The **Tilings** package: code

Andrew Stacey
loopspace@mathforge.org

v2.0 from 2023/06/01

Contents

1 Introduction

This is a TikZ library for drawing tiles, such as Penrose tiles (kite/dart, rhombus, and pentagon versions) and the aperiodical polykite tiles. It provides two methods of drawing: one in which an automatic pattern is built, and one where the tiles can be placed "by hand". The tiles can be shaped and (hopefully!) still fit together. For full user documentation, see the `tilings.pdf` file.

2 Implementation

```
⟨@@=tilings⟩
```

2.1 Initialisation

We use the `spath3` library for manipulating the paths that will make up the tiles.

```
\ProvidesExplFile
      {tikzlibrarytilings.code.tex}
      {2023/06/01}
      {2.0}
      {TikZ pics for tilings such as Penrose tiles}
\RequirePackage{tikz}
\usetikzlibrary{spath3}
```

Now we move in to the realm of LaTeX3.

```
\ExplSyntaxOn
```

Start with some basic paths (lines) for the sides of the tiles so that we know that we have well-defined tiles at the outset. These are globally defined as we will frequently want to define them in one tikzpicture and use them in another.

```
\tl_new:N \g__tilings_side_a_tl
\tl_new:N \g__tilings_side_b_tl
\tl_new:N \g__tilings_side_c_tl
```

```
\tl_new:N \g__tilings_side_d_tl
\tl_new:N \g__tilings_side_e_tl
\tl_new:N \g__tilings_side_A_tl
\tl_new:N \g__tilings_side_B_tl
\tl_new:N \g__tilings_side_C_tl
\tl_new:N \g__tilings_side_D_tl
\tl_new:N \g__tilings_side_E_tl
\tl_new:c {g__tilings_side_1_tl}
\tl_new:c {g__tilings_side_2_tl}
\tl_new:c {g__tilings_side_3_tl}
\tl_gset:Nn \g__tilings_side_a_tl
{
  \pgfsyssoftpath@movetotoken{0pt}{0pt}
  \pgfsyssoftpath@linetotoken{1pt}{0pt}
}
\tl_gset_eq:NN \g__tilings_side_b_tl \g__tilings_side_a_tl
\tl_gset_eq:NN \g__tilings_side_c_tl \g__tilings_side_a_tl
\tl_gset_eq:NN \g__tilings_side_d_tl \g__tilings_side_a_tl
\tl_gset_eq:NN \g__tilings_side_e_tl \g__tilings_side_a_tl
\tl_gset_eq:NN \g__tilings_side_A_tl \g__tilings_side_a_tl
\tl_gset_eq:NN \g__tilings_side_B_tl \g__tilings_side_a_tl
\tl_gset_eq:NN \g__tilings_side_C_tl \g__tilings_side_a_tl
\tl_gset_eq:NN \g__tilings_side_D_tl \g__tilings_side_a_tl
\tl_gset_eq:NN \g__tilings_side_E_tl \g__tilings_side_a_tl
\tl_gset_eq:cN {g__tilings_side_1_tl} \g__tilings_side_a_tl
\tl_gset_eq:cN {g__tilings_side_2_tl} \g__tilings_side_a_tl
\tl_gset_eq:cN {g__tilings_side_3_tl} \g__tilings_side_a_tl
```

ngs_tmpa_fp
ngs_tmpb_fp
ngs_tmpc_fp
_saved_x_fp
_saved_y_fp
gs_tmpa_str
gs_tmpb_str
gs_tmpa_seq
ngs_tmpa_tl
ngs_tmpb_tl
ngs_tmpc_tl
ngs_tmpd_tl
gs_tmpa_int
gs_tmpb_int
ings_xa_dim
ings_ya_dim
ings_xb_dim
ings_yb_dim
ings_xa_dim
ings_ya_dim
ings_xb_dim
ings_yb_dim
s_tmpa_prop
ngs_cw_bool
pdate_saved_bool
lative_bool
s_output_tl

We need a few temporary variables to hold intermediate calculations.

```
\fp_new:N \l__tilings_tmpa_fp
\fp_new:N \l__tilings_tmpb_fp
\fp_new:N \l__tilings_tmpc_fp
\fp_new:N \l__tilings_saved_x_fp
\fp_new:N \l__tilings_saved_y_fp
\str_new:N \l__tilings_tmpa_str
\str_new:N \l__tilings_tmpb_str
\seq_new:N \l__tilings_tmpa_seq
\tl_new:N \l__tilings_tmpa_tl
\tl_new:N \l__tilings_tmpb_tl
\tl_new:N \l__tilings_tmpc_tl
\tl_new:N \l__tilings_tmpd_tl
\tl_new:N \l__tilings_tmp_tile_path_tl
\tl_new:N \l__tilings_action_lms_tl
\tl_new:N \l__tilings_parameters_lms_tl
\int_new:N \l__tilings_tmpa_int
\int_new:N \l__tilings_tmpb_int
\fp_new:N \l__tilings_xa_fp
\fp_new:N \l__tilings_ya_fp
\fp_new:N \l__tilings_xb_fp
\fp_new:N \l__tilings_yb_fp
\dim_new:N \l__tilings_xa_dim
\dim_new:N \l__tilings_ya_dim
\dim_new:N \l__tilings_xb_dim
\dim_new:N \l__tilings_yb_dim
\dim_new:N \g__tilings_xa_dim
\dim_new:N \g__tilings_ya_dim
\dim_new:N \g__tilings_xb_dim
\dim_new:N \g__tilings_yb_dim
\prop_new:N \l__tilings_tmpa_prop
\bool_new:N \l__tilings_cw_bool
\bool_new:N \l__tilings_update_saved_bool
```

```
\bool_new:N \l__tilings_relative_bool
\bool_new:N \l__tilings_edge_bool
\str_const:Nn \c__tilings_colon_str {:}
\str_const:Nn \c__tilings_comma_str {,}
\fp_const:Nn \c__tilings_cm_fp {\dim_to_fp:n {1cm}}
\tl_new:N \g__tilings_output_tl
\fp_new:N \g__tilings_output_a_fp
\fp_new:N \g__tilings_output_b_fp
\prop_new:N \g__tilings_tilenames_prop
\regex_const:Nn \c__tilings_anchor_regex {\s\w+\Z}
\cs_generate_variant:Nn \seq_set_split:Nnn {NVV}
\cs_generate_variant:Nn \regex_extract_once:NnNTF {NVNTF}
\cs_generate_variant:Nn \tl_if_eq:nnT {nVT}
\cs_generate_variant:Nn \tl_if_in:NnT {NVT}
\cs_generate_variant:Nn \prop_item:Nn {cV}
\cs_generate_variant:Nn \tl_if_head_is_group_p:n {V}
```

(*End definition for* `\l__tilings_tmpa_fp` *and others.*)

2.2 Helpful Error Messages

```
\msg_new:nnn { tilings }{ not baked }
{ Tile~ #1~ has~ not~ been~ baked. }
\msg_new:nnn { tilings }{ no tile }
{ Tile~ #1~ has~ not~ been~ defined. }
\msg_new:nnn { tilings }{ no side }
{ Tile~ side~ #1~ has~ not~ been~ defined,~ using~ default. }
\msg_new:nnn { tilings }{ tile no edge }
{ Tile~ #1~ doesn't~ have~ an~ edge~ labelled~ #2;
  ~ available~ edges~ are~ #3.}
\msg_new:nnn { tilings }{ no edge }
{ Either~ tile~ #1~ doesn't~ exist
  ~ or~ it~ doesn't~ have~ an~ edge~ labelled~ #2.}
```

2.3 Creating the Tiles

`ise_path:Nn` When defining the path for a side, we normalise so that it starts at the origin and ends at `(1pt,0pt)`.

```
\cs_new_nopar:Npn \__tilings_normalise_path:Nn #1#2
{
```

Get the initial point of the path and convert to floating point.

```
  \group_begin:
  \spath_initialpoint:Nn \l__tilings_tmpa_tl {#2}
  \fp_set:Nn \l__tilings_tmpa_fp {\tl_head:N \l__tilings_tmpa_tl}
  \tl_set:Nx \l__tilings_tmpa_tl {\tl_tail:N \l__tilings_tmpa_tl}
  \fp_set:Nn \l__tilings_tmpb_fp {\tl_head:N \l__tilings_tmpa_tl}
```

Get the final point of the path, and compute the difference of the final and initial points.

The resulting numbers, say a and b, will be put into a matrix to rotate and scale the path. The formula for the matrix is:

$$\frac{1}{a^2+b^2}\begin{bmatrix} a & b \\ -b & a \end{bmatrix}$$

```
  \spath_finalpoint:Nn \l__tilings_tmpa_tl {#2}
  \fp_set:Nn \l__tilings_tmpa_fp
  {\tl_head:N \l__tilings_tmpa_tl - \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpa_tl {\tl_tail:N \l__tilings_tmpa_tl}
  \fp_set:Nn \l__tilings_tmpb_fp
  {\tl_head:N \l__tilings_tmpa_tl - \l__tilings_tmpb_fp}
```

Now compute the square of the length of the path for scaling.

```
  \fp_set:Nn \l__tilings_tmpc_fp
  {\l__tilings_tmpa_fp^2 + \l__tilings_tmpb_fp^2}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_tmpa_fp/\l__tilings_tmpc_fp}
  \fp_set:Nn \l__tilings_tmpb_fp {\l__tilings_tmpb_fp/\l__tilings_tmpc_fp}
  \fp_set:Nn \l__tilings_tmpc_fp {-\l__tilings_tmpb_fp}
```

Now construct the matrix.

```
  \tl_set:Nx \l__tilings_tmpb_tl
  {
    {\fp_use:N \l__tilings_tmpa_fp}
    {\fp_use:N \l__tilings_tmpc_fp} % swapped
    {\fp_use:N \l__tilings_tmpb_fp} % swapped
    {\fp_use:N \l__tilings_tmpa_fp}
  }
```

Get the initial point back again for the translation part.

```
  \spath_initialpoint:Nn \l__tilings_tmpa_tl {#2}
```

But we need to premultiply by the matrix because of how the transformations are applied.

```
  \fp_set:Nn \l__tilings_tmpa_fp
  {
    (-1) * \l__tilings_tmpa_fp * \tl_head:N \l__tilings_tmpa_tl
    + (-1) * \l__tilings_tmpb_fp * \tl_tail:N \l__tilings_tmpa_tl
  }
  \fp_set:Nn \l__tilings_tmpb_fp
  {
    (-1) * \l__tilings_tmpa_fp * \tl_tail:N \l__tilings_tmpa_tl
    +  \l__tilings_tmpb_fp * \tl_head:N \l__tilings_tmpa_tl
  }
```

Finally, we apply the transformation to the path.

```
  \tl_put_right:Nx \l__tilings_tmpb_tl {
    {\fp_to_dim:N \l__tilings_tmpa_fp}
    {\fp_to_dim:N \l__tilings_tmpb_fp}
  }
  \spath_transform:NnV \l__tilings_tmpa_tl {#2} \l__tilings_tmpb_tl
  \tl_gset_eq:NN \g__tilings_output_tl \l__tilings_tmpa_tl
  \group_end:
  \tl_set_eq:NN #1 \g__tilings_output_tl
  \tl_gclear:N \g__tilings_output_tl
}
\cs_generate_variant:Nn \__tilings_normalise_path:Nn {NV, cn, cV}
\cs_new_protected_nopar:Npn \__tilings_normalise_path:N #1
{
  \__tilings_normalise_path:NV #1#1
}
\cs_generate_variant:Nn \__tilings_normalise_path:N {c}
```

(*End definition for* `\__tilings_normalise_path:Nn`.)

`etTilingPath` This sets the path corresponding to a particular side to the current path, and normalises it.

```
\cs_new_protected_nopar:Npn \__tilings_set_tiling_path:n #1
{
  \group_begin:
  \pgfsyssoftpath@getcurrentpath\l__tilings_tmpa_tl
  \__tilings_normalise_path:N \l__tilings_tmpa_tl
  \tl_gset_eq:cN {g__tilings_side_#1_tl} \l__tilings_tmpa_tl
  \group_end:
}

\NewDocumentCommand \SetTilingPath { m }
{
  \__tilings_set_tiling_path:n {#1}
}
```

can_point:n

This is a wrapper around `\tikz@scan@one@point` to make it easier to use with LaTeX3 variables.

```
\cs_new_nopar:Npn \tikz_scan_point:n #1
{
  \tikz@scan@one@point\pgfutil@firstofone#1\relax
}
\cs_generate_variant:Nn \tikz_scan_point:n {V}
```

(*End definition for* `\tikz_scan_point:n`*.*)

_defined:TF

This extracts the code that tests if a node is defined.

```
\prg_new_conditional:Npnn \tikz_node_if_defined:n #1 {p,T,F,TF}
{
  \tl_if_exist:cTF {pgf@sh@ns@\use:c{tikz@pp@name}{#1}}
  {
    \prg_return_true:
  }{
    \tl_if_exist:cTF
    {pgf@sh@ns@not yet positionedPGFINTERNAL\use:c{tikz@pp@name}{#1}}
    {
      \pgf_return_true:
    }
    {
      \prg_return_false:
    }
  }
}
```

(*End definition for* `\tikz_node_if_defined:TF`*.*)

keys_get:Nn
fkeys_get:n

This is a wrapper around `\pgfkeysgetvalue` to make it easier to use with LaTeX3 variables.

```
\cs_new_nopar:Npn \__tilings_keys_get:Nn #1#2
{
  \pgfkeysgetvalue{/tikz/tiling/#2}{#1}
}
\cs_new_nopar:Npn \__tilings_keys_get:n #1
{
  \pgfkeysvalueof{/tikz/tiling/#1}
}
\cs_new_nopar:Npn \__tilings_tikz_keys_get:Nn #1#2
{
  \pgfkeysgetvalue{/tikz/#2}{#1}
}
\cs_new_nopar:Npn \__tilings_tikz_keys_get:n #1
{
  \pgfkeysvalueof{/tikz/#1}
}
\cs_new_nopar:Npn \__tilings_pgf_keys_get:Nn #1#2
{
  \pgfkeysgetvalue{#2}{#1}
}
\cs_new_nopar:Npn \__tilings_pgf_keys_get:n #1
{
  \pgfkeysvalueof{#1}
}
```

(*End definition for* `\__tilings_keys_get:Nn` *and* `\pgfkeys_get:n`*.*)

ke_tile:nnn

This builds the tile path from its pieces. The arguments are the name of the tile, the descriptions of the sides, and a token list of the coordinates.

```
\cs_new_nopar:Npn \__tilings_make_tile:nnn #1#2#3
{
```

Get the first coordinate and initialise the path with a move to this point.

```
\group_begin:
\tl_set:Nn \l__tilings_tmpa_tl {#3}
\tl_set:Nx \l__tilings_tmpb_tl {\tl_head:N \l__tilings_tmpa_tl}
\tl_set:Nn \l__tilings_tmpa_tl {\pgfsyssoftpath@movetotoken}
\tl_put_right:Nx \l__tilings_tmpa_tl
{
  {
    \fp_to_dim:n
    {(\tl_item:Nn \l__tilings_tmpb_tl {1}) * \c__tilings_cm_fp}
  }
  {
    \fp_to_dim:n
    {(\tl_item:Nn \l__tilings_tmpb_tl {2}) * \c__tilings_cm_fp}
  }
}
\tl_set_eq:NN \l__tilings_tmp_tile_path_tl \l__tilings_tmpa_tl
```

Now we have our path initialised, we can start appending the side paths according to the specification in the second argument.

