iii

neural-sketch.sty (main code)

Vincenzo Buono

March 22, 2025

implementation ¡¡¡ ¡*package¿

```
1
2 % external ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
3 \RequirePackage{tikz}
4 \RequirePackage{expl3}
5 \usetikzlibrary{
6  calc, fit, intersections, spath3,
7  shapes.geometric,
8  shapes.symbols,
9  shapes.misc,
10  arrows.meta,
11  positioning,
12  arrows.meta,
13  shadows,
14  shadings,
15  patterns,
16  decorations.markings,
17  backgrounds,
18  tikzmark,
19  math,
20 }
21
22
23 \ExplSyntaxOn
24
25 % ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
26 % Pkg opts
27 % ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
28 \seq_new:N \l__nsk_pkgopts_opts_seq
29 \keys_define:nn { nsk / pkgopts }
30 {
31   unknown .code:n = {
32     \seq_put_right:Nn \l__nsk_pkgopts_opts_seq {##1~={##2},}
33   }
34 }
35 \ProcessKeysOptions { nsk / pkgopts }
36
37 \bool_new:N \g__nsk_dark_mode_bool
38 \keys_set:nn { nsk / style } { \seq_use:Nn \l__nsk_pkgopts_opts_seq {, } }
39
40 \ExplSyntaxOff
41
```

```
% internal ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\RequirePackage{neural-sketch-colors}
\RequirePackage{neural-sketch-styles}
\RequirePackage{neural-sketch-loader}

\ExplSyntaxOn
% \bool_if:NTF \g__nsk_dark_mode_bool
% {
%  \definecolor{ c__nsk_dark_bg_clr }      {HTML}{1d2021}
%  \definecolor{ c__nsk_principal } {HTML}{ebdbb2}
%  \pagecolor{c__nsk_dark_bg_clr}
%  \colorlet{nskBg}{c__nsk_dark_bg_clr}
%  \colorlet{nskFg}{c__nsk_principal}
% }
% {
%  \definecolor{ c__nsk_principal } {HTML}{000000}
%  \colorlet{nskBg}{c__nsk_principal}
%  \colorlet{nskFg}{black}
% }

% -\ExplSyntaxOn
% -\bool_new:N \g__nsk_dark_mode_bool
% -\ifnsk@dark_mode
% -\bool_set_true:N \g__nsk_dark_mode_bool
% -\definecolor{ c__nsk_dark_bg_clr }      {HTML}{1d2021}
% -\definecolor{ c__nsk_principal } {HTML}{ebdbb2}
% -\pagecolor{c__nsk_dark_bg_clr}
% -\colorlet{nskBg}{c__nsk_dark_bg_clr}
% -\colorlet{nskFg}{c__nsk_principal}
% -\color{c__nsk_principal}
% -\else
% -\bool_set_false:N \g__nsk_dark_mode_bool
% -\fi
% -\ExplSyntaxOff

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Global props -
% to track auto-increment counters
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\prop_new:N \g_nsk_block_counters_prop
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Global seq -
% to track block-id reference counters
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\seq_new:N \g_nsk_block_id_history_seq

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Main nsk / Block Primitive
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\keys_define:nn {nsk / block}
{
 % block type ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 type .tl_set:N = \l_nsk_block_type_tl,
 type .initial:n = {rectangle},
```

```
type .default:n = {rectangle},

% ref-id ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
id .tl_set:N = \l_nsk_block_id_tl,
id .initial:n = { },

% coords ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
x .fp_set:N = \l_nsk_block_x_fp,
x .initial:n = {0},
x .default:n = {0},
y .fp_set:N = \l_nsk_block_y_fp,
y .initial:n  = {0},
y .default:n = {0},

anchor .tl_set:N = \l_nsk_block_anchor_tl,
anchor .default:n = {},
anchor .initial:n = {},

% natural positioning using positioning library
pos .tl_set:N = \l_nsk_block_pos_tl,
pos .initial:n = { },

% last-nat pos ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
last-pos .tl_set:N = \l_nsk_block_last_pos_tl,
last-pos .initial:n = { },
last-pos .default:n = { },

% (works like last-pos but without error)
last-pos-s .tl_set:N = \l_nsk_block_last_pos_s_tl,
last-pos-s .initial:n = { },
last-pos-s .default:n = { },

% dimensions ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
width .dim_set:N =  \l_nsk_block_width_dim,
width .initial:V = \g__nsk_style_block_width_dim,
width .default:V = \g__nsk_style_block_width_dim,

height .dim_set:N = \l_nsk_block_height_dim,
height .default:V = \g__nsk_style_block_height_dim,
height .initial:V = \g__nsk_style_block_height_dim,


% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Inner (regular) border
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
border-type .code:n =
 {
  \tl_if_eq:nnTF {#1}{none}
  {
   \tl_set:Nn \l_nsk_block_border_type_tl {draw=none}
  }
  {
   \tl_set:Nn \l_nsk_block_border_type_tl {#1}
  }
```

```
 },

border-type .initial:n = {solid},
border-type .default:n = {solid},

border-color .tl_set:N = \l_nsk_block_border_color_tl,
border-color .initial:V = \g__nsk_style_block_border_color_tl,
border-color .default:V = \g__nsk_style_block_border_color_tl,

border-radius .dim_set:N = \l_nsk_block_border_radius_dim,
border-radius .initial:V = \g__nsk_style_block_border_radius_dim,
border-radius .default:V = \g__nsk_style_block_border_radius_dim,

border-size .dim_set:N = \l_nsk_block_border_size_dim,
border-size .initial:V = \g__nsk_style_block_border_size_dim,
border-size .default:V = \g__nsk_style_block_border_size_dim,

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Outer (halo) border
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<

outer-border-type .code:n =
 {
  \tl_if_eq:nnTF {#1}{none}
  {
   \tl_set:Nn \l_nsk_block_outer_border_type_tl {draw=none}
  }
  {
   \tl_set:Nn \l_nsk_block_outer_border_type_tl {#1}
  }

 },

outer-border-type .initial:n = {none},
outer-border-type .default:n = {none},

outer-border-color .tl_set:N = \l_nsk_block_outer_border_color_tl,
outer-border-color .initial:n = { nskMainAccent },
outer-border-color .default:n = { nskMainAccent },

outer-border-radius .dim_set:N = \l_nsk_block_outer_border_radius_dim,
outer-border-radius .initial:V = \g__nsk_style_block_border_radius_dim,
outer-border-radius .default:V = \g__nsk_style_block_border_radius_dim,

outer-border-size .dim_set:N = \l_nsk_block_outer_border_size_dim,
outer-border-size .initial:n = {7mm},
outer-border-size .default:n = {7mm},

% styles ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
fill .tl_set:N = \l_nsk_block_fill_tl,
fill .initial:V = \g__nsk_style_block_fill_tl,
fill .default:V = \g__nsk_style_block_fill_tl,

