \l__nsk_offset_south_dim)$),

 \p{BR_a} = ($(\l_nsk_fillbetween_from_tl.south\space east)
  + (\dim_use:N \l__nsk_offset_east_dim,
  \dim_use:N \l__nsk_offset_south_dim)$),

 % -- \To" corners --
 \p{TL_b} = ($(\l_nsk_fillbetween_to_tl.north\space west)
  + (\dim_use:N \l__nsk_offset_west_dim,
  \dim_use:N \l__nsk_offset_north_dim)$),

 \p{TR_b} = ($(\l_nsk_fillbetween_to_tl.north\space east)
  + (\dim_use:N \l__nsk_offset_east_dim,
  \dim_use:N \l__nsk_offset_north_dim)$),

 \p{BL_b} = ($(\l_nsk_fillbetween_to_tl.south\space west)
  + (\dim_use:N \l__nsk_offset_west_dim,
  \dim_use:N \l__nsk_offset_south_dim)$),

 \p{BR_b} = ($(\l_nsk_fillbetween_to_tl.south\space east)
  + (\dim_use:N \l__nsk_offset_east_dim,
  \dim_use:N \l__nsk_offset_south_dim)$)
 in
 \str_case:VnF { \l_nsk_fillbetween_edge_tl }
 {
  {straight}
   {
    \str_case:Vn { \l_nsk_fillbetween_orient_tl }
    {
     {vertical}   {
       (\p{BL_a}) -- (\p{TL_b}) -- (\p{TR_b}) -- (\p{BR_a})
       -- cycle;
      }
      {horizontal} {
```

```
        (\p{TR_a}) -- (\p{TL_b}) -- (\p{BL_b}) -- (\p{BR_a})
        -- cycle;
       }
     }{}
    }
    {curved}
    {
     \str_case:Vn { \l_nsk_fillbetween_orient_tl }
     {
      {vertical}   {
        (\p{BL_a}) to[out=\tl_use:N \l_nsk_fillbetween_in_tl, in=\tl_use:N \l_nsk_fillbetween
        -- (\p{TR_b}) to[out=\tl_use:N \l_nsk_fillbetween_out_tl, in=\tl_use:N \l_nsk_fillbet
        -- cycle;
       }
       {horizontal} {
        (\p{TR_a}) to[out=\tl_use:N \l_nsk_fillbetween_in_tl, in=\tl_use:N \l_nsk_fillbetween
        -- (\p{BL_b}) to[out=\tl_use:N \l_nsk_fillbetween_out_tl, in=\tl_use:N \l_nsk_fillbet
        -- cycle;
       }
     }{}
    }
  }
  {
   % TODO: you may wanna add some defaults here
  }
 \end{pgfonlayer}
}

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Public Interface
% - (user-facing)
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\NewDocumentCommand \nskFillBetween { O{} }
{
 \group_begin:
 % 1. Parse the keys
 \keys_set:nn {nsk / fillbetween}{#1}

 % 2. Draw
 \__nsk_fillbetween_draw:
 \group_end:
}

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Angled Connect
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\keys_define:nn { nsk/angles }
{
 dir .choice:,
 dir / left .code:n = { \tl_set:Nn \l__nsk_dir_tl {left} },
 dir / right .code:n = { \tl_set:Nn \l__nsk_dir_tl {right} },
 dir / up .code:n = { \tl_set:Nn \l__nsk_dir_tl {up} },
 dir / down .code:n = { \tl_set:Nn \l__nsk_dir_tl {down} },

```

```
 dir-align .default:n = {down},
 dir-align .initial:n = {down},

 alpha .fp_set:N = \l__nsk_alpha_fp
}

\NewDocumentCommand{\nskAngledc}{ O{} }
{
 \keys_set:nn { nsk/angles } { #1 }


 \tikzmath{
  \da=\fp_use:N \l__nsk_alpha_fp;
 }

 \str_case:Nn \l__nsk_dir_tl
 {
  {down}{
    \tikzmath{
     \i = 180;
     \o = -\da;
     \ii = 0;
     \oo = 180 + \da;
    }
   }
   {up}{
    \tikzmath{
     \i = 180;
     \o = \da;
     \ii = 0;
     \oo = 180 - \da;
    }
   }
   {right}{
    \tikzmath{
     \i = 90;
     \o = -\da;
     \ii = -90;
     \oo = \da;
    }
   }
   {left}{
    \tikzmath{
     \i = 90;
     \o = 180 + \da;
     \ii = -90;
     \oo = 180 - \da;
    }
   }

 }
}

\ExplSyntaxOff
```

¡/package¿