We append the initial coordinate to the end of the list to make a closed cycle.

```
\tl_set:Nn \l__tilings_tmpa_tl {#3}
\tl_put_right:Nx \l__tilings_tmpa_tl {{\tl_head:N \l__tilings_tmpa_tl}}
```

Now we walk through the description of the sides, adding the specified paths to our tile path.

```
\tl_map_inline:nn {#2} {
```

Clone the path for this side.

```
  \tl_set:Nx \l__tilings_tmpc_tl {\tl_head:n {##1}}
  \tl_if_exist:cF {g__tilings_side_ \tl_use:N \l__tilings_tmpc_tl _tl}
  {
    \msg_error:nnx { tilings }{ no side } { \tl_use:N \l__tilings_tmpc_tl }
    \tl_gset_eq:cc
    {g__tilings_side_ \tl_use:N \l__tilings_tmpc_tl _tl}
    {g__tilings_side_a_tl}
  }
  \tl_set_eq:Nc \l__tilings_tmpd_tl
  {g__tilings_side_ \tl_use:N \l__tilings_tmpc_tl _tl}
```

Strip off the next coordinate, and convert it to a point.

```
  \tl_set:Nx \l__tilings_tmpb_tl {\tl_head:N \l__tilings_tmpa_tl}
  \tl_set:Nx \l__tilings_tmpa_tl {\tl_tail:N \l__tilings_tmpa_tl}
```

Store the resulting coordinate.

```
  \fp_set:Nn \l__tilings_tmpa_fp
  { \tl_item:Nn \l__tilings_tmpb_tl {1} }
  \fp_set:Nn \l__tilings_tmpb_fp
  { \tl_item:Nn \l__tilings_tmpb_tl {2} }
```

Now get the next coordinate.

```
  \tl_set:Nx \l__tilings_tmpb_tl {\tl_head:N \l__tilings_tmpa_tl}
```

We want the difference between the two coordinates.

```
  \fp_set:Nn \l__tilings_tmpa_fp
  {\tl_item:Nn \l__tilings_tmpb_tl {1} - \l__tilings_tmpa_fp}
  \fp_set:Nn \l__tilings_tmpb_fp
  {\tl_item:Nn \l__tilings_tmpb_tl {2} - \l__tilings_tmpb_fp}
```

This is converted into a transformation matrix.

```
  \fp_set:Nn \l__tilings_tmpc_fp {-\l__tilings_tmpb_fp}
  \tl_set:Nx \l__tilings_tmpb_tl
  {
    {\fp_to_dim:n { \l__tilings_tmpa_fp * \c__tilings_cm_fp }}
    {\fp_to_dim:n { \l__tilings_tmpb_fp * \c__tilings_cm_fp }}% not swapped
```

```
      {\fp_to_dim:n { \l__tilings_tmpc_fp * \c__tilings_cm_fp }}% not swapped
      {\fp_to_dim:n { \l__tilings_tmpa_fp * \c__tilings_cm_fp }}
      {0}
      {0}
    }
```

The transformation is applied to the cloned path.

```
    \spath_transform:NV \l__tilings_tmpd_tl \l__tilings_tmpb_tl
```

And this is welded to the tile path.

```
    \spath_weld:NV \l__tilings_tmp_tile_path_tl \l__tilings_tmpd_tl
  }
```

At the end we close the path.

```
  \spath_close:N \l__tilings_tmp_tile_path_tl
  \tl_gset_eq:NN \g__tilings_output_tl \l__tilings_tmp_tile_path_tl
  \group_end:
  \tl_gclear_new:c {g__tilings_tile_#1_tl}
  \tl_gset_eq:cN {g__tilings_tile_#1_tl} \g__tilings_output_tl
  \tl_gclear:N \g__tilings_output_tl
}
```

(*End definition for* `\__tilings_make_tile:nnn`*.*)

`ake_tile:nn`

A wrapper around the above which allows us to specify the second two arguments as two items in a token list.

```
\cs_new_nopar:Npn \__tilings_make_tile:nn #1#2
{
  \__tilings_make_tile:nnn {#1} #2
}
\cs_generate_variant:Nn \__tilings_make_tile:nn {nV}
```

(*End definition for* `\__tilings_make_tile:nn`*.*)

2.4 Specifying the Tiles

The tile specifications are contained in a `prop`.

```
\prop_new:N \g__tilings_tiles_prop
```

`d_coordinate:Nnn`
`oordinate:w`

Process a coordinate through `fp` and adds it to a token list.

```
\cs_new_nopar:Npn \__tilings_add_coordinate:Nnn #1#2#3 {
  \group_begin:
  \fp_set:Nn \l__tilings_tmpa_fp {#2}
  \fp_set:Nn \l__tilings_tmpb_fp {#3}
  \bool_if:NT \l__tilings_relative_bool
  {
    \fp_add:Nn \l__tilings_tmpa_fp {\l__tilings_saved_x_fp}
    \fp_add:Nn \l__tilings_tmpb_fp {\l__tilings_saved_y_fp}
  }
  \fp_gset_eq:NN \g__tilings_output_a_fp \l__tilings_tmpa_fp
  \fp_gset_eq:NN \g__tilings_output_b_fp \l__tilings_tmpb_fp
  \group_end:
  \tl_put_right:Nx #1
  {
    {{\fp_use:N \g__tilings_output_a_fp}{\fp_use:N \g__tilings_output_b_fp}}
  }
  \bool_if:NT \l__tilings_update_saved_bool
  {
    \fp_set_eq:NN \l__tilings_saved_x_fp \g__tilings_output_a_fp
    \fp_set_eq:NN \l__tilings_saved_y_fp \g__tilings_output_b_fp
  }
  \fp_gzero:N \g__tilings_output_a_fp
  \fp_gzero:N \g__tilings_output_b_fp
}
```

Wrapper around the add coordinate command to split at a comma.

```
\cs_new_nopar:Npn \__tilings_add_xy_coordinate:w #1#2,#3 \q_stop
{
  \__tilings_add_coordinate:Nnn #1 {#2}{#3}
}
```

Wrapper around the add coordinate command to split at a colon.

```
\cs_new_nopar:Npn \__tilings_add_rth_coordinate:w #1#2:#3 \q_stop
{
  \__tilings_add_coordinate:Nnn #1 {(#3) * cosd(#2)}{(#3) * sind(#2)}
}
```

(*End definition for* `\__tilings_add_coordinate:Nnn` *and* `\__tilings_add_coordinate:w`.)

side_to_axis:Nnn Apply a transformation to make a given side lie on the x-axis. Second argument is the tile, third is the side, first is whether to reverse the side.

```
\cs_new_nopar:Npn \__tilings_transform_side_to_axis:Nnn #1#2#3
{
```

Get our tile data, checking if the tile exists.

```
  \group_begin:
  \prop_get:NnNTF \g__tilings_tiles_prop {#2} \l__tilings_tmpa_tl
  {
```

Start with the edge list.

Initialise the counter.

```
  \int_zero:N \l__tilings_tmpb_int
  \int_incr:N \l__tilings_tmpb_int
```

Get the path type list.

```
  \tl_set:Nx \l__tilings_tmpc_tl {\tl_head:N \l__tilings_tmpa_tl}
```

Iterate through the path type list, looking for the requested path.

```
    \bool_set_false:N \l__tilings_edge_bool
    \tl_map_inline:Nn \l__tilings_tmpc_tl {
      \str_if_eq:nnT {##1} {#3} {
        \bool_set_true:N \l__tilings_edge_bool
        \tl_map_break:
      }
      \int_incr:N \l__tilings_tmpb_int
    }
    \bool_if:NTF \l__tilings_edge_bool
    {
```

Get the coordinate list.

```
      \tl_set:Nx \l__tilings_tmpc_tl {\tl_tail:N \l__tilings_tmpa_tl}
```

Strip off the outer braces.

```
      \tl_set:Nx \l__tilings_tmpc_tl {\tl_item:Nn \l__tilings_tmpc_tl {1}}
```

Add the first coordinate at the end.

```
      \tl_put_right:Nx \l__tilings_tmpc_tl
      {{\tl_item:Nn \l__tilings_tmpc_tl {1}}}
```

Get the coordinates for this edge.

```
      \tl_set:Nx \l__tilings_tmpa_tl
      {\tl_item:Nn \l__tilings_tmpc_tl {\int_use:N \l__tilings_tmpb_int}}
      \tl_set:Nx \l__tilings_tmpb_tl
      {\tl_item:Nn \l__tilings_tmpc_tl {\int_use:N \l__tilings_tmpb_int + 1}}
```

Possibly swap the ends.

```
      \bool_if:NT #1
      {
        \tl_set:NV \l__tilings_tmpc_tl \l__tilings_tmpa_tl
        \tl_set:NV \l__tilings_tmpa_tl \l__tilings_tmpb_tl
        \tl_set:NV \l__tilings_tmpb_tl \l__tilings_tmpc_tl
      }
```

Get the coordinates of the first point, which will be the origin of the transformation.

```
      \fp_set:Nn \l__tilings_xa_fp {\tl_item:Nn \l__tilings_tmpb_tl {1}}
      \fp_set:Nn \l__tilings_ya_fp {\tl_item:Nn \l__tilings_tmpb_tl {2}}
```

Get the coordinates of the second point and adjust relative to the first.

```
      \fp_set:Nn \l__tilings_xb_fp
      {\tl_item:Nn \l__tilings_tmpa_tl {1} - \l__tilings_xa_fp}
      \fp_set:Nn \l__tilings_yb_fp
      {\tl_item:Nn \l__tilings_tmpa_tl {2} - \l__tilings_ya_fp}
```

And normalise the vector along it.

```
% \fp_set:Nn \l__tilings_xb_fp {\l__tilings_xb_fp / \c__tilings_cm_fp}
% \fp_set:Nn \l__tilings_yb_fp {\l__tilings_yb_fp / \c__tilings_cm_fp}
      \fp_set:Nn \l__tilings_tmpa_fp
      {(\l__tilings_xb_fp)^2 + (\l__tilings_yb_fp)^2}
      \fp_set:Nn \l__tilings_xb_fp { \l__tilings_xb_fp / \l__tilings_tmpa_fp}
      \fp_set:Nn \l__tilings_yb_fp { \l__tilings_yb_fp / \l__tilings_tmpa_fp}
```

Now rotate so that the x–axis lies along the edge.

```
      \tl_gset:Nx \g__tilings_output_tl
      {
        \exp_not:N \pgftransformtriangle
        {
          \exp_not:N \pgfpoint{0pt}{0pt}
        }
        {
          \exp_not:N \pgfpoint
          {\fp_to_dim:N \l__tilings_xb_fp}{\fp_to_dim:n {-\l__tilings_yb_fp}}
        }
        {
          \exp_not:N \pgfpoint
          {\fp_to_dim:N \l__tilings_yb_fp}{\fp_to_dim:N \l__tilings_xb_fp}
        }
        \exp_not:N \pgftransformshift
        {
          \exp_not:N \pgfpoint
          {
            \fp_to_dim:n {-\l__tilings_xa_fp * \c__tilings_cm_fp}
          }
          {
            \fp_to_dim:n {-\l__tilings_ya_fp * \c__tilings_cm_fp}
          }
        }
      }
    }
    {
      \msg_error:nnxxx {tilings} {tile no edge} {#2} {#3}
      {\tl_use:N  \l__tilings_tmpc_tl }
      \tl_gclear:N \g__tilings_output_tl
    }
  }
  {
    \msg_error:nnn {tilings} {no tile} {#2}
    \tl_gclear:N \g__tilings_output_tl
  }
  \group_end:
  \tl_use:N \g__tilings_output_tl
  \tl_gclear:N \g__tilings_output_tl
}
```

(*End definition for* `\__tilings_transform_side_to_axis:Nnn`.)

```
\cs_generate_variant:Nn
\__tilings_transform_side_to_axis:Nnn {Nnx,NnV,NVV}
```

`ex_to_origin:Nnn` Apply a transformation to make a given vertex sit at the origin. Second argument is the tile, third is the side, first is a boolean to determine whether to use the start or end.

```
\cs_new_nopar:Npn \__tilings_translate_vertex_to_origin:Nnn #1#2#3
{
```

Get our tile data, checking if the tile exists.

```
  \group_begin:
  \prop_get:NnNTF \g__tilings_tiles_prop {#2} \l__tilings_tmpa_tl
  {
```

Start with the edge list.