```

```
% shadow ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
shadow .bool_set:N = \l_nsk_block_shadow_bool,
shadow .initial:n = {true},
shadow .default:n = {true},

% importance ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
importance .fp_set:N = \l_nsk_block_importance_fp,
importance .initial:n = {1.0},
importance .default:n = {1.0},

% -- The special pass-through key for arbitrary TikZ options.
tikz-opts  .code:n =
 {
  % #1 is exactly what is inside { ... }, including any commas.
  % store it literally into \l_nsk_block_extra_style_tl
  \tl_set:Nn \l_nsk_block_extra_style_tl {#1}
 },
tikz-opts  .initial:n = {},
tikz-opts  .default:n = {},

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Text Anchoring
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
%---------------------------
% text-north
%---------------------------
text-north .tl_set:N = \l_nsk_text_north_tl,
text-north .initial:n = { },

text-north-loc .tl_set:N = \l_nsk_text_north_loc_tl,
text-north-loc .initial:V = \g__nsk_style_text_north_loc_tl,

text-north-style .tl_set:N = \l_nsk_text_north_style_tl,
text-north-style .initial:n = { },

%---------------------------
% text-south
%---------------------------
text-south .tl_set:N = \l_nsk_text_south_tl,
text-south .initial:n = { },

text-south-loc .tl_set:N = \l_nsk_text_south_loc_tl,
text-south-loc .initial:V = \g__nsk_style_text_south_loc_tl,

text-south-style .tl_set:N = \l_nsk_text_south_style_tl,
text-south-style .initial:n = { },

%---------------------------
% text-east
%---------------------------
text-east .tl_set:N = \l_nsk_text_east_tl,
text-east .initial:n = { },

text-east-loc .tl_set:N = \l_nsk_text_east_loc_tl,
```

```
text-east-loc .initial:V = \g__nsk_style_text_east_loc_tl,

text-east-style .tl_set:N = \l_nsk_text_east_style_tl,
text-east-style .initial:n = { },

%---------------------------
% text-west
%---------------------------
text-west .tl_set:N = \l_nsk_text_west_tl,
text-west .initial:n = { },

text-west-loc .tl_set:N = \l_nsk_text_west_loc_tl,
text-west-loc .initial:V = \g__nsk_style_text_west_loc_tl,

text-west-style .tl_set:N = \l_nsk_text_west_style_tl,
text-west-style .initial:n = { },

%---------------------------
% text-center
%---------------------------
text-center .tl_set:N = \l_nsk_text_center_tl,
text-center .initial:n = { },

text-center-loc .tl_set:N = \l_nsk_text_center_loc_tl,
text-center-loc .initial:V = \g__nsk_style_text_center_loc_tl,

text-center-style .tl_set:N = \l_nsk_text_center_style_tl,
text-center-style .initial:n = { },

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Embedding
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
embed .tl_set:N = \l_nsk_embed_tl,
embed .default:n = {},
embed .initial:n = {},

embed-gfx .tl_set:N = \l_nsk_embed_gfx_tl,
embed-gfx .default:n = {},
embed-gfx .initial:n = {},

embed-padding .dim_set:N = \l_nsk_block_embed_padding_dim,
embed-padding .default:n = \g__nsk_style_block_embed_padding_dim,
embed-padding .initial:n = \g__nsk_style_block_embed_padding_dim,

embed-border-radius .dim_set:N = \l_nsk_block_embed_border_radius_dim,
embed-border-radius .initial:V = \g__nsk_style_block_embed_border_radius_dim,
embed-border-radius .default:V = \g__nsk_style_block_embed_border_radius_dim,

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Patterns
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
pattern .tl_set:N = \l_nsk_block_pattern_tl,
pattern .initial:n = { },
pattern .default:n = { },
```

```
 pattern-color .tl_set:N = \l_nsk_block_pattern_color_tl,
 pattern-color .initial:n = { },
 pattern-color .default:n = { },
}

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Auto-generate an ID if user does not supply one
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\cs_new_protected_nopar:Npn \nsk__maybe_set_id:
{
 \tl_if_blank:VTF \l_nsk_block_id_tl
 { \nsk__assign_autogenerated_id: }
 { }
}

\cs_new_protected_nopar:Npn \nsk__assign_autogenerated_id:
{