Initialise the counter.

```
  \int_zero:N \l__tilings_tmpb_int
  \int_incr:N \l__tilings_tmpb_int
```

Get the path type list.

```
  \tl_set:Nx \l__tilings_tmpc_tl {\tl_head:N \l__tilings_tmpa_tl}
```

Iterate through the path type list, looking for the requested path.

```
    \bool_set_false:N \l__tilings_edge_bool
    \tl_map_inline:Nn \l__tilings_tmpc_tl {
      \str_if_eq:nnT {##1} {#3} {
        \bool_set_true:N \l__tilings_edge_bool
        \tl_map_break:
      }
      \int_incr:N \l__tilings_tmpb_int
    }
    \bool_if:NTF \l__tilings_edge_bool
    {
```

Get the coordinate list.

```
      \tl_set:Nx \l__tilings_tmpc_tl {\tl_tail:N \l__tilings_tmpa_tl}
```

Strip off the outer braces.

```
      \tl_set:Nx \l__tilings_tmpc_tl {\tl_item:Nn \l__tilings_tmpc_tl {1}}
```

Add the first coordinate at the end.

```
      \tl_put_right:Nx \l__tilings_tmpc_tl
      {{\tl_item:Nn \l__tilings_tmpc_tl {1}}}
```

Get the coordinates for this edge.

```
      \tl_set:Nx \l__tilings_tmpa_tl
      {\tl_item:Nn \l__tilings_tmpc_tl {\int_use:N \l__tilings_tmpb_int}}
      \tl_set:Nx \l__tilings_tmpb_tl
      {\tl_item:Nn \l__tilings_tmpc_tl {\int_use:N \l__tilings_tmpb_int + 1}}
```

Possibly swap the ends.

```
      \bool_if:NT #1
      {
        \tl_set:NV \l__tilings_tmpc_tl \l__tilings_tmpa_tl
        \tl_set:NV \l__tilings_tmpa_tl \l__tilings_tmpb_tl
        \tl_set:NV \l__tilings_tmpb_tl \l__tilings_tmpc_tl
      }
```

Get the coordinates of the first point, which will be the origin of the transformation.

```
      \fp_set:Nn \l__tilings_xa_fp {\tl_item:Nn \l__tilings_tmpb_tl {1}}
      \fp_set:Nn \l__tilings_ya_fp {\tl_item:Nn \l__tilings_tmpb_tl {2}}
```

Shift to place the selected vertex at the origin.

```
      \tl_gset:Nx \g__tilings_output_tl
      {
        \exp_not:N \pgftransformshift
        {
          \exp_not:N \pgfpoint
          {
            \fp_to_dim:n {-\l__tilings_xa_fp * \c__tilings_cm_fp}
          }
          {
            \fp_to_dim:n {-\l__tilings_ya_fp * \c__tilings_cm_fp}
          }
        }
      }
    }
    {
      \msg_error:nnxxx {tilings} {tile no edge} {#2} {#3}
      {\tl_use:N  \l__tilings_tmpc_tl }
      \tl_gclear:N \g__tilings_output_tl
    }
  }
  {
    \msg_error:nnn {tilings} {no tile} {#2}
    \tl_gclear:N \g__tilings_output_tl
  }
  \group_end:
  \tl_use:N \g__tilings_output_tl
  \tl_gclear:N \g__tilings_output_tl
}
```

(*End definition for* `\__tilings_translate_vertex_to_origin:Nnn`.)

```
\cs_generate_variant:Nn
\__tilings_translate_vertex_to_origin:Nnn {Nnx,NnV,NVV}
```

rmAlongSide Make this available outside the LaTeX3 environment. The starred version allows for reversing the side.

```
\DeclareDocumentCommand \TransformAlongSide {s m m}
{
```

Store the star

```
  \IfBooleanTF {#1}
  {
    \bool_set_true:N \l__tilings_cw_bool
  }
  {
    \bool_set_false:N \l__tilings_cw_bool
  }
  \__tilings_transform_side_to_axis:Nnx \l__tilings_cw_bool {#2}{#3}
}
```

(*End definition for* `\TransformAlongSide`.)

es_at_vertices:n This places TikZ coordinates at the vertices of the tile.

```
\cs_new_nopar:Npn \__tilings_coordinates_at_vertices:n #1
{
  \group_begin:
```

Get our tile data

```
  \prop_get:NnN \g__tilings_tiles_prop {#1} \l__tilings_tmpa_tl
```

Start with the edge list

```
  \tl_set:Nx \l__tilings_tmpb_tl {\tl_head:N \l__tilings_tmpa_tl}
```

Get the coordinate list

```
  \tl_set:Nx \l__tilings_tmpc_tl {\tl_tail:N \l__tilings_tmpa_tl}
```

Strip off the outer braces

```
  \tl_set:Nx \l__tilings_tmpc_tl
  {\tl_item:Nn \l__tilings_tmpc_tl {1}}
```

Add the first coordinate at the end

```
  \tl_put_right:Nx \l__tilings_tmpc_tl
  {{\tl_item:Nn \l__tilings_tmpc_tl {1}}}
```

Get the first coordinate

```
  \tl_set:Nx \l__tilings_tmpa_tl {\tl_head:N \l__tilings_tmpc_tl}
  \tl_set:Nx \l__tilings_tmpc_tl {\tl_tail:N \l__tilings_tmpc_tl}
```

Iterate through the edge list, placing coordinates

```
  \tl_map_inline:Nn \l__tilings_tmpb_tl {
    \tl_set:Nx \l__tilings_tmpd_tl {
      \exp_not:N \coordinate
      (-edge~ ##1~ start)~
      at (
      \tl_item:Nn \l__tilings_tmpa_tl {1},
      \tl_item:Nn \l__tilings_tmpa_tl {2}
      );
    }
    \tl_use:N \l__tilings_tmpd_tl
    \tl_set:Nx \l__tilings_tmpa_tl {\tl_head:N \l__tilings_tmpc_tl}
    \tl_set:Nx \l__tilings_tmpc_tl {\tl_tail:N \l__tilings_tmpc_tl}
    \tl_set:Nx \l__tilings_tmpd_tl {
      \exp_not:N \coordinate
      (-edge~ ##1~ end)~
      at (
      \tl_item:Nn \l__tilings_tmpa_tl {1},
      \tl_item:Nn \l__tilings_tmpa_tl {2}
      );
    }
    \tl_use:N \l__tilings_tmpd_tl
  }
  \group_end:
}
```

(*End definition for* `\__tilings_coordinates_at_vertices:n`.)

`sAtVertices`

User-accessible wrapper around the above.

```
\DeclareDocumentCommand \CoordinatesAtVertices {m}
{
  \__tilings_coordinates_at_vertices:n {#1}
}
```

(*End definition for* `\CoordinatesAtVertices`.)

```
\tikzset{
  transform~ to~ tile/.code~ args={#1~ along~ #2}{%
    \group_begin:
    \tl_if_in:nnTF {#1} {back}
    {
      \tikzset{
        tiling/alignment~ set~ location=#1,
        tiling/alignment~ direction={backwards}
      }
    }
    {
      \tikzset{
        tiling/alignment~ location=#1,
        tiling/alignment~ direction={forewards}
      }
    }
    \tl_if_in:nnTF {#2} {using}
    {
      \tikzset{
        tiling/alignment~ set~ edges=#2,
      }
    }
    {
      \tikzset{
        tiling/alignment~ edge=#2,
      }
    }
    \tikz_scan_point:n {
      (\__tilings_keys_get:n {alignment~ location}
      -edge~ \__tilings_keys_get:n {alignment~ edge}~ start)
    }
    \dim_set_eq:Nc \l__tilings_xa_dim {pgf@x}
    \dim_set_eq:Nc \l__tilings_ya_dim {pgf@y}
    \tikz_scan_point:n {
      (\__tilings_keys_get:n {alignment~ location}
      -edge~ \__tilings_keys_get:n {alignment~ edge}~ end)
    }
    \dim_set_eq:Nc \l__tilings_xb_dim {pgf@x}
    \dim_set_eq:Nc \l__tilings_yb_dim {pgf@y}
    \__tilings_keys_get:Nn \l__tilings_tmpb_tl {alignment~ direction}
    \tl_if_eq:NnTF \l__tilings_tmpb_tl {forewards}
    {
      \dim_gset_eq:NN \g__tilings_xa_dim \l__tilings_xa_dim
      \dim_gset_eq:NN \g__tilings_ya_dim \l__tilings_ya_dim
      \dim_gset_eq:NN \g__tilings_xb_dim \l__tilings_xb_dim
      \dim_gset_eq:NN \g__tilings_yb_dim \l__tilings_yb_dim
    }
    {
      \dim_gset_eq:NN \g__tilings_xa_dim \l__tilings_xb_dim
      \dim_gset_eq:NN \g__tilings_ya_dim \l__tilings_yb_dim
      \dim_gset_eq:NN \g__tilings_xb_dim \l__tilings_xa_dim
      \dim_gset_eq:NN \g__tilings_yb_dim \l__tilings_ya_dim
    }
    \dim_gsub:Nn \g__tilings_xb_dim {\g__tilings_xa_dim}
    \dim_gsub:Nn \g__tilings_yb_dim {\g__tilings_ya_dim}
    \dim_gset:Nn \g__tilings_xb_dim
    {\g__tilings_xb_dim * \dim_ratio:nn {1pt}{1cm}}
    \dim_gset:Nn \g__tilings_yb_dim
    {\g__tilings_yb_dim * \dim_ratio:nn {1pt}{1cm}}
    \group_end:
```

We store the initial points in `\pgf@xa` and `\pgf@ya` but we want `\pgf@xb` and `\pgf@yb` to be a vector along the edge.

We shift to the start of the edge.

```
    \pgftransformshift{\pgfpoint{\g__tilings_xa_dim}{\g__tilings_ya_dim}}
```

Now rotate so that the x–axis lies along the edge.

```
    \pgftransformtriangle
    {\pgfpoint{0pt}{0pt}}
    {\pgfpoint{\g__tilings_xb_dim}{\g__tilings_yb_dim}}
    {\pgfpoint{-\g__tilings_yb_dim}{\g__tilings_xb_dim}}
```

```
},
align~ with/.code~ args={#1~ along~ #2}{%
  \tl_if_in:nnTF {#1} {back}
  {
    \tikzset{
      tiling/alignment~ set~ location=#1,
      tiling/alignment~ direction={backwards}
    }
  }
  {
    \tikzset{
      tiling/alignment~ location=#1,
      tiling/alignment~ direction={forewards}
    }
  }
  \tl_if_in:nnTF {#2} {using}
  {
    \tikzset{
      tiling/alignment~ set~ edges=#2,
    }
  }
  {
    \tikzset{
      tiling/alignment~ edge=#2,
    }
  }
  \tikz_node_if_defined:nTF
  {
    \__tilings_keys_get:n {alignment~ location}
    -edge~ \__tilings_keys_get:n {alignment~ edge}~ start
  }
  {
    \tikzset{
      tiling/alignment~ start/.expanded={
        (\__tilings_keys_get:n {alignment~ location}
        -edge~ \__tilings_keys_get:n {alignment~ edge}~ start)
      },
      tiling/alignment~ end/.expanded={
        (\__tilings_keys_get:n {alignment~ location}
        -edge~ \__tilings_keys_get:n {alignment~ edge}~ end)
      },
    }
  }
  {
    \__tilings_keys_get:Nn \l__tilings_tmpa_tl {alignment~ location}
    \tl_set:Nx \l__tilings_tmpa_tl {\tl_use:N \l__tilings_tmpa_tl}
    \prop_get:NVNTF \g__tilings_tilenames_prop
    \l__tilings_tmpa_tl \l__tilings_tmpb_tl
    {
      \prop_get:NVN \g__tilings_tiles_prop
      \l__tilings_tmpb_tl \l__tilings_tmpc_tl
      \msg_error:nnxxx { tilings }{ tile no edge }
      {
        \tl_use:N \l__tilings_tmpa_tl \c_space_tl
        (type~ \tl_use:N \l__tilings_tmpb_tl)
      }
      {\__tilings_keys_get:n {alignment~ edge} }
      { \tl_item:Nn \l__tilings_tmpc_tl {1} }
    }
    {
      \msg_error:nnx { tilings }{ no tile }
```

```
        {\__tilings_keys_get:n {alignment~ location} }
      }
    }
  },
  tiling/.is~ family,
  tiling/alignment~ set~ location/.code~ args={#1~ back}{
    \tikzset{
      tiling/alignment~ location=#1,
    }
  },
  tiling/alignment~ set~ edges/.code~ args={#1~ using~ #2}{
    \tikzset{
      tiling/alignment~ edge=#1,
      tiling/alignment~ new~ edge=#2
    }
  },
  align~ between/.code~ args={#1~ and~ #2~ using~ #3}{
    \tikzset{
      tiling/alignment~ start={#1},
      tiling/alignment~ end={#2},
    }
    \str_set:Nn \l__tilings_tmpa_str {#3}
    \str_set:Nx \l__tilings_tmpb_str {\str_tail:N \l__tilings_tmpa_str}
    \tikzset{
      tiling/alignment~ new~ edge/.expanded={\str_use:N \l__tilings_tmpb_str}
    }
    \str_set:Nx \l__tilings_tmpa_str {\str_head:N \l__tilings_tmpa_str}
    \str_set:Nx \l__tilings_tmpb_str {\str_lowercase:f { \l__tilings_tmpa_str}}
    \str_if_eq:NNT \l__tilings_tmpa_str \l__tilings_tmpb_str
    {
      \str_set:Nx \l__tilings_tmpb_str
      {\str_uppercase:f { \l__tilings_tmpa_str}}
    }
    \tikzset{
      tiling/alignment~ edge/.expanded={\str_use:N \l__tilings_tmpb_str},
    }
  },
  tiling/alignment~ location/.initial={},
  tiling/alignment~ edge/.initial=a,
  tiling/alignment~ new~ edge/.initial={},
  tiling/alignment~ direction/.initial={forewards},
  tiling/alignment~ start/.initial={},
  tiling/alignment~ end/.initial={},
  tiling/anchor/.initial={},
```

Default clipping style.

```
  every~ tile~ clip/.style={clip}
}
```

`\DefineTile` This is the user function for defining a tile.

```
\DeclareDocumentCommand \DefineTile { s m m m }
{
```

Clear the temporary variable.

```
  \tl_clear:N \l__tilings_tmpa_tl
```

The 3rd parameter is a list of coordinates at vertices, iterate through them and add them to the list.

```
  \int_zero:N \l__tilings_tmpa_int
  \fp_zero:N \l__tilings_saved_x_fp
  \fp_zero:N \l__tilings_saved_y_fp
  \tl_map_inline:nn {#4} {
    \str_set:Nn \l__tilings_tmpa_str {##1}
    \str_if_eq:VnTF \l__tilings_tmpa_str {+}
    {
      \int_incr:N \l__tilings_tmpa_int
    }
    {
      \int_case:nn {\l__tilings_tmpa_int}
      {
        {0} {
          \bool_set_false:N \l__tilings_relative_bool
          \bool_set_true:N \l__tilings_update_saved_bool
        }
        {1} {
          \bool_set_true:N \l__tilings_relative_bool
          \bool_set_false:N \l__tilings_update_saved_bool
        }
        {2} {
          \bool_set_true:N \l__tilings_relative_bool
          \bool_set_true:N \l__tilings_update_saved_bool
        }

      }
      \str_if_in:NnTF \l__tilings_tmpa_str {:}
      {
        \seq_set_split:NVV \l__tilings_tmpa_seq \c__tilings_colon_str \l__tilings_tmpa_str
        \__tilings_add_coordinate:Nnn \l__tilings_tmpa_tl
        {
          (\seq_item:Nn \l__tilings_tmpa_seq {2}) * cosd (\seq_item:Nn \l__tilings_tmpa_seq {1})
        }
        {
          (\seq_item:Nn \l__tilings_tmpa_seq {2}) * sind (\seq_item:Nn \l__tilings_tmpa_seq {1})
        }
      }
      {
        \seq_set_split:NVV \l__tilings_tmpa_seq \c__tilings_comma_str \l__tilings_tmpa_str
        \__tilings_add_coordinate:Nnn \l__tilings_tmpa_tl
        {
          (\seq_item:Nn \l__tilings_tmpa_seq {1})
        }
        {
          (\seq_item:Nn \l__tilings_tmpa_seq {2})
        }
      }
      \int_zero:N \l__tilings_tmpa_int
    }
  }
```

Now we make a list of the edge types (from the 2nd parameter), using a prop to keep track of whether an edge is repeated.

```
  \prop_clear:N \l__tilings_tmpa_prop
  \tl_map_inline:nn {#3} {
    \prop_if_in:NnTF \l__tilings_tmpa_prop {##1}
    {
      \prop_put:Nnn \l__tilings_tmpa_prop {##1} {1}
    }
    {
      \prop_put:Nnn \l__tilings_tmpa_prop {##1} {0}
    }
```

`\BakeTile` This is the user wrapper around the tile creation macros.

```
\cs_new_protected_nopar:Npn \__tilings_bake_tile:n #1
{
  \prop_get:NnN \g__tilings_tiles_prop {#1} \l__tilings_tmpa_tl
  \__tilings_make_tile:nV {#1} \l__tilings_tmpa_tl
}

\NewDocumentCommand \BakeTile {m}
{
  \__tilings_bake_tile:n {#1}
}
```

`\UseTile` This is the command that actually places a tile on the page. The first argument is optional and is for styling.