 % If shape key not in the counters prop yet, init to 0
 \prop_if_in:NVF \g_nsk_block_counters_prop \l_nsk_block_type_tl
 {
  % () Use \prop_gput to store globally:
  \prop_gput:NVn \g_nsk_block_counters_prop \l_nsk_block_type_tl { 0 }
 }

 % Retrieve current count
 \prop_get:NVN \g_nsk_block_counters_prop \l_nsk_block_type_tl \l_tmpa_tl

 % Convert to int and increment
 \int_set:Nn \l_tmpa_int { \l_tmpa_tl + 1 }

 % Update the counters property globally:
 \prop_gput:NVx \g_nsk_block_counters_prop \l_nsk_block_type_tl { \int_use:N \l_tmpa_int }

 % Construct "diamond1", "diamond2", "rectangle3", etc.
 \tl_set:No \l_nsk_block_id_tl
 {
  \tl_use:N \l_nsk_block_type_tl
  \int_use:N \l_tmpa_int
 }
}

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Border size importance
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\cs_new_protected_nopar:Npn \nsk__maybe_scale_border_by_importance:
{
 % \fp_if_gt:NNT \l_nsk_block_importance_fp \l_nsk_block_importance_fp
 \fp_compare:nT {\l_nsk_block_importance_fp > 1.0}
 {
  % Convert the current border-size dimension into a float:
  \fp_set:Nn \l_tmpa_fp { \l_nsk_block_border_size_dim }

  % Multiply by the importance factor:
```

```
  \fp_set:Nn \l_tmpa_fp { \l_tmpa_fp * \l_nsk_block_importance_fp }

  % Update the actual dimension variable:
  \dim_set:Nn \l_nsk_block_border_size_dim { \fp_to_dim:N \l_tmpa_fp }
 }
}

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Variant Generation :V
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\cs_generate_variant:Nn \nsk__render_one_text_aux:nnnn { nVVV }
\cs_generate_variant:Nn \nsk__draw_block_aux:n { V }


% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Text Rendering
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\cs_new:Npn \nsk_compute_text_node_anchor:nn #1 #2
{
 \str_case:nn {#1}
 {
  {north}{\str_case:nnF {#2}{ {west}{south~west} {east}{south~east} }{south}}
   {south}{\str_case:nnF {#2}{ {west}{north~west} {east}{north~east} }{north}}
   {east}{\str_case:nnF {#2}{ {north}{north~west} {south}{south~west} }{west}}
   {west}{\str_case:nnF {#2}{ {north}{north~east} {south}{south~east} }{east}}
   {center}{center}
 }
}

\cs_new_protected_nopar:Npn \nsk__render_one_text_aux:nnnn #1 #2 #3 #4
{
 % Compute the node anchor based on the block's side (#1)
 % and the extra alignment provided (#3)
 \tl_set:Nx \l_tmpa_tl { \nsk_compute_text_node_anchor:nn {#1} {#3} }
 \tl_show:N \l_tmpa_tl
 % Draw the node at the block's anchor coordinate (e.g. (blockID.north))
 % but with the computed node anchor.
 \tl_if_blank:nTF {#3}
 {
  \draw[shift=(\l_nsk_block_id_tl.#1)]
  node[anchor=\l_tmpa_tl, #4] {#2};
 }{
  % If extra alignment (#3) is "north" or "south", swap the order.
  \str_if_eq:nnTF {#3}{north}
  {
   \draw[shift=(\l_nsk_block_id_tl.#3~#1)]
   node[anchor=\l_tmpa_tl, #4] {#2};
  }
  {
   \str_if_eq:nnTF {#3}{south}
   {
    \draw[shift=(\l_nsk_block_id_tl.#3~#1)]
    node[anchor=\l_tmpa_tl, #4] {#2};
   }
```

```
   {
    \draw[shift=(\l_nsk_block_id_tl.#1~#3)]
    node[anchor=\l_tmpa_tl, #4] {#2};
   }
  }
 }
}

\cs_new_protected_nopar:Npn \nsk__render_one_text:nnnn #1 #2 #3 #4
{
 % #1 = the actual anchor name in TikZ (e.g., "north", "south", "center", etc.)
 % #2 = text content
 % #3 = location/placement (e.g., "above left", "right", etc.)
 % #4 = style (e.g., "color=red, font=\bfseries")
 % If the user did not provide text (#2), do nothing
 \tl_if_blank:VTF {#2}
 {
  % do nothing
 }
 {
  % Otherwise, call the :VVVV variant
  \nsk__render_one_text_aux:nVVV {#1} {#2} {#3} {#4}
 }
}

\cs_new_protected_nopar:Npn \nsk__render_all_text:
{
 \nsk__render_one_text:nnnn {north}
 { \l_nsk_text_north_tl }
 { \l_nsk_text_north_loc_tl }
 { \l_nsk_text_north_style_tl }

 \nsk__render_one_text:nnnn {south}
 { \l_nsk_text_south_tl }
 { \l_nsk_text_south_loc_tl }
 { \l_nsk_text_south_style_tl }

 \nsk__render_one_text:nnnn {east}
 { \l_nsk_text_east_tl }
 { \l_nsk_text_east_loc_tl }
 { \l_nsk_text_east_style_tl }