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

We need to transform the tile to correspond to the current transformation matrix. To ensure that we only transform the current tile, we clone it first.

```
  \tl_if_exist:cTF {g__tilings_tile_#2_tl}
  {
    \tl_set_eq:Nc \l__tilings_tmp_tile_path_tl {g__tilings_tile_#2_tl}
```

We get the current transformation to apply to this path.

```
    \pgfgettransform \l__tilings_tmpa_tl
```

Apply the transformation, protocol the path, and render it.

```
    \spath_transform:NV \l__tilings_tmp_tile_path_tl \l__tilings_tmpa_tl
    \spath_tikz_path:nV {#1} \l__tilings_tmp_tile_path_tl
  }
  {
    \msg_error:nnn { tilings }{ not baked }{#2}
  }
}

\NewDocumentCommand \UseTile {O{} m}
{
  \__tilings_use_tile:nn {#1}{#2}
}
```

`tiling␣path` This is a style for a user to take a path and make it into the path for one of the sides. It needs to store both that side and the reverse.

```
\tikzset{
  save~ tiling~ path/.code={
    \tikz@addmode{
```

Get the current path.

```
      \pgfsyssoftpath@getcurrentpath\l__tilings_tmpa_tl
```

Normalise the path and save.

```
      \__tilings_normalise_path:N \l__tilings_tmpa_tl
      \tl_gclear_new:c {g__tilings_side_#1_tl}
      \tl_gset_eq:cN {g__tilings_side_#1_tl} \l__tilings_tmpa_tl
```

Now create the reverse path. The name is the upper case version.

```
      \tl_set:Nx \l__tilings_tmpb_tl {\str_uppercase:n {#1}}
```

Reverse the path, and relocate to the interval $[0, 1]$.

```
      \spath_reverse:N \l__tilings_tmpa_tl
      \spath_transform:Nnnnnnn \l__tilings_tmpa_tl {-1} {0} {0} {-1} {1} {0}
      \tl_gclear_new:c {g__tilings_side_ \tl_use:N \l__tilings_tmpb_tl _tl}
      \tl_gset_eq:cN {g__tilings_side_ \tl_use:N \l__tilings_tmpb_tl _tl} \l__tilings_tmpa_tl
    }
  },
  clone~ tiling~ side~ path/.style~ 2~ args={
    spath/set~ name=tiling~ side,
    spath/clone~ global={#1}{#2}
  },
  flip~ tile/.code={
    \tl_set:Nn \l__tilings_tmpa_tl {#1}
    \tl_set:Nn \l__tilings_tmpb_tl {true}
    \bool_set:Nn \l__tilings_cw_bool {\tl_if_eq_p:NN \l__tilings_tmpa_tl \l__tilings_tmpb_tl}
  },
  flip~ tile/.default={true},
  spath/prefix/tiling~side/.style={
    spath/set~ prefix=g__tilings_side_,
  },
  spath/suffix/tiling~side/.style={
    spath/set~ suffix=_tl,
  },
  clone~ tile~ path/.style~ 2~ args={
    spath/set~ name=tiling~tile,
    spath/clone~ global={#1}{#2}
  },
  spath/prefix/tiling~tile/.style={
    spath/set~ prefix=g__tilings_tile_,
  },
  spath/suffix/tiling~tile/.style={
    spath/set~ suffix=_tl,
  },
  expand~ key/.code={
    \exp_args:NV \pgfkeysalso #1
  }
}
```

(*End definition for* `save tiling path`. *This function is documented on page* ??.)

2.5 Lindenmayer System

This is an implementation of the Lindenmayer System description of Penrose and other tilings as a way of generating tilings from a specific starting seed.

The implementation uses `props` to store *rules* and *actions*. The rules are used to expand the starting seed to a certain level, after which the actions are carried out. The syntax is based on the PGF library, but as we're already using LaTeX3 it is reimplemented in that.

`ke_lms:Nnnn` This creates the token list of actions, starting with the seed. The arguments are: a token list to store the result in, the name of the system, the number of iterations, and the initial state.

```
\cs_new_nopar:Npn \__tilings_make_lms:Nnnn #1#2#3#4
{
  \group_begin:
```

On the first time round, we start with the given seed.

```
  \tl_set:Nn \l__tilings_tmpb_tl {#4}
```

We repeat the specified number of times.

```
  \prg_replicate:nn {#3} {
```

Duplicate the current state.

```
    \tl_set_eq:NN \l__tilings_tmpa_tl \l__tilings_tmpb_tl
```

Clear the receiving token list.

```
    \tl_clear:N \l__tilings_tmpb_tl
```

Walk through the current list, appending to the receiving list according to the rules.

```
    \tl_map_inline:Nn \l__tilings_tmpa_tl
    {
```

If a rule exists, copy that.

```
      \tl_set:Nx \l__tilings_action_lms_tl {\tl_head:n {##1}}
      \tl_set:Nx \l__tilings_parameters_lms_tl {\tl_tail:n {##1}}
      \prop_if_in:cVTF {g__tilings_#2_lms_rule_prop} \l__tilings_action_lms_tl
      {
        \prop_get:cVN {g__tilings_#2_lms_rule_prop} \l__tilings_action_lms_tl \l__tilings_tmpc_tl
        \tl_put_right:Nx \l__tilings_tmpb_tl {\tl_use:N \l__tilings_tmpc_tl}
%         {\prop_item:cn {g__tilings_#2_lms_rule_prop} {##1} }
      }
      {
```

Otherwise, just copy the token.

```
        \tl_if_single:nTF {##1}
        {
          \tl_put_right:Nn \l__tilings_tmpb_tl {##1}
        }
        {
          \tl_put_right:Nn \l__tilings_tmpb_tl {{##1}}
        }
      }
    }
  }
```

We've done all this inside a group, now pass the result outside.

```
  \tl_set:Nn \l__tilings_tmpa_tl {
    \group_end:
    \tl_set:Nn #1
  }
  \tl_put_right:Nx \l__tilings_tmpa_tl {{\tl_use:N \l__tilings_tmpb_tl}}
  \tl_use:N \l__tilings_tmpa_tl
}
\cs_generate_variant:Nn \__tilings_make_lms:Nnnn {Nnnx}
```

(*End definition for* `\__tilings_make_lms:Nnnn`*.*)

voke_lms:nn

This carries out the actions specified by the resulting rules.

```
\cs_new_nopar:Npn \__tilings_invoke_lms:nn #1#2
{
  \group_begin:
```

Walk through the given list, carrying out the corresponding action if it exists. If not, look at the default. Otherwise, just do nothing.

```
  \tl_map_inline:nn {#1} {
    \tl_set:Nx \l__tilings_action_lms_tl {\tl_head:n {##1}}
    \tl_set:Nx \l__tilings_parameters_lms_tl {\tl_tail:n {##1}}
    \prop_if_in:cVTF {g__tilings_#2_lms_action_prop} \l__tilings_action_lms_tl
    {
      \prop_item:cV {g__tilings_#2_lms_action_prop} \l__tilings_action_lms_tl
    }
    {
      \prop_if_in:cVT {g__tilings_default_lms_action_prop} \l__tilings_action_lms_tl
      {
        \prop_item:cV {g__tilings_default_lms_action_prop} \l__tilings_action_lms_tl
      }
    }
  }
  \group_end:
}
\cs_generate_variant:Nn \__tilings_invoke_lms:nn {Vn}
```

(*End definition for* `\__tilings_invoke_lms:nn`*.*)

We need some parameters.

```
\dim_new:N \l__tilings_step_dim
\dim_set:Nn \l__tilings_step_dim {1cm}
```

These are the defaults, which will be used in all the rule sets.

```
\prop_new:N \g__tilings_default_lms_action_prop
\prop_gput:Nnn \g__tilings_default_lms_action_prop {[} {\group_begin:}
\prop_gput:Nnn \g__tilings_default_lms_action_prop {]} {\group_end:}
\prop_gput:Nnn \g__tilings_default_lms_action_prop {f}
{\pgftransformxshift{\l__tilings_step_dim}}
\prop_gput:Nnn \g__tilings_default_lms_action_prop {b}
{\pgftransformxshift{-\l__tilings_step_dim}}
```

Holds a list of the tiles that actually draw for each tile set

```
\prop_new:N \g__tilings_drawables_lms_prop
```

We keep track of the number of tiles.

```
\int_new:N \g__tilings_tile_int
\int_new:N \g__tilings_tiles_int
```

`ecomposition` This is the user macro to invoke the decomposition. The arguments are: optional styles, the name, number of iterations, and starting seed.

```
\cs_new_protected_nopar:Npn \__tilings_tiling_decomposition:nnnn #1#2#3#4
{
  \group_begin:
  \tikzset{
    every~ #2~ decomposition/.try,
    #1
  }
  \__tilings_make_lms:Nnnx \l__tilings_tmpa_tl {#2} {#3} {#4}
  \__tilings_count_lms:Vn \l__tilings_tmpa_tl {#2}
  \int_gzero:N \g__tilings_tile_int
  \__tilings_invoke_lms:Vn \l__tilings_tmpa_tl {#2}
  \group_end:
}
\cs_new_protected_nopar:Npn \__tilings_tiling_decomposition:nnn #1#2#3
{
  \__tilings_tiling_decomposition:nnnn {}{#1}{#2}{#3}
}
\cs_generate_variant:Nn \__tilings_tiling_decomposition:nnn {VVV}

\NewDocumentCommand \TilingDecomposition { O{} m m m }
{
 \__tilings_tiling_decomposition:nnnn {#1}{#2}{#3}{#4}
}

\tikzset{
  pics/decomposition/.style~ n~ args={3}{
    code={
      \__tilings_tiling_decomposition:nnn {#1}{#2}{#3}
    }
  }
}
```

`ount_lms:nn` This counts the number of tiles in the string.

```
\cs_new_nopar:Npn \__tilings_count_lms:nn #1#2
{
  \group_begin:
  \int_gzero:N \g__tilings_tiles_int
  \prop_get:NnNT \g__tilings_drawables_lms_prop {#2} \l__tilings_tmpa_tl
  {
    \tl_map_variable:nNn {#1} \l__tilings_tmpb_tl
    {
      \tl_set:Nx \l__tilings_tmpb_tl {\tl_head:N \l__tilings_tmpb_tl}
      \bool_do_while:nn
      {
        !\tl_if_empty_p:N \l__tilings_tmpb_tl
        &&
        \tl_if_head_is_group_p:V \l__tilings_tmpb_tl
      }
      {
        \tl_set:Nx \l__tilings_tmpb_tl {\tl_head:N \l__tilings_tmpb_tl}
      }
      \tl_if_in:NVT \l__tilings_tmpa_tl \l__tilings_tmpb_tl
      {
        \int_gincr:N \g__tilings_tiles_int
      }
    }
  }
  \group_end:
}
\cs_generate_variant:Nn \__tilings_count_lms:nn {Vn}
```

(*End definition for* `\__tilings_count_lms:nn`.)

This is a `\tikzset` mechanism for setting the dimensions of the tiling.

```
\tikzset{
  tiling~ step/.code={
    \dim_set:Nn \l__tilings_step_dim {#1}
  }
}
```

```
\ExplSyntaxOff
```

```
\RequirePackage{tikz}
\usetikzlibrary{tilings}
\ProvidesFile {tikzlibrarytilings.penrose.code.tex}
[2023/06/01 v2.0 TikZ pics for Penrose tiles]
```

Create the pre-defined tile shapes.

- Thin Rhombus.

```
    \DefineTile{thin rhombus}{a A B b}
    {
     {0 , 0}
     {cosd(18) , sind(18)}
     {2*cosd(18) , 0}
     {cosd(18) , -sind(18)}
    }
```

- Thick Rhombus.

```
    \DefineTile{thick rhombus}{B a A b}
    {
     {0 , 0}
     {cosd(36) , sind(36)}
     {2*cosd(36) , 0}
     {cosd(36) , -sind(36)}
    }
```

- Dart.

```
\DefineTile{dart}{c a A C}
{
 {0 , 0}
 {2*sind(18)*cosd(108) , 2*sind(18)*sind(108)}
 {2*sind(18) , 0}
 {2*sind(18)*cosd(108) , -2*sind(18)*sind(108)}
}
```

- Kite.

```
\DefineTile{kite}{a c C A}
{
 {0 , 0}
 {cosd(36) , sind(36)}
 {1 , 0}
 {cosd(36) , -sind(36)}
}
```

- Golden Triangle.

```
\DefineTile{golden triangle}{a c b}
{
 {0 , 0}
 {cosd(18) , sind(18)}
 {cosd(18) , -sind(18)}
}
```

- Reverse Golden Triangle.

```
\DefineTile {reverse golden triangle}{B C A}
{
 {0 , 0}
 {cosd(18) , sind(18)}
 {cosd(18) , -sind(18)}
}
```

- Golden Gnomon

```
\DefineTile {golden gnomon}{C b A}
{
 {0 , 0}
 {cosd(36) , sind(36)}
 {2*cosd(36) , 0}
}
```

- Reverse Golden Gnomon

```
\DefineTile {reverse golden gnomon}{a B c}
{
 {0 , 0}
 {2*cosd(36) , 0}
 {cosd(36) , -sind(36)}
}
```

- Primary Pentagon (pentagon 5)

```
\DefineTile {pentagon 5}{a a a a a}
{
 {0 , 0}
 {cosd(108) , sind(108)}
 {1+cosd(72)+cosd(144) , sind(72)+sind(144)}
 {1+cosd(72) , sind(72)}
 {1 , 0}
}
```

- Secondary Pentagon (pentagon 3)

```
\DefineTile {pentagon 3}{A b a a b}
{
 {0 , 0}
 {cosd(108) , sind(108)}
 {1+cosd(72)+cosd(144) , sind(72)+sind(144)}
 {1+cosd(72) , sind(72)}
 {1 , 0}
}
```

- Tertiary Pentagon (pentagon 2)

```
\DefineTile {pentagon 2}{d A e c A}
{
 {0 , 0}
 {cosd(108) , sind(108)}
 {1+cosd(72)+cosd(144) , sind(72)+sind(144)}
 {1+cosd(72) , sind(72)}
 {1 , 0}
}
```