 \nsk__render_one_text:nnnn {west}
 { \l_nsk_text_west_tl }
 { \l_nsk_text_west_loc_tl }
 { \l_nsk_text_west_style_tl }

 \nsk__render_one_text:nnnn {center}
 { \l_nsk_text_center_tl }
 { \l_nsk_text_center_loc_tl }
 { \l_nsk_text_center_style_tl }
}

% Internal debug/log block ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
```

```
\cs_new_protected_nopar:Npn \nsk__block_debug:
{
 \iow_term:x {type   = \tl_use:N \l_nsk_block_type_tl}
 \iow_term:x {x      = \fp_use:N \l_nsk_block_x_fp}
 \iow_term:x {y      = \fp_use:N \l_nsk_block_y_fp}
 \iow_term:x {width  = \dim_use:N \l_nsk_block_width_dim}
 \iow_term:x {height = \dim_use:N \l_nsk_block_height_dim}
}


% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Last relative pos
% Automagically relative positioning to last previous block
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\cs_new_protected_nopar:Npn \nsk__process_last_pos:
{
 % Only proceed if the user gave 'last-pos'
 \tl_if_blank:VF \l_nsk_block_last_pos_tl
 {
  \seq_if_empty:NTF \g_nsk_block_id_history_seq
  {
   \msg_warning:nn {nsk}{No-previous-block-for-last-pos}
  }
  {
   % exec last block id retrieval ~~~~~~~~~~~~~~~ <<<
   % block is already registered, so get prev
   \nsk__internal_get_nth_block_id:n{2}
   \tl_set:Nx \l_tmpa_tl { \g_nsk_block_last_requested_block_id }

   \str_if_in:NnTF \l_nsk_block_last_pos_tl {=}
   {
    % case (a) ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
    % There's at least one '='
    % Next see if "of" is in there
    % e.g. "below=0cm"
    % => add " of <blockID>"
    \tl_set:Nx \l_nsk_block_pos_tl
    {
     \exp_not:V \l_nsk_block_last_pos_tl\space of~\exp_not:V \l_tmpa_tl
    }
   }
   {
    % case (b) ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
    % There's NO '=' in the user's string
    % e.g. "below right"
    % => produce "below right=of <blockID>"
    \tl_set:Nx \l_nsk_block_pos_tl
    {
     \exp_not:V \l_nsk_block_last_pos_tl =of~\exp_not:V \l_tmpa_tl
    }
   }
  }
 }
}
```

```
528
529 \cs_new_protected_nopar:Npn \nsk__process_last_pos_s:
530 {
531  % Only proceed if the user gave a nonblank last-pos-x value:
532  \tl_if_blank:VF \l_nsk_block_last_pos_s_tl
533  {
534   % Check that there is more than one block in the history.
535   \int_compare:nNnTF {\seq_count:N \g_nsk_block_id_history_seq} > {1}
536    {
537     % There is a previous block. Retrieve it without calling the error-triggering routine.
538     \int_set:Nn \l_tmpa_int { \seq_count:N \g_nsk_block_id_history_seq }
539     \tl_set:No \l_tmpa_tl
540     {
541      \seq_item:Nn \g_nsk_block_id_history_seq
542      { \int_eval:n { \l_tmpa_int - 1 } }
543     }
544     % Process the key's value as in the original routine:
545     \str_if_in:NnTF \l_nsk_block_last_pos_s_tl {=}
546     {
547      \tl_set:Nx \l_nsk_block_pos_tl
548      {
549       \exp_not:V \l_nsk_block_last_pos_s_tl\space of~\exp_not:V \l_tmpa_tl
550      }
551     }
552     {
553      \tl_set:Nx \l_nsk_block_pos_tl
554      {
555       \exp_not:V \l_nsk_block_last_pos_s_tl =of~\exp_not:V \l_tmpa_tl
556      }
557     }
558    }
559    {
560     % There is no previous block: do nothing silently.
561    }
562  }
563 }
564 % ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
565 % Block drawing [Aux]
566 % ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
567 \cs_new_protected_nopar:Npn \nsk__draw_block_aux:n #1
568 {
569
570  \node [
571   #1,
572  ] {};
573
574  % embedding ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
575  \tl_if_blank:VF {\l_nsk_embed_tl}
576  {
577   \dim_set:Nn \l_tmpa_dim { \l_nsk_block_width_dim - 2\l_nsk_block_embed_padding_dim}
578   \dim_set:Nn \l_tmpb_dim { \l_nsk_block_height_dim - 2\l_nsk_block_embed_padding_dim }
579
580   \begin{scope}
581    \pgftransformshift{\pgfpointanchor{\tl_use:N \l_nsk_block_id_tl}{center}}
```

```
   \pgfset{minimum\space width=\l_tmpa_dim, minimum\space height=\l_tmpb_dim}
   \pgfsetcornersarced{\pgfpoint{\l_nsk_block_embed_border_radius_dim}{\l_nsk_block_embed_boro
   \pgfnode{\l_nsk_block_type_tl}{center}{}{nodename}{\pgfusepath{stroke,clip}}

   % raw embedding ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
   \tl_use:N \l_nsk_embed_tl
  \end{scope}

 }

 \tl_if_blank:VF {\l_nsk_embed_gfx_tl}
 {
  \dim_set:Nn \l_tmpa_dim { \l_nsk_block_width_dim - 2\l_nsk_block_embed_padding_dim}
  \dim_set:Nn \l_tmpb_dim { \l_nsk_block_height_dim - 2\l_nsk_block_embed_padding_dim }

  \begin{scope}
   \pgftransformshift{\pgfpointanchor{\tl_use:N \l_nsk_block_id_tl}{center}}
   \pgfset{minimum\space width=\l_tmpa_dim, minimum\space height=\l_tmpb_dim}
   \pgfsetcornersarced{\pgfpoint{\l_nsk_block_embed_border_radius_dim}{\l_nsk_block_embed_boro
   \pgfnode{\l_nsk_block_type_tl}{center}{}{nodename}{\pgfusepath{stroke,clip}}

   % gfx node ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
   \node[]
   {\includegraphics[width=\l_nsk_block_width_dim, height=\l_nsk_block_height_dim]{\tl_use:N \
  \end{scope}

 }


}

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Pattern Aliasing resolution
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\cs_new_protected_nopar:Npn \nsk__resolve_pattern_alias:
{
 % If \l_nsk_block_pattern_tl is empty, do nothing
 \tl_if_blank:VTF \l_nsk_block_pattern_tl
 { }
 {
  % Check if the user's pattern key is in the alias property
  \prop_if_in:NVTF \g_nsk_pattern_aliases_prop \l_nsk_block_pattern_tl
  {
   % It's an alias => retrieve the real pattern
   \prop_get:NVN \g_nsk_pattern_aliases_prop \l_nsk_block_pattern_tl \l_tmpa_tl
   \tl_set:NV \l_nsk_block_pattern_tl \l_tmpa_tl
  }
  {
   % Not an alias => user gave a real pattern or a custom name => do nothing
  }
 }
}
```

```
\cs_new_protected_nopar:Npn \nsk__build_block_style:
{
 % Build the style token list incrementally:
 \tl_clear_new:N \l_tmps_tl

 \tl_set:Nn \l_tmp_pre_acts {}
 \tl_set:Nn \l_tmp_post_acts {}

 % extras ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 \nsk__resolve_pattern_alias:

 % Possibly scale border-size according to importance:
 \nsk__maybe_scale_border_by_importance:

 % id-ref ------------------------------------------------
 \tl_put_right:No \l_tmps_tl { name=\l_nsk_block_id_tl,}

 % {x,y} pos ---------------------------------------------
 \tl_if_blank:VTF \l_nsk_block_pos_tl
 {
  % -- pos is blank => place the node using x,y
  \tl_put_right:Nx \l_tmps_tl
  { at={(\fp_use:N \l_nsk_block_x_fp,\fp_use:N \l_nsk_block_y_fp)}, }
 }
 {
  % -- pos is given => place the node by raw TikZ positioning
  % i.e., [right=2cm of otherNode]
  \tl_put_right:No \l_tmps_tl { \l_nsk_block_pos_tl, }
 }

 \tl_if_blank:VF \l_nsk_block_anchor_tl
 {
  \tl_put_right:No \l_tmps_tl {anchor=\l_nsk_block_anchor_tl,}
 }

 % shape-type ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 \tl_put_right:No \l_tmps_tl {shape=\l_nsk_block_type_tl,}
 % dims ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 \tl_put_right:No \l_tmps_tl {minimum\space width=\l_nsk_block_width_dim,}
 \tl_put_right:No \l_tmps_tl {minimum\space height=\l_nsk_block_height_dim,}
 % borders ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 \tl_put_right:No \l_tmps_tl {draw=\l_nsk_block_border_color_tl,}
 \tl_put_right:No \l_tmps_tl {\l_nsk_block_border_type_tl,}
 \tl_put_right:No \l_tmps_tl {rounded\space corners=\l_nsk_block_border_radius_dim,}
 \tl_put_right:No \l_tmps_tl {line\space width=\l_nsk_block_border_size_dim,}
 % outer border ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 \tl_put_right:No \l_tmp_pre_acts {draw=\l_nsk_block_outer_border_color_tl,}
 \tl_put_right:No \l_tmp_pre_acts {\l_nsk_block_outer_border_type_tl,}
 \tl_put_right:No \l_tmp_pre_acts {rounded\space corners=\l_nsk_block_outer_border_radius_dim
 \tl_put_right:No \l_tmp_pre_acts {line\space width=\l_nsk_block_outer_border_size_dim,}

 % fill ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 \tl_put_right:No \l_tmps_tl {fill=\l_nsk_block_fill_tl,}