- Pentagram

```
\DefineTile {pentagram}{C E C E C E C E C E}
{
  {1 , 0}
  {1-cosd(36) , -sind(36)}
  {1-cosd(36)-cosd(108) , -sind(36)-sind(108)}
  {cosd(108) , -sind(108)}
  {-1+3*cosd(108)+cosd(36) , -sind(36)-sind(108)}
  {-1+2*cosd(108)+cosd(36) , -sind(36)}
  {-1+2*cosd(108) , 0}
  {2*cosd(108) , 0}
  {cosd(108) , sind(108)}
  {0 , 0}
}
```

- Boat

```
\DefineTile {boat}{C E C E B D B}
{
 {-1+2*cosd(108) , 0}
 {2*cosd(108) , 0}
 {cosd(108) , sind(108)}
 {0 , 0}
 {1 , 0}
 {1-cosd(36) , -sind(36)}
 {-1+2*cosd(108)+cosd(36) , -sind(36)}
}
```

- Diamond.

```
     \DefineTile {diamond}{D B B D}
     {
      {0 , 0}
      {cosd(18) , sind(18)}
      {2*cosd(18) , 0}
      {cosd(18) , -sind(18)}
     }
```

Place the arcs

```
\tikzset{
  every thin rhombus before path/.code={
    \path[every circle arc/.try] (-edge a end) circle[radius=1/4];
    \path[every long arc/.try] (-edge b start) circle[radius=1/4];
  },
  every thick rhombus before path/.code={
    \path[every circle arc/.try] (-edge a end) circle[radius=1/4];
    \path[every long arc/.try] (-edge B start) circle[radius=3/4];
  },
  every kite before path/.code={
    \path[every circle arc/.try] (-edge a start) circle[radius=2/(sqrt(5)+1)];
    \path[every long arc/.try] (-edge c end) circle[radius=2/(3+sqrt(5))];
  },
  every dart before path/.code={
    \path[every circle arc/.try] (-edge a end) circle[radius=1 - 2/(sqrt(5)+1)];
    \path[every long arc/.try] (-edge c start) circle[radius=2/(sqrt(5)+1) - 2/(3+sqrt(5))];
  }
}
```

Now bake the tiles.

```
\BakeTile {thin rhombus}
\BakeTile {thick rhombus}
\BakeTile {dart}
\BakeTile {kite}
\BakeTile {golden triangle}
\BakeTile {reverse golden triangle}
\BakeTile {golden gnomon}
\BakeTile {reverse golden gnomon}
\BakeTile {pentagon 5}
\BakeTile {pentagon 3}
\BakeTile {pentagon 2}
\BakeTile {pentagram}
\BakeTile {boat}
\BakeTile {diamond}
```

2.6 Lindenmayer System

```
\ExplSyntaxOn
```

These are the rules for generating rhombus tilings with the Lindenmayer System procedure.

```
\prop_new:N \g__tilings_rhombus_lms_rule_prop
\prop_gput:Nnn \g__tilings_rhombus_lms_rule_prop {T} {[f*sT][f>g]}
\prop_gput:Nnn \g__tilings_rhombus_lms_rule_prop {t} {[f_st][f>G]}
\prop_gput:Nnn \g__tilings_rhombus_lms_rule_prop {G} {[f+sG][sf>g][sf*sT]}
\prop_gput:Nnn \g__tilings_rhombus_lms_rule_prop {g} {[f-sg][sf>G][sf_st]}
```

These are the rules for generating kite and dart tilings.

```
\prop_new:N \g__tilings_kite_lms_rule_prop
\prop_gput:Nnn \g__tilings_kite_lms_rule_prop {T} {[f*sT][f>st][+sg]}
\prop_gput:Nnn \g__tilings_kite_lms_rule_prop {t} {[f_st][f>sT][-sG]}
\prop_gput:Nnn \g__tilings_kite_lms_rule_prop {G} {[f*+sG][sT]}
\prop_gput:Nnn \g__tilings_kite_lms_rule_prop {g} {[f-_sg][st]}
```

These are the rules for generating pentagon tilings.

```
\prop_new:N \g__tilings_pentagon_lms_rule_prop
\prop_gput:Nnn \g__tilings_pentagon_lms_rule_prop {P}
{[s>P][1sF+Q][1+sF+Q][1*sF+Q][1-sF+Q][1_sF+Q]} % pentagon 5
\prop_gput:Nnn \g__tilings_pentagon_lms_rule_prop {Q}
{[s>P][1+sFR][1*sF*R][1-sF+Q][1_sF+Q][1sF+Q][->fsD]} % pentagon 3
\prop_gput:Nnn \g__tilings_pentagon_lms_rule_prop {R}
{[s>P][1-sF+Q][1+sF*R][1*sFR][1_sF*R][1sFR][_>fsD][>fsD]} % pentagon 2
\prop_gput:Nnn \g__tilings_pentagon_lms_rule_prop {G}
{
  [s>G]
  [se[>d+R][e1B]]
  [+se[>d+R][e1B]]
  [-se[>d+R][e1B]]
  [*se[>d+R][e1B]]
  [_se[>d+R][e1B]]
} % pentagram
\prop_gput:Nnn \g__tilings_pentagon_lms_rule_prop {B}
{
  [s>G]
  [se[>d+R][e1B]]
  [+se[>d+R][e1B]]
  [-se[>d+R][e1B]]
} % boat
\prop_gput:Nnn \g__tilings_pentagon_lms_rule_prop {D}
{[s>d+R][s>eG][se1B]} % diamond
```

Each of the standard tilings can also be drawn using triangles using the same rules.

```
\prop_gset_eq:NN \g__tilings_rtriangle_lms_rule_prop
\g__tilings_rhombus_lms_rule_prop
\prop_gset_eq:NN \g__tilings_ktriangle_lms_rule_prop
\g__tilings_kite_lms_rule_prop
```

These are the lists of tokens that actually draw things

```
\prop_gput:Nnn \g__tilings_drawables_lms_prop {rhombus} {TG}
\prop_gput:Nnn \g__tilings_drawables_lms_prop {kite} {Tg}
\prop_gput:Nnn \g__tilings_drawables_lms_prop {rtriangle} {TtGg}
\prop_gput:Nnn \g__tilings_drawables_lms_prop {ktriangle} {TtGg}
\prop_gput:Nnn \g__tilings_drawables_lms_prop {pentagon} {PQRGBD}
```

These hold the various actions.

```
\prop_new:N \g__tilings_rhombus_lms_action_prop
\prop_new:N \g__tilings_kite_lms_action_prop
\prop_new:N \g__tilings_rtriangle_lms_action_prop
\prop_new:N \g__tilings_ktriangle_lms_action_prop
\prop_new:N \g__tilings_pentagon_lms_action_prop
```

The rhombus rules need a variety of turns.

```
\prop_gput:Nnn \g__tilings_rhombus_lms_action_prop {+}
{\pgftransformrotate{144}}
\prop_gput:Nnn \g__tilings_rhombus_lms_action_prop {*}
{\pgftransformrotate{108}}
\prop_gput:Nnn \g__tilings_rhombus_lms_action_prop {-}
{\pgftransformrotate{216}}
\prop_gput:Nnn \g__tilings_rhombus_lms_action_prop {_}
{\pgftransformrotate{252}}
\prop_gput:Nnn \g__tilings_rhombus_lms_action_prop {>}
{\pgftransformrotate{180}}
\prop_gput:Nnn \g__tilings_rhombus_lms_action_prop {s} {
  \fp_set:Nn \l__tilings_tmpa_fp { 2 * sind(18) * \l__tilings_step_dim }
  \dim_set:Nn \l__tilings_step_dim {\fp_to_dim:N \l__tilings_tmpa_fp}
}
```

Up to now, the actions for the rhombus and its triangle replacement are the same.

```
\prop_gset_eq:NN \g__tilings_rtriangle_lms_action_prop
```

```
\g__tilings_rhombus_lms_action_prop
```

Now we do the actions that actually draw something.

```
\prop_gput:Nnn \g__tilings_rhombus_lms_action_prop {T} {
  \group_begin:
```

As we go through, we keep track of how many tiles we've drawn.

```
  \int_gincr:N \g__tilings_tile_int
```

Set up the position, size, and angle correctly.

```
  \pgftransformrotate{198}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim*2*cosd(18)}
  \pgftransformxshift{-\fp_to_dim:N \l__tilings_tmpa_fp}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
```

Now we draw the thin rhombus, applying every style we can possibly imagine. The `tile` style gets the current tile and total tile numbers passed to it.

```
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ thin~ rhombus/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{thin~rhombus}
  \group_end:
}
```

Same for the thick rhombus.

```
\prop_gput:Nnn \g__tilings_rhombus_lms_action_prop {G} {
  \group_begin:
  \int_gincr:N \g__tilings_tile_int
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)/(2*cosd(36))}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ thick~ rhombus/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{thick~rhombus}
  \group_end:
}
```

Now we do the same for the kite and dart tiling.

```
\prop_gput:Nnn \g__tilings_kite_lms_action_prop {+} {\pgftransformrotate{36}}
\prop_gput:Nnn \g__tilings_kite_lms_action_prop {*} {\pgftransformrotate{108}}
\prop_gput:Nnn \g__tilings_kite_lms_action_prop {-} {\pgftransformrotate{-36}}
\prop_gput:Nnn \g__tilings_kite_lms_action_prop {_}
{\pgftransformrotate{-108}}
\prop_gput:Nnn \g__tilings_kite_lms_action_prop {>} {\pgftransformrotate{180}}
\prop_gput:Nnn \g__tilings_kite_lms_action_prop {s} {
  \fp_set:Nn \l__tilings_tmpa_fp { 2 * sind(18) * \l__tilings_step_dim }
  \dim_set:Nn \l__tilings_step_dim {\fp_to_dim:N \l__tilings_tmpa_fp}
}
```

```
\prop_gset_eq:NN \g__tilings_ktriangle_lms_action_prop
\g__tilings_kite_lms_action_prop
```

```
\prop_gput:Nnn \g__tilings_kite_lms_action_prop {T} {
  \group_begin:
  \int_gincr:N \g__tilings_tile_int
  \pgftransformrotate{36}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ kite/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{kite}
  \group_end:
}
\prop_gput:Nnn \g__tilings_kite_lms_action_prop {g} {
  \group_begin:
  \int_gincr:N \g__tilings_tile_int
  \pgftransformrotate{144}
  \pgftransformxshift{-\l__tilings_step_dim * 2 * sin(18)}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ dart/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{dart}
  \group_end:
}
```

Now we set up the actions for the triangle variations.

```
\prop_gput:Nnn \g__tilings_rtriangle_lms_action_prop {T} {
  \group_begin:
  \int_gincr:N \g__tilings_tile_int
  \pgftransformrotate{18}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ reverse~ golden~ triangle/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{reverse~ golden~ triangle}
  \group_end:
}
\prop_gput:Nnn \g__tilings_rtriangle_lms_action_prop {t} {
  \group_begin:
  \int_gincr:N \g__tilings_tile_int
  \pgftransformrotate{-18}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ golden~ triangle/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{golden~ triangle}
  \group_end:
}
\prop_gput:Nnn \g__tilings_rtriangle_lms_action_prop {G} {
  \group_begin:
  \int_gincr:N \g__tilings_tile_int
  \pgftransformrotate{180}
  \pgftransformxshift{-\l__tilings_step_dim}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)/(2*cosd(36))}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ reverse~ golden~ gnomon/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{reverse~ golden~ gnomon}
  \group_end:
}
\prop_gput:Nnn \g__tilings_rtriangle_lms_action_prop {g} {
  \group_begin:
  \int_gincr:N \g__tilings_tile_int
  \pgftransformrotate{180}
  \pgftransformxshift{-\l__tilings_step_dim}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)/(2*cosd(36))}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ golden~ gnomon/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{golden~ gnomon}
  \group_end:
}
\prop_gput:Nnn \g__tilings_ktriangle_lms_action_prop {T} {
  \group_begin:
  \int_gincr:N \g__tilings_tile_int
  \pgftransformrotate{18}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ reverse~ golden~ triangle/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{reverse~ golden~ triangle}
  \group_end:
}
\prop_gput:Nnn \g__tilings_ktriangle_lms_action_prop {t} {
  \group_begin:
  \int_gincr:N \g__tilings_tile_int
  \pgftransformrotate{-18}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ golden~ triangle/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{golden~ triangle}
  \group_end:
}
\prop_gput:Nnn \g__tilings_ktriangle_lms_action_prop {G} {
  \group_begin:
  \int_gincr:N \g__tilings_tile_int
  \pgftransformrotate{180}
  \pgftransformxshift{-\l__tilings_step_dim}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)/(2*cosd(36))}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ reverse~ golden~ gnomon/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{reverse~ golden~ gnomon}
  \group_end:
}
```

```
\prop_gput:Nnn \g__tilings_ktriangle_lms_action_prop {g} {
  \group_begin:
  \int_gincr:N \g__tilings_tile_int
  \pgftransformrotate{180}
  \pgftransformxshift{-\l__tilings_step_dim}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)/(2*cosd(36))}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ golden~ gnomon/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{golden~ gnomon}
  \group_end:
}
```

Now we do the same for the pentagonal tilings.