```

```
% shadow ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\bool_if:NT \l_nsk_block_shadow_bool
{
 \tl_put_right:No \l_tmps_tl {nskShadow, }
}
% patterns ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\tl_if_blank:VTF \l_nsk_block_pattern_tl
{ }
{
 % something like:
 %   postaction={pattern=..., pattern color=...}
 \tl_clear_new:N \l_tmpb_tl
 \tl_put_right:Nx \l_tmpb_tl { pattern=\l_nsk_block_pattern_tl, }
 \tl_if_blank:VTF \l_nsk_block_pattern_color_tl
 { }
 {
  \tl_put_right:Nx \l_tmpb_tl { pattern\space color=\l_nsk_block_pattern_color_tl, }
 }

 % append to post-actions
 \tl_put_right:Nx \l_tmp_post_acts {\l_tmpb_tl}
}


% pass-through ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\tl_if_empty:VTF \l_nsk_block_extra_style_tl
{ }
{
 \tl_put_right:NV \l_tmps_tl {\l_nsk_block_extra_style_tl,}
}


% add any pre-actions ~~~~~~~~~~~~~~~~~~~~~~~ <<<
\tl_put_right:Nx \l_tmps_tl { preaction={\tl_use:N \l_tmp_pre_acts}, }
% add any post-actions ~~~~~~~~~~~~~~~~~~~~~~ <<<
\tl_put_right:Nx \l_tmps_tl { postaction={\tl_use:N \l_tmp_post_acts}, }

% Now call function which takes
% params by value: --
\nsk__draw_block_aux:V
\l_tmps_tl
}


% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Ref-Counter history
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\cs_new_protected_nopar:Npn \nsk__store_block_in_history:
{
 % Push the current block's ID into the global sequence
 \seq_gput_right:Nx \g_nsk_block_id_history_seq \l_nsk_block_id_tl
}

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