The rules need a variety of turns.

```
\int_new:N \l__tilings_pentagon_parity_int
\seq_new:N \l__tilings_pentagon_parity_seq
\seq_set_from_clist:Nn \l__tilings_pentagon_parity_seq {odd,even}
\prop_gput:Nnn \g__tilings_pentagon_lms_action_prop {1} {
  \int_set:Nn \l__tilings_pentagon_parity_int
  {3 -  \l__tilings_pentagon_parity_int}
}
\tikzset{
  every~ pentagon~ decomposition/.code={%
    \int_set:Nn \l__tilings_pentagon_parity_int {2}
  }
}
\prop_gput:Nnn \g__tilings_pentagon_lms_action_prop {+}
{\pgftransformrotate{72}}
\prop_gput:Nnn \g__tilings_pentagon_lms_action_prop {*}
{\pgftransformrotate{144}}
\prop_gput:Nnn \g__tilings_pentagon_lms_action_prop {-}
{\pgftransformrotate{288}}
\prop_gput:Nnn \g__tilings_pentagon_lms_action_prop {_}
{\pgftransformrotate{216}}
\prop_gput:Nnn \g__tilings_pentagon_lms_action_prop {>}
{\pgftransformrotate{180}}
\prop_gput:Nnn \g__tilings_pentagon_lms_action_prop {|}
{\pgftransformxscale{-1}}
```

The scale factor is different.

```
\prop_gput:Nnn \g__tilings_pentagon_lms_action_prop {s} {
  \fp_set:Nn \l__tilings_tmpa_fp
  {
    1/(2 + 2 * cosd(72) ) * \l__tilings_step_dim
  }
  \dim_set:Nn \l__tilings_step_dim {\fp_to_dim:N \l__tilings_tmpa_fp}
}
```

And we tend to work better vertically.

```
\prop_gput:Nnn \g__tilings_pentagon_lms_action_prop {f} {
  \fp_set:Nn \l__tilings_tmpa_fp {  tand(54)/2 * \l__tilings_step_dim }
  \pgftransformyshift{\fp_to_dim:N \l__tilings_tmpa_fp}
}
\prop_gput:Nnn \g__tilings_pentagon_lms_action_prop {F} {
  \fp_set:Nn \l__tilings_tmpa_fp {  tand(54) * \l__tilings_step_dim }
  \pgftransformyshift{\fp_to_dim:N \l__tilings_tmpa_fp}
}
\prop_gput:Nnn \g__tilings_pentagon_lms_action_prop {d} {
  \fp_set:Nn \l__tilings_tmpa_fp
  {
    (tand(54)/2 - tand(72)/2 + sind(36) ) * \l__tilings_step_dim
  }
  \pgftransformyshift{\fp_to_dim:N \l__tilings_tmpa_fp}
}
\prop_gput:Nnn \g__tilings_pentagon_lms_action_prop {e} {
  \fp_set:Nn \l__tilings_tmpa_fp
  {
    tand(54) * cosd(36) * \l__tilings_step_dim
  }
  \pgftransformyshift{\fp_to_dim:N \l__tilings_tmpa_fp}
}
\prop_gput:Nnn \g__tilings_pentagon_lms_action_prop {P} {
  \group_begin:
  \int_gincr:N \g__tilings_tile_int
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/2}
  \pgftransformxshift{-\fp_to_dim:N \l__tilings_tmpa_fp}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim*tand(54)/2}
  \pgftransformyshift{-\fp_to_dim:N \l__tilings_tmpa_fp}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ pentagon/.try,
    every~
    \seq_item:Nn \l__tilings_pentagon_parity_seq
    {\l__tilings_pentagon_parity_int}
    \space pentagon/.try,
    every~ pentagon~ 5/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{pentagon~5}
  \group_end:
}
\prop_gput:Nnn \g__tilings_pentagon_lms_action_prop {Q} {
  \group_begin:
  \int_gincr:N \g__tilings_tile_int
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/2}
  \pgftransformxshift{-\fp_to_dim:N \l__tilings_tmpa_fp}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim*tand(54)/2}
  \pgftransformyshift{-\fp_to_dim:N \l__tilings_tmpa_fp}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ pentagon/.try,
    every~
    \seq_item:Nn \l__tilings_pentagon_parity_seq
    {\l__tilings_pentagon_parity_int}
    \space pentagon/.try,
    every~ pentagon~ 3/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{pentagon~3}
  \group_end:
}
\prop_gput:Nnn \g__tilings_pentagon_lms_action_prop {R} {
  \group_begin:
  \int_gincr:N \g__tilings_tile_int
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/2}
  \pgftransformxshift{-\fp_to_dim:N \l__tilings_tmpa_fp}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim*tand(54)/2}
  \pgftransformyshift{-\fp_to_dim:N \l__tilings_tmpa_fp}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ pentagon/.try,
    every~
    \seq_item:Nn \l__tilings_pentagon_parity_seq
    {\l__tilings_pentagon_parity_int}
    \space pentagon/.try,
    every~ pentagon~ 2/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{pentagon~2}
  \group_end:
}
\prop_gput:Nnn \g__tilings_pentagon_lms_action_prop {G} {
  \group_begin:
  \int_gincr:N \g__tilings_tile_int
%  \pgftransformrotate{198}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim*cosd(72)}
  \pgftransformxshift{\fp_to_dim:N \l__tilings_tmpa_fp}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim*tand(54)*cosd(72)}
  \pgftransformyshift{\fp_to_dim:N \l__tilings_tmpa_fp}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ pentagram/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{pentagram}
  \group_end:
}
```

```
\prop_gput:Nnn \g__tilings_pentagon_lms_action_prop {B} {
  \group_begin:
  \int_gincr:N \g__tilings_tile_int
%  \pgftransformrotate{198}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim*cosd(72)}
  \pgftransformxshift{\fp_to_dim:N \l__tilings_tmpa_fp}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim*tand(54)*cosd(72)}
  \pgftransformyshift{\fp_to_dim:N \l__tilings_tmpa_fp}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ boat/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{boat}
  \group_end:
}
```

```
\prop_gput:Nnn \g__tilings_pentagon_lms_action_prop {D} {
  \group_begin:
  \int_gincr:N \g__tilings_tile_int
  \pgftransformrotate{90}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim*cosd(18)}
  \pgftransformxshift{-\fp_to_dim:N \l__tilings_tmpa_fp}
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ diamond/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{diamond}
  \group_end:
}
```

```
\ExplSyntaxOff
```

```
\RequirePackage{tikz}
\usetikzlibrary{tilings}
\ProvidesFile {tikzlibrarytilings.polykite.code.tex}
 [2023/06/01 v2.0 TikZ pics for Aperiodical Polykite tiles]
```

`PolykiteTile` Define one of the family of polykite tiles. Needs a name and length parameters. A star option switches the side labels to enable the hat-turtle pairing of a tile with its “opposite”.

```
\DeclareDocumentCommand \DefinePolykiteTile {s m m m}
{
  \IfBooleanTF {#1}
  {
    \DefineTile {#2} {2 2 1 1 2 2 1 1 1 1 2 2 1 1}
  }
  {
    \DefineTile {#2} {1 1 2 2 1 1 2 2 2 2 1 1 2 2}
  }
  {
    {0 : #4}
    ++{90 : #3}
    ++{150 : #3}
    ++{240 : #4}
    ++{180 : #4}
    ++{-90 : #3}
    ++{210 : #3}
    ++{-60 : #4}
    ++{0 : #4}
    ++{0 : #4}
    ++{60 : #4}
    ++{-30 : #3}
    ++{30 : #3}
    ++{120 : #4}
  }
}
```

Some predefined tiles. The aperiodical hat and turtle can be swapped in for each other as they use the same edge definitions. The spectral hat and turtle are designed to be used in the same diagram. Technically, the spectral and aperiodical hats are the same, but I figured it better to have two names for the two uses. The spectre uses the alternating edge scheme.

- Aperiodical and Spectral Hat.

```
    \DefinePolykiteTile{aperiodical hat}{sqrt(3)/2}{1/2}
    \DefinePolykiteTile{spectral hat}{sqrt(3)/2}{1/2}
```

- Aperiodical and Spectral Turtles.

```
    \DefinePolykiteTile{aperiodical turtle}{1/2}{sqrt(3)/2}
    \DefinePolykiteTile*{spectral turtle}{1/2}{sqrt(3)/2}
```

- Aperiodical Spectre.

```
    \DefineTile {spectre} {a A a A a A a A a A a A a A}
      {
        {0 : 1}
        ++{90 : 1}
        ++{150 : 1}
        ++{240 : 1}
        ++{180 : 1}
        ++{-90 : 1}
        ++{210 : 1}
        ++{-60 : 1}
        ++{0 : 1}
        ++{0 : 1}
        ++{60 : 1}
        ++{-30 : 1}
        ++{30 : 1}
        ++{120 : 1}
      }
```

- Meta Clusters

  The mapping between the notation in the https://arxiv.org/abs/2303.10798 is:

$$\begin{aligned} A^{+}, A^{-} &\mapsto a, A \\ B^{+}, B^{-} &\mapsto b, B \\ F^{+}, F^{-} &\mapsto c, C \\ X^{+}, X^{-} &\mapsto d, D \\ L &\mapsto 1 \end{aligned}$$

```
\DefineTile{meta cluster T}{A A b}
{
{ -1.5 , -sqrt(3)/2 }
{  1.5 , -sqrt(3)/2 }
{  0   ,  sqrt(3) }
}
\DefineTile{meta cluster P}{1 D d A 1 D d b}
{
{ -2.5, sqrt(3)/2 }
{ -2, 0 }
{ -1.5, -sqrt(3)/2 }
{ -0.5 , -sqrt(3)/2 }
{  2.5 , -sqrt(3)/2 }
{  2 , 0 }
{ 1.5 , sqrt(3)/2 }
{ .5 , sqrt(3)/2 }
}
\DefineTile{meta cluster F}{1 D d 1 D c C d b}
{
{ -2.5, sqrt(3)/2 }
{ -2, 0 }
{ -1.5, -sqrt(3)/2 }
{ -0.5 , -sqrt(3)/2 }
{ .5 , -sqrt(3)/2 }
{ 1.5 , -sqrt(3)/2 }
{  2 , 0 }
{ 1.5 , sqrt(3)/2 }
{ .5 , sqrt(3)/2 }
}
\DefineTile{meta cluster H}{B D d B D d a D d}
{
{ -2, -sqrt(3) }
{ 1, -sqrt(3) }
{ 2, -sqrt(3) }
{ 2.5, -sqrt(3)/2 }
{ 1, sqrt(3)}
{ .5 , 3*sqrt(3)/2 }
{ -.5 , 3*sqrt(3)/2 }
{ -2, 0 }
{ -2.5, -sqrt(3)/2 }
}
```

- Super Clusters

```
\DefineTile{super cluster T}{A A b}
```

```
{
{ -30 : 3 * (1 + sqrt(5))/2 / sqrt(3) }
{ 90 : 3 * (1 + sqrt(5))/2 / sqrt(3) }
{ 210 : 3 * (1 + sqrt(5))/2 / sqrt(3) }
}
% 1 + 3phi, 1 + 2phi
\DefineTile{super cluster P}{1 D d A 1 D d b}
{
{ - 1.75 - sqrt(5), (sqrt(5)/2 + 1) * sqrt(3)/2 }
++{ -60 : 1 + sqrt(5) }
++{ -60 : 1}
++{ 1, 0}
++{ 3*(1 + sqrt(5))/2, 0 }
++{ 120 : 1 + sqrt(5) }
++{ 120 : 1 }
++{ -1, 0 }
}
\DefineTile{super cluster F}{1 D d 1 D c C d b}
{
{ - 1.75 - sqrt(5), (sqrt(5)/2 + 1) * sqrt(3)/2 }
++{ -60 : 1 + sqrt(5) }
++{ -60 : 1}
++{ 1, 0}
++{ 1 + sqrt(5), 0 }
++{ 1, 0 }
+{ -.75 + (2 + sqrt(5)) * sqrt(3)/4 * sqrt(3)/3,
  (2 + sqrt(5)) * sqrt(3)/4 + 3/4 * sqrt(3)/3 }
++{ -1.5, (2 + sqrt(5)) * sqrt(3)/2 }
++{ -1, 0 }
}
\DefineTile{super cluster H}{B D d B D d a D d}
{
{1.75 + 3*sqrt(5)/4, -(1 + sqrt(5))*sqrt(3)/4}
++{120 : 3*(1+sqrt(5))/2 }
++{120 : 1 }
++{-1,0}
++{240 : 3*(1+sqrt(5))/2}
++{240 : 1}
++{300 : 1}
++{ 3*(1+sqrt(5))/2, 0 }
++{1, 0}
++{60 : 1}
}
```

- Subclusters

```
\DefineTile{subcluster H}{B B a}
{
  { 0, 0 }
  { 3, 0 }
  { 60 : 3 }
}
\DefineTile{subcluster T}{A A b}
{
  { 0, 0 }
  { 3, 0 }
  { 60 : 3 }
}
\DefineTile{subcluster P}{ 1 A 1 b }
{
  { 0, 0 }
```

```
        { 1, 0 }
        { 4, 0 }
        { 3, 0 }
      }
      \DefineTile{subcluster F}{ 1 1 f F b }
      {
        { 0, 0 }
        { 1, 0 }
        +{ 60 : 1 }
        { 2, 0 }
        { 3, 0 }
      }
```

The P and F subclusters have no area, so clipping against them is not helpful.

```
\tikzset{
  no clip/.code={%
    \tikz@addmode{\tikz@mode@clipfalse}%
  },
  every subcluster P clip/.style={no clip},
  every subcluster F clip/.style={no clip},
}
```

```
\BakeTile {aperiodical hat}
\BakeTile {aperiodical turtle}
\BakeTile {spectral hat}
\BakeTile {spectral turtle}
\BakeTile {spectre}
\BakeTile {meta cluster T}
\BakeTile {meta cluster P}
\BakeTile {meta cluster F}
\BakeTile {meta cluster H}
\BakeTile {super cluster T}
\BakeTile {super cluster P}
\BakeTile {super cluster F}
\BakeTile {super cluster H}
```

The subclusters are deformed by default.

```
\ExplSyntaxOn
\clist_map_inline:nn {a,A,b,B,f,F}
{
  \tl_new:c {g__tilings_side_polykite_#1_tl}
  \tl_if_exist:cF {g__tilings_side_#1_tl}
  {
    \tl_new:c {g__tilings_side_#1_tl}
  }
}

\tl_gset:cn {g__tilings_side_polykite_A_tl}
{
  \pgfsyssoftpath@movetotoken {0pt}{-0.3333332942822268pt}
  \pgfsyssoftpath@linetotoken {0.0833331478405773pt}{-0.1889954840909892pt}
  \pgfsyssoftpath@linetotoken {0.3333332942822268pt}{-0.3333332942822268pt}
  \pgfsyssoftpath@linetotoken {0.5833333235705567pt}{-0.1889954840909892pt}
  \pgfsyssoftpath@linetotoken {0.6666667057177732pt}{-0.3333332942822268pt}
  \pgfsyssoftpath@linetotoken {1pt}{-0.3333332942822268pt}
}
\tl_gset:cn {g__tilings_side_polykite_a_tl}
{
  \pgfsyssoftpath@movetotoken {0pt}{0.33333pt}
  \pgfsyssoftpath@linetotoken {0.33333pt}{0.33333pt}
  \pgfsyssoftpath@linetotoken {0.41667pt}{0.189pt}
  \pgfsyssoftpath@linetotoken {0.66667pt}{0.33333pt}
  \pgfsyssoftpath@linetotoken {0.91667pt}{0.189pt}
  \pgfsyssoftpath@linetotoken {1pt}{0.33333pt}
}
\tl_gset:cn {g__tilings_side_polykite_B_tl}
{
  \pgfsyssoftpath@movetotoken {0pt}{0pt}
  \pgfsyssoftpath@linetotoken {0.3333332942822268pt}{0pt}
  \pgfsyssoftpath@linetotoken {0.4166665592761237pt}{0.1443378101912376pt}
  \pgfsyssoftpath@linetotoken {0.6666667057177732pt}{0pt}
  \pgfsyssoftpath@linetotoken {0.9166666178527835pt}{0.1443378101912376pt}
  \pgfsyssoftpath@linetotoken {1pt}{0pt}
}
\tl_gset:cn {g__tilings_side_polykite_b_tl}
{
  \pgfsyssoftpath@movetotoken {0pt}{0pt}
  \pgfsyssoftpath@linetotoken {0.08333pt}{-0.14433pt}
  \pgfsyssoftpath@linetotoken {0.33333pt}{0pt}
  \pgfsyssoftpath@linetotoken {0.58333pt}{-0.14433pt}
  \pgfsyssoftpath@linetotoken {0.66667pt}{0pt}
  \pgfsyssoftpath@linetotoken {1pt}{0pt}
}
\tl_gset:cn {g__tilings_side_polykite_F_tl}
{
  \pgfsyssoftpath@movetotoken {0pt}{-2.00000070292pt}
  \pgfsyssoftpath@linetotoken {0.74999982427pt}{-1.566987221617321pt}
  \pgfsyssoftpath@linetotoken {1pt}{-2.00000070292pt}
}
\tl_gset:cn {g__tilings_side_polykite_f_tl}
{
  \pgfsyssoftpath@movetotoken {0pt}{2pt}
  \pgfsyssoftpath@linetotoken {0.25pt}{1.56699pt}
  \pgfsyssoftpath@linetotoken {1pt}{2pt}
}

\clist_map_inline:nn {a,A,b,B,f,F}
{
  \tl_gclear_new:c {g__tilings_side_backup_#1_tl}
  \tl_gset_eq:cc {g__tilings_side_backup_#1_tl} {g__tilings_side_#1_tl}
  \tl_gclear_new:c {g__tilings_side_#1_tl}
  \tl_gset_eq:cc {g__tilings_side_#1_tl}{g__tilings_side_polykite_#1_tl}
}

\BakeTile{subcluster~ H}
\BakeTile{subcluster~ T}
\BakeTile{subcluster~ P}
\BakeTile{subcluster~ F}

\clist_map_inline:nn {a,A,b,B,f,F}
{
  \tl_gset_eq:cc {g__tilings_side_#1_tl} {g__tilings_side_backup_#1_tl}
}
```

2.7 Lindenmayer System

These are the rules for generating the super cluster tilings with the Lindenmayer System procedure.

_cluster_tile:nn

Useful auxiliary for placing a cluster tile from a particular set

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

```
  \int_gincr:N \g__tilings_tile_int
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)}
  \pgftransformscale{\fp_use:N \l__tilings_tmpa_fp}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \UseTile[
    every~ tile/.try,
    every~ #1~#2/.try,
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try/.expand~ once=\l__tilings_tmpc_tl
  ]{#1~ #2}
  \group_end:
}
\cs_generate_variant:Nn \__tilings_place_cluster_tile:nn {Vn}
```

(*End definition for* `\__tilings_place_cluster_tile:nn`.)

```
\prop_new:N \g__tilings_supercluster_lms_rule_prop
\prop_gput:Nnn \g__tilings_supercluster_lms_rule_prop {T}
{
  [s H]
}
\prop_gput:Nnn \g__tilings_supercluster_lms_rule_prop {H}
{
  [s {r{-60}} T]
  [s {x{\fp_to_decimal:n{1}}} {y{\fp_to_decimal:n{(1+2*\c__tilings_phi_fp)}}} H]
  [s
    {x{\fp_to_decimal:n{-2-3*\c__tilings_phi_fp}}}
    {y{\fp_to_decimal:n{-\c__tilings_phi_fp}}}
    H]
  [s
    {x{\fp_to_decimal:n{1+3*\c__tilings_phi_fp}}}
    {y{\fp_to_decimal:n{-1-\c__tilings_phi_fp}}}
    {r{-120}}
    H]
  [s
    {x{\fp_to_decimal:n{-1.5-3.5*\c__tilings_phi_fp}}}
    {y{\fp_to_decimal:n{2.5*\c__tilings_phi_fp+1.5}}}
    {r{-120}}
    P]
  [s
    {x{\fp_to_decimal:n{-1.5-2*\c__tilings_phi_fp}}}
    {y{\fp_to_decimal:n{-1.5-3*\c__tilings_phi_fp}}}
    {r{180}}
    P]
  [s
    {x{\fp_to_decimal:n{3+5.5*\c__tilings_phi_fp}}}
    {y{\fp_to_decimal:n{.5*\c__tilings_phi_fp}}}
    {r{120}}
    P]
  [s
    {x{\fp_to_decimal:n{-4.5-6.5*\c__tilings_phi_fp}}}
    {y{\fp_to_decimal:n{.5-.5*\c__tilings_phi_fp}}}
    {r{-120}}
    F]
  [s
    {x{\fp_to_decimal:n{1.5+4*\c__tilings_phi_fp}}}
    {y{\fp_to_decimal:n{-2.5-3*\c__tilings_phi_fp}}}
    F]
  [s
    {x{\fp_to_decimal:n{3+2.5*\c__tilings_phi_fp}}}
    {y{\fp_to_decimal:n{2+3.5*\c__tilings_phi_fp}}}
    {r{120}}
    F]
}
\prop_gput:Nnn \g__tilings_supercluster_lms_rule_prop {P}
{
  [s {r{60}} P]
  [s
    {x{\fp_to_decimal:n{2.5+3.5*\c__tilings_phi_fp}}}
    {y{\fp_to_decimal:n{-.5-.5*\c__tilings_phi_fp}}}
    {r{-120}}
    H]
  [s
    {x{\fp_to_decimal:n{-2.5-3.5*\c__tilings_phi_fp}}}
    {y{\fp_to_decimal:n{.5+.5*\c__tilings_phi_fp}}}
    {r{180}}
    H]
  [s
    {x{\fp_to_decimal:n{4.5+6*\c__tilings_phi_fp}}}
    {y{\fp_to_decimal:n{.5+\c__tilings_phi_fp}}}
    {r{120}}
    F]
  [s
    {x{\fp_to_decimal:n{-4.5-6*\c__tilings_phi_fp}}}
    {y{\fp_to_decimal:n{-.5-\c__tilings_phi_fp}}}
    {r{-60}}
    F]
}
\prop_gput:Nnn \g__tilings_supercluster_lms_rule_prop {F}
{
  [s {r{60}} P]
  [s
    {x{\fp_to_decimal:n{2.5+3.5*\c__tilings_phi_fp}}}
    {y{\fp_to_decimal:n{-.5-.5*\c__tilings_phi_fp}}}
    {r{-120}}
    H]
  [s
    {x{\fp_to_decimal:n{-2.5-3.5*\c__tilings_phi_fp}}}
    {y{\fp_to_decimal:n{.5+.5*\c__tilings_phi_fp}}}
    {r{180}}
    H]
  [s
    {x{\fp_to_decimal:n{4.5+6*\c__tilings_phi_fp}}}
    {y{\fp_to_decimal:n{.5+\c__tilings_phi_fp}}}
    {r{120}}
    F]
  [s
    {x{\fp_to_decimal:n{-4.5-6*\c__tilings_phi_fp}}}
    {y{\fp_to_decimal:n{-.5-\c__tilings_phi_fp}}}
    {r{-60}}
    F]
  [s
    {x{\fp_to_decimal:n{3+4.5*\c__tilings_phi_fp}}}
    {y{\fp_to_decimal:n{-2-2.5*\c__tilings_phi_fp}}}
    F]
}
\prop_gput:Nnn \g__tilings_drawables_lms_prop {supercluster} {HTPF}
\fp_const:Nn \c__tilings_phi_fp {(1 + sqrt(5))/2}
\prop_new:N \g__tilings_supercluster_lms_action_prop
```

```

\prop_gput:Nnn \g__tilings_supercluster_lms_action_prop {r}
{
  \pgftransformrotate{\l__tilings_parameters_lms_tl}
}
\prop_gput:Nnn \g__tilings_supercluster_lms_action_prop {x}
{
  \pgftransformxshift{
    \fp_to_dim:n
    {.5 * (\l__tilings_parameters_lms_tl) * \l__tilings_step_dim}
  }
}
\prop_gput:Nnn \g__tilings_supercluster_lms_action_prop {y}
{
  \pgftransformyshift{
    \fp_to_dim:n
    {.5 * sqrt(3) * (\l__tilings_parameters_lms_tl) * \l__tilings_step_dim}
  }
}

\prop_gput:Nnn \g__tilings_supercluster_lms_action_prop {s}
{
  \fp_set:Nn \l__tilings_tmpa_fp {
    \l__tilings_step_dim
    /
    \c__tilings_phi_fp
    /
    \c__tilings_phi_fp
  }
  \dim_set:Nn \l__tilings_step_dim {\fp_to_dim:N \l__tilings_tmpa_fp}
}
\prop_gput:Nnn \g__tilings_supercluster_lms_action_prop {H}
{
  \__tilings_place_cluster_tile:nn {super~ cluster}{H}
}
\prop_gput:Nnn \g__tilings_supercluster_lms_action_prop {T}
{
  \__tilings_place_cluster_tile:nn {super~ cluster}{T}
}
\prop_gput:Nnn \g__tilings_supercluster_lms_action_prop {P}
{
  \__tilings_place_cluster_tile:nn {super~ cluster}{P}
}
\prop_gput:Nnn \g__tilings_supercluster_lms_action_prop {F}
{
  \__tilings_place_cluster_tile:nn {super~ cluster}{F}
}
```

Parameters:

1. Cluster type (super cluster, meta cluster, subcluster)
2. This tile type (H, T, P, F)
3. This tile's name
4. Alignment tile's name
5. Edge to align along
6. Edge to align with

```
\cs_new_protected_nopar:Npn \__tilings_place_cluster_tile_as_pic:nnnnnn #1#2#3#4#5#6
{
  \group_begin:
  \int_gincr:N \g__tilings_tile_int
  \fp_set:Nn \l__tilings_tmpa_fp {\l__tilings_step_dim/(1cm)}
  \tl_set:Nx \l__tilings_tmpc_tl
  {
    {\int_use:N  \g__tilings_tile_int}
    {\int_use:N \g__tilings_tiles_int}
  }
  \tl_clear:N \l__tilings_tmpa_tl
  \tl_put_right:Nn \l__tilings_tmpa_tl
  {
    \pic[
      every~ tile/.try,
      every~ #1~#2/.try,
    }
    \tl_put_right:Nx \l__tilings_tmpa_tl
    {
    tile~ \int_use:N \g__tilings_tile_int/.try,
    tile/.try=\l__tilings_tmpc_tl,
    scale=\fp_use:N \l__tilings_tmpa_fp,
    }
    \tl_put_right:Nn \l__tilings_tmpa_tl
    {
      name=#3,
    }
    \tl_if_empty:nTF {#4}
    {
      \tl_put_right:Nn \l__tilings_tmpa_tl
      {
        first~ tile/.try,
      }
    }
    {
      \tl_put_right:Nn \l__tilings_tmpa_tl
      {
        align~ with=#4~along~#5
      }
      \tl_if_single:nF {#6}
      {
        \tl_put_right:Nx \l__tilings_tmpa_tl
        {
          \c_space_tl using~\tl_tail:n {#6}
        }
      }
      \tl_put_right:Nn \l__tilings_tmpa_tl {,}
    }
    \tl_put_right:Nn \l__tilings_tmpa_tl
    {
      #1~ #2
    ];
  }
  \tl_use:N \l__tilings_tmpa_tl
  \group_end:
}
\cs_generate_variant:Nn \__tilings_place_cluster_tile_as_pic:nnnnnn {
  Vnnnnn, VnVnnn, VnVVnn
}
\tikzset{
  cluster~ type/.initial=super~ cluster,
  first~ file/.style={transform~ shape}
}
\prop_new:N \g__tilings_cluster_lms_rule_prop
```

The first set of rules govern when the tile being replaced is a root tile, in which case one of the new tiles becomes the new root and all others are placed with respect to them.

It's convenient for code readability to have aliases for the labels for the parent and adjoining tiles, which are stored in the `\l_@@_parameters_lms_tl` token list.

```
\cs_new_nopar:Npn \__tilings_tile_label:
{
  \tl_item:Nn \l__tilings_parameters_lms_tl {1}
}
\cs_new_nopar:Npn \__tilings_adjoint_label:
{
  \tl_item:Nn \l__tilings_parameters_lms_tl {2}
}
```

A single T tile is replaced by a single H tile

```
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {T}
{
  [s {H{\__tilings_tile_label:0}{}}]
}
```

An H tile is replaced by 10 tiles, consisting of a T tile and 3 each of H, P, and F.

```
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {H}
{
  [s {r{-60}} {T{\__tilings_tile_label:0}{}}]
  [{ {HTa{A1}} {\__tilings_tile_label:1} {\__tilings_tile_label:0} }]
  [{ {HTa{A2}} {\__tilings_tile_label:2} {\__tilings_tile_label:0} }]
  [{ {HT{B1}b} {\__tilings_tile_label:3} {\__tilings_tile_label:0} }]
  [{ {PHb{B2}} {\__tilings_tile_label:4} {\__tilings_tile_label:1} }]
  [{ {PHb{B2}} {\__tilings_tile_label:5} {\__tilings_tile_label:2} }]
  [{ {PHAa} {\__tilings_tile_label:6} {\__tilings_tile_label:3} }]
  [{ {FHb{B1}} {\__tilings_tile_label:7} {\__tilings_tile_label:1} }]
  [{ {FHb{B1}} {\__tilings_tile_label:8} {\__tilings_tile_label:2} }]
  [{ {FHb{B2}} {\__tilings_tile_label:9} {\__tilings_tile_label:3} }]
}
```

Lastly, the P and F tile substitutions.

```
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {P}
{
  [s {r{60}} {P{\__tilings_tile_label:0}{}}]
  [{ {HPaA} {\__tilings_tile_label:1} {\__tilings_tile_label:0} }]
  [{ {HP{B2}b} {\__tilings_tile_label:2} {\__tilings_tile_label:0} }]
  [{ {FHb{B2}} {\__tilings_tile_label:3} {\__tilings_tile_label:1} }]
  [{ {FHb{B1}} {\__tilings_tile_label:4} {\__tilings_tile_label:2} }]
}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {F}
{
  [s {r{60}} {P{\__tilings_tile_label:0}{}}]
  [{ {HPaA} {\__tilings_tile_label:1} {\__tilings_tile_label:0} }]
  [{ {HP{B2}b} {\__tilings_tile_label:2} {\__tilings_tile_label:0} }]
  [{ {FHb{B2}} {\__tilings_tile_label:3} {\__tilings_tile_label:1} }]
  [{ {FHb{B1}} {\__tilings_tile_label:4} {\__tilings_tile_label:2} }]
  [{ {FHb{B1}} {\__tilings_tile_label:5} {\__tilings_tile_label:1} }]
}
```

The rest of the rules are for when the tile being replaced was itself positioned by aligning it with another tile. For these tiles, one of its edge tiles will be its root and positioned alongside one of the edge tiles of the replacement of the original tile's alignment tile. Then all the other tiles are positioned out from that root. The labelling has to be the same regardless of the order of drawing the tiles.