```
{
 % #1 => integer offset n (nth-last)
 % We will reference the global \g_nsk_block_id_history_seq inside.


 \seq_if_empty:NTF \g_nsk_block_id_history_seq
 {
  \tl_set:Nn \l_tmpa_tl {ERROR-NoBlocks}
  \GenericError{}{No block in the history!}{}{%
   Attempted~to~reference~\string\nskBlockIDLast[#1],~but~no~blocks~exist.%
  }
 }
 {
  \int_compare:nNnTF {#1} > { \seq_count:N \g_nsk_block_id_history_seq }
   {
    \tl_set:Nn \l_tmpa_tl {ERROR-BadIndex}
    \GenericError{}{Block~index~out~of~bounds~}{}{%
     \string\nskBlockIDLast[#1] exceeds the number of existing blocks.%
    }
   }
   {
    % We want the item at \(total - n + 1)."
    \tl_set:No \l_tmpa_tl
    {
     \seq_item:Nn
     \g_nsk_block_id_history_seq
     { \int_eval:n{ \seq_count:N \g_nsk_block_id_history_seq - #1 + 1} }
    }
   }
 }


 % Return it by expanding the token list \l_tmpa_tl
 % \tl_use:N \l_tmpa_tl
 \tl_gset:Nx \g_nsk_block_last_requested_block_id {\l_tmpa_tl}

 % \seq_show:N \g_nsk_block_id_history_seq
 % \tl_show:N \g_nsk_block_last_requested_block_id
}


% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Main block draw
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\cs_new_protected_nopar:Npn \nsk__draw_block:
{
 % 1. draw the main block shape
 \nsk__build_block_style:

 % 2. draw the text anchorings
 \nsk__render_all_text:
}

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
% Main nsk / figure Primitive
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<

\keys_define:nn { nsk / figure }
{
 center .bool_set:N = \l_nsk_figure_center_bool,
 center .initial:n  = { false },
 center .default:n  = { true },

 % pass-through ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
 unknown .code:n =
  {
   \prop_put:NVn \l_nsk_figure_unknown_prop { \l_keys_key_tl } { #1 }
  },
}

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

 % # 1 -> k=v list
 \keys_set:nn {nsk / block } {#1}

 % possibly autogenerate the id
 \nsk__maybe_set_id:

 % store id in history
 \nsk__store_block_in_history:


 % process last-pos -> set cpos if neeeded
 % if last-pos-s is given, use it (without error); otherwise use last-pos
 \tl_if_blank:VTF \l_nsk_block_last_pos_s_tl
 { \nsk__process_last_pos: }
 { \nsk__process_last_pos_s: }

 \nsk__block_debug:

 % actually draw the block
 \nsk__draw_block:

 \group_end:
}

\NewDocumentEnvironment{nskFigure}{O{}}
{
\prop_gclear:N \g_nsk_block_counters_prop
% (b) Clear the unknown-props container
\prop_clear:N \l_nsk_figure_unknown_prop

```

```
% (c) Parse the user-provided options
\keys_set:nn { nsk / figure } { #1 }

% (d) Convert the unknown key=value pairs into a comma list
\tl_clear_new:N \l_nsk_figure_opts_tl
\prop_map_inline:Nn \l_nsk_figure_unknown_prop
{
 \tl_put_right:Nx \l_nsk_figure_opts_tl { ##1 = { ##2 }, }
}

% (e) If center=true, wrap the tikzpicture in a center environment
\bool_if:NTF \l_nsk_figure_center_bool
{
 \begin{center}
 \begin{tikzpicture}[\tl_use:N \l_nsk_figure_opts_tl]
}
{
 \begin{tikzpicture}[\tl_use:N \l_nsk_figure_opts_tl]
  }
  }{
 \end{tikzpicture}
 \bool_if:NT \l_nsk_figure_center_bool
 {
  \end{center}
 }
}

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Miscs Utils
% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ <<<
\NewDocumentCommand \nskBlockID {}
{
 \tl_use:N \l_nsk_block_id_tl
}

% reference counting ~~~~~~~~~~~~~~~~~~~~~~~~ <<<
% \nskBlockIDLast[1] -> "the nth last block ID"
\NewDocumentCommand \nskBlockIDLast { O{1} }
{
 \nsk__internal_get_nth_block_id:n {#1}
 \tl_use:N \g_nsk_block_last_requested_block_id
}

\NewDocumentCommand \nskLastBlockID {} {
 % "Alias" for the single last block (i.e., \nskBlockIDLast[])
 % block is already registered, so get prev ->2
 \nskBlockIDLast[2]
}


\ExplSyntaxOff

¡/package¿
```