Not every edge pairing is necessary to generate a pattern as the edges that can be created are all between $A^{\pm}$ edges and between $B^{\pm}$ edges. However, to avoid errors in case they are part of the seed then for now we create blank substitution rules that will effectively remove any such rogue elements.

```
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {TH{A1}a} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {TH{A2}a} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {THb{B1}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {THb{B2}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {HTa{A1}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {HTa{A2}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {HT{B1}b} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {HT{B2}b} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {HPaA} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {HP{B1}b} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {HP{B2}b} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {HF{B1}b} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {HF{B2}b} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {PHAa} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {PHb{B1}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {PHb{B2}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {PF{11}{11}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {PF{12}{11}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {PF{11}{12}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {PF{12}{12}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {FHb{B1}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {FHb{B2}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {FP{11}{11}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {FP{12}{11}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {FP{11}{12}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {FP{12}{12}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {FFfF} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {FFFf} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {FF{11}{11}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {FF{12}{11}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {FF{11}{12}} {}
\prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {FF{12}{12}} {}
```

To help create the rules then we start with some helper macros. Each of these creates the substitution rule for a tile given certain information about where the parent tile is positioned. Most of the substitution information consists of placing the tiles next to each other, so only the first tile needs to know about a tile from a different set. This makes it relatively easy to set up some templates for the substitution rules.

```
\cs_new_nopar:cpn {__tilings_T{A1}_creator:nnnnn} #1#2#3#4#5
{
  \prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {T#1{A1}#2}
  {
    [{ {H#3{B1}#4} {\__tilings_tile_label:0} {\__tilings_adjoint_label:#5} }]
  }
}
\cs_new_nopar:cpn {__tilings_T{A2}_creator:nnnnn} #1#2#3#4#5
{
  \prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {T#1{A2}#2}
  {
    [{ {H#3{B2}#4} {\__tilings_tile_label:0} {\__tilings_adjoint_label:#5} }]
  }
}
\cs_new_nopar:cpn {__tilings_Tb_creator:nnnnn} #1#2#3#4#5
{
  \prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {T#1b#2}
  {
    [{ {H#3a#4} {\__tilings_tile_label:0} {\__tilings_adjoint_label:#5} }]
  }
}
```

```

\cs_new_nopar:cpn {__tilings_H{B1}_creator:nnnnn} #1#2#3#4#5
{
  \prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {H#1{B1}#2}
  {
    [{ {P#3A#4} {\__tilings_tile_label:4} {\__tilings_adjoint_label:#5} }]

    [{ {HP{B2}b} {\__tilings_tile_label:1} {\__tilings_tile_label:4} }]
    [{ {TH{A1}a} {\__tilings_tile_label:0} {\__tilings_tile_label:1} }]

    [{ {HTa{A2}} {\__tilings_tile_label:2} {\__tilings_tile_label:0} }]
    [{ {PHb{B2}} {\__tilings_tile_label:5} {\__tilings_tile_label:2} }]

    [{ {HT{B1}b} {\__tilings_tile_label:3} {\__tilings_tile_label:0} }]
    [{ {PHAa} {\__tilings_tile_label:6} {\__tilings_tile_label:3} }]

    [{ {FHb{B1}} {\__tilings_tile_label:7} {\__tilings_tile_label:1} }]
    [{ {FHb{B1}} {\__tilings_tile_label:8} {\__tilings_tile_label:2} }]
    [{ {FHb{B2}} {\__tilings_tile_label:9} {\__tilings_tile_label:3} }]
  }
}

\cs_new_nopar:cpn {__tilings_H{B2}_creator:nnnnn} #1#2#3#4#5
{
  \prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {H#1{B2}#2}
  {
    [{ {P#3A#4} {\__tilings_tile_label:5} {\__tilings_adjoint_label:#5} }]

    [{ {HP{B2}b} {\__tilings_tile_label:2} {\__tilings_tile_label:5} }]
    [{ {TH{A2}a} {\__tilings_tile_label:0} {\__tilings_tile_label:2} }]

    [{ {HTa{A1}} {\__tilings_tile_label:1} {\__tilings_tile_label:0} }]
    [{ {PHb{B2}} {\__tilings_tile_label:4} {\__tilings_tile_label:1} }]

    [{ {HT{B1}b} {\__tilings_tile_label:3} {\__tilings_tile_label:0} }]
    [{ {PHAa} {\__tilings_tile_label:6} {\__tilings_tile_label:3} }]

    [{ {FHb{B1}} {\__tilings_tile_label:7} {\__tilings_tile_label:1} }]
    [{ {FHb{B1}} {\__tilings_tile_label:8} {\__tilings_tile_label:2} }]
    [{ {FHb{B2}} {\__tilings_tile_label:9} {\__tilings_tile_label:3} }]
  }
}

\cs_new_nopar:cpn {__tilings_Ha_creator:nnnnn} #1#2#3#4#5
{
  \prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {H#1a#2}
  {
    [{ {P#3b#4} {\__tilings_tile_label:6} {\__tilings_adjoint_label:#5} }]

    [{ {HPaA} {\__tilings_tile_label:3} {\__tilings_tile_label:6} }]
    [{ {THb{B1}} {\__tilings_tile_label:0} {\__tilings_tile_label:3} }]

    [{ {HTa{A1}} {\__tilings_tile_label:1} {\__tilings_tile_label:0} }]
    [{ {PHb{B2}} {\__tilings_tile_label:4} {\__tilings_tile_label:1} }]

    [{ {HTa{A2}} {\__tilings_tile_label:2} {\__tilings_tile_label:0} }]
    [{ {PHb{B2}} {\__tilings_tile_label:5} {\__tilings_tile_label:2} }]

    [{ {FHb{B1}} {\__tilings_tile_label:7} {\__tilings_tile_label:1} }]
    [{ {FHb{B1}} {\__tilings_tile_label:8} {\__tilings_tile_label:2} }]
    [{ {FHb{B2}} {\__tilings_tile_label:9} {\__tilings_tile_label:3} }]
  }
}

\cs_new_nopar:cpn {__tilings_PA_creator:nnnnn} #1#2#3#4#5
{
  \prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {P#1A#2}
  {
    [{ {H#3{B1}#4} {\__tilings_tile_label:1} {\__tilings_adjoint_label:#5} }]
    [{ {PHAa} {\__tilings_tile_label:0} {\__tilings_tile_label:1} }]
    [{ {HP{B2}b} {\__tilings_tile_label:2} {\__tilings_tile_label:0} }]

    [{ {FHb{B2}} {\__tilings_tile_label:3} {\__tilings_tile_label:1} }]
    [{ {FHb{B1}} {\__tilings_tile_label:4} {\__tilings_tile_label:2} }]
  }
}

\cs_new_nopar:cpn {__tilings_Pb_creator:nnnnn} #1#2#3#4#5
{
  \prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {P#1b#2}
  {
    [{ {H#3a#4} {\__tilings_tile_label:2} {\__tilings_adjoint_label:#5} }]
    [{ {PHb{B2}} {\__tilings_tile_label:0}{\__tilings_tile_label:2} }]
    [{ {HPaA} {\__tilings_tile_label:1} {\__tilings_tile_label:0} }]

    [{ {FHb{B2}} {\__tilings_tile_label:3} {\__tilings_tile_label:1} }]
    [{ {FHb{B1}} {\__tilings_tile_label:4} {\__tilings_tile_label:2} }]
  }
}

\cs_new_nopar:cpn {__tilings_Fb_creator:nnnnn} #1#2#3#4#5
{
  \prop_gput:Nnn \g__tilings_cluster_lms_rule_prop {F#1b#2}
  {
    [{ {H#3a#4} {\__tilings_tile_label:2} {\__tilings_adjoint_label:#5} }]
    [{ {PHb{B2}} {\__tilings_tile_label:0}{\__tilings_tile_label:2} }]
    [{ {HPaA} {\__tilings_tile_label:1} {\__tilings_tile_label:0} }]

    [{ {FHb{B2}} {\__tilings_tile_label:3} {\__tilings_tile_label:1} }]
    [{ {FHb{B1}} {\__tilings_tile_label:4} {\__tilings_tile_label:2} }]
    [{ {FHb{B1}} {\__tilings_tile_label:5} {\__tilings_tile_label:1} }]
  }
}
```

Now that the creators are set up it is time to invoke them.

```
\clist_map_inline:nn
{
  TH{{A1}}a HP{{{B1}}}b06,
  TH{{A2}}a HP{{{B2}}}b06,
  THb{{B1}} HPaA04,
  THb{{B2}} HPaA05,
  PHAa HP{{{B1}}}b16,
  PHb{{B1}} HPaA24,
  PHb{{B2}} HPaA25,
  FHb{{B1}} HPaA24,
  FHb{{B2}} HPaA25,
}
{
  \tl_clear:N \l__tilings_tmpa_tl
  \tl_put_right:Nn \l__tilings_tmpa_tl { \use:c }
  \tl_put_right:Nx \l__tilings_tmpa_tl
  {
    {__tilings_ \tl_item:nn {#1}{2}\tl_item:nn {#1}{4} _creator:nnnnn}
    {\tl_item:nn {#1}{1}}{\tl_item:nn{#1}{3}}
    \tl_item:nn {#1}{5}\tl_item:nn{#1}{7}
    \tl_item:nn {#1}{9}
  }
  \tl_use:N \l__tilings_tmpa_tl
  \tl_clear:N \l__tilings_tmpa_tl
  \tl_put_right:Nn \l__tilings_tmpa_tl { \use:c }
  \tl_put_right:Nx \l__tilings_tmpa_tl
  {
    {__tilings_ \tl_item:nn {#1}{1}\tl_item:nn {#1}{3} _creator:nnnnn}
    {\tl_item:nn {#1}{2}}{\tl_item:nn{#1}{4}}
    \tl_item:nn {#1}{6}\tl_item:nn{#1}{8}
    \tl_item:nn {#1}{10}
  }
  \tl_use:N \l__tilings_tmpa_tl
}

\prop_new:N \g__tilings_cluster_lms_action_prop
\prop_gput:Nnn \g__tilings_drawables_lms_prop {cluster} {HTPF}
\prop_gput:Nnn \g__tilings_cluster_lms_action_prop {r}
{
  \pgftransformrotate{\l__tilings_parameters_lms_tl}
}
\prop_gput:Nnn \g__tilings_cluster_lms_action_prop {x}
{
  \pgftransformxshift{
    \fp_to_dim:n
    {.5 * (\l__tilings_parameters_lms_tl) * \l__tilings_step_dim}
  }
}
\prop_gput:Nnn \g__tilings_cluster_lms_action_prop {y}
{
  \pgftransformyshift{
    \fp_to_dim:n
    {.5 * sqrt(3) * (\l__tilings_parameters_lms_tl) * \l__tilings_step_dim}
  }
}

\prop_gput:Nnn \g__tilings_cluster_lms_action_prop {s}
{
  \fp_set:Nn \l__tilings_tmpa_fp {
    \l__tilings_step_dim
    /
    \c__tilings_phi_fp
    /
    \c__tilings_phi_fp
  }
  \dim_set:Nn \l__tilings_step_dim {\fp_to_dim:N \l__tilings_tmpa_fp}
}
```

The first set of actions are for when this tile is the root so doesn't have a parent

```
\clist_map_inline:nn {H,T,P,F}
{

  \prop_gput:Nnn \g__tilings_cluster_lms_action_prop {#1}
  {
    \__tilings_tikz_keys_get:Nn \l__tilings_tmpa_tl {cluster~type}
    \tl_set:Nx \l__tilings_tmpb_tl {\tl_item:Nn \l__tilings_parameters_lms_tl {1}}
    \__tilings_place_cluster_tile_as_pic:VnVnnn
    \l__tilings_tmpa_tl {#1} \l__tilings_tmpb_tl {}{}{}
  }
}
```

The second set is for when there is an adjoining edge

```
\cs_new_protected_nopar:Npn \__tilings_place_cluster_tile_as_pic_aux:nnnn #1#2#3#4
{
  \__tilings_tikz_keys_get:Nn \l__tilings_tmpa_tl {cluster~type}
  \tl_set:Nx \l__tilings_tmpb_tl {\__tilings_tile_label:}
  \tl_set:Nx \l__tilings_tmpc_tl {\__tilings_adjoint_label:}
  \__tilings_place_cluster_tile_as_pic:VnVVnn
  \l__tilings_tmpa_tl {#1} \l__tilings_tmpb_tl \l__tilings_tmpc_tl {#4}{#3}
}

\clist_map_inline:nn {
  TH{A1}a,
  TH{A2}a,
  THb{B1},
  THb{B2},
  HTa{A1},
  HTa{A2},
  HT{B1}b,
  HT{B2}b,
  HPaA,
  HP{B1}b,
  HP{B2}b,
  HF{B1}b,
  HF{B2}b,
  PHAa,
  PHb{B1},
  PHb{B2},
  PF{11}{11},
  PF{12}{11},
  PF{11}{12},
  PF{12}{12},
  FHb{B1},
  FHb{B2},
  FP{11}{11},
  FP{12}{11},
  FP{11}{12},
  FP{12}{12},
  FFfF,
  FFFf,
  FF{11}{11},
  FF{12}{11},
  FF{11}{12},
  FF{12}{12}
}
{
  \prop_gput:Nnn \g__tilings_cluster_lms_action_prop {#1}
  {
    \__tilings_place_cluster_tile_as_pic_aux:nnnn #1
  }
}

\ExplSyntaxOff
```

```
\ProvidesFile {tikzlibrarypenrose.code.tex}
 [2023/06/01 v2.0 TikZ pics for Penrose tiles]
\usetikzlibrary{tilings.penrose}
```

Backwards compatibility mode:

- `\SetPenrosePath` is `\SetTilingPath`

- `\BakePenroseTile` and `\MakePenroseTile` are `\BakeTile`
- `\UsePenroseTile` is `\UseTile`
- `\PenroseDecomposition` is `\TilingDecomposition`

```
\ExplSyntaxOn
\NewDocumentCommand \SetPenrosePath { m }
{
  \__tilings_set_tiling_path:n {#1}
}
\NewDocumentCommand \BakePenroseTile {m}
{
  \__tilings_bake_tile:n {#1}
}
\NewDocumentCommand \MakePenroseTile {m}
{
  \__tilings_bake_tile:n {#1}
}
\NewDocumentCommand \UsePenroseTile {O{} m}
{
  \__tilings_use_tile:nn {#1}{#2}
}
\NewDocumentCommand \PenroseDecomposition { O{} m m m }
{
 \__tilings_tiling_decomposition:nnnn {#1}{#2}{#3}{#4}
}
\ExplSyntaxOff
```

```
\tikzset{
  save Penrose path/.forward to=/tikz/save tiling path,
  clone Penrose side path/.forward to=/tikz/clone tiling side path,
  spath/prefix/Penrose side/.forward to=/tikz/spath/prefix/tiling side,
  spath/suffix/Penrose side/.forward to=/tikz/spath/suffix/tiling side,
  clone Penrose tile path/.forward to=/tikz/clone tiling tile path,
  spath/prefix/Penrose tile/.forward to=/tikz/spath/prefix/tiling tile,
  spath/suffix/Penrose tile/.forward to=/tikz/spath/suffix/tiling tile,
  Penrose step/.forward to=/tikz/tiling step,
  every tile/.append style={
    every Penrose tile/.try
  },
  every tile clip/.append style={
    every Penrose tile clip/.try
  },
  every tile pic/.append style={
    every Penrose pic/.try
  },
  tile/.append style={
    Penrose tile #1/.try,
    Penrose tile/.try=#1
  }
}
```