The `polyomino` package

Polyominoes using Ti*k*Z and LaTeX3

Matthias Floré

Version 1.0 (2024/08/01)

Abstract

This package is based on the package `tikz` (see [1]) and can be used to draw polyominoes. It is possible to define custom styles, pics and grids.

Contents

1 Usage

The package `polyomino` can be used by putting the following in the preamble.

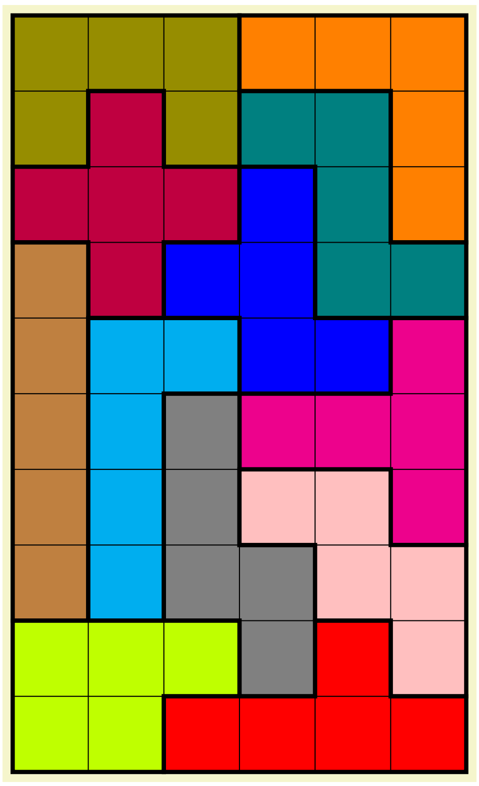
```
\usepackage{polyomino}
```

The package `polyomino` loads the package `tikz`.

2 The command \polyomino

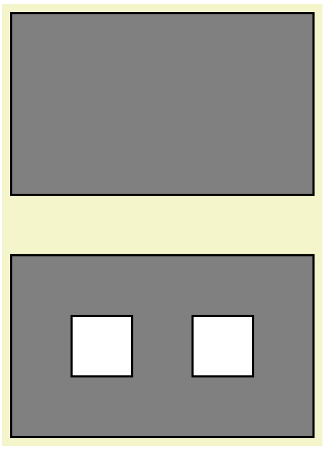
`\polyomino[⟨options⟩]{⟨polyomino specification⟩}`

This command can be placed inside a `tikzpicture` environment. The ⟨*polyomino specification*⟩ is a token list. Spaces in this list are ignored. With the initial settings, a `,` starts a new row. Otherwise each element in this list corresponds to a cell. An element can consist of multiple characters by surrounding it with braces. The ⟨*options*⟩ can be given with the keys described in Section 3.



```
\pgfkeys{
  /polyomino,
  p={F}{style={blue,draw=black,ultra thick}},
  p={I}{style={brown,draw=black,ultra thick}},
  p={L}{style={cyan,draw=black,ultra thick}},
  p={N}{style={gray,draw=black,ultra thick}},
  p={P}{style={lime,draw=black,ultra thick}},
  p={T}{style={magenta,draw=black,ultra thick}},
  p={U}{style={olive,draw=black,ultra thick}},
  p={V}{style={orange,draw=black,ultra thick}},
  p={W}{style={pink,draw=black,ultra thick}},
  p={X}{style={purple,draw=black,ultra thick}},
  p={Y}{style={red,draw=black,ultra thick}},
  p={Z}{style={teal,draw=black,ultra thick}}
}
\begin{tikzpicture}
\polyomino[
  grid
]{
  UUUVVV,
  UXUZZV,
  XXXFZV,
  IXFFZZ,
  ILLFFT,
  ILNTTT,
  ILNWWT,
  ILNNWW,
  PPPNYW,
  PPYYYY
}
\end{tikzpicture}
```

The algorithm constructs the border of each polyomino. It does not consider holes determined by empty cells. Although it does detect a cell inside a polyomino which has a different style. This is illustrated in the example below.



```
\begin{tikzpicture}[scale=0.8]
\polyomino[
  p={a}{style={gray,draw=black,thick}}
]{
  aaaaa,
  a.a.a,
  aaaaa
}
\polyomino[
  at={(0,-4)},
  p={a}{style={gray,draw=black,thick}},
  p={*}{style={white,draw=black,thick}}
]{
  aaaaa,
  a*a*a,
  aaaaa
}
\end{tikzpicture}
```

3 Keys

The keys in this Section can be given as ⟨*options*⟩ to the command `\polyomino`.

There are two key families: `/polyomino` and `/polyomino/p_2`. The key family `/polyomino` is intended for usage in documents whereas `/polyomino/p_2` is not. In the key family `/polyomino`, also keys from the key family `/polyomino/p_2` will be looked up. The second argument from the key `p` only accepts keys from the key family `/polyomino/p_2`.

`/polyomino/at={⟨point⟩}` (no default, initially `(0,0)`)

This key defines the bottom left coordinate of the polyomino.

`/polyomino/p_2/connected` (no value)

This key sets the `pic` type (which is activated by the key `pic`) to false. This is the initial setting.

`/polyomino/p_2/discrete` (no value)

This key sets the `pic` type (which is activated by the key `pic`) to true.

`/polyomino/empty cell={`⟨*token list*⟩`}` (no default, initially `.`)

A cell corresponding to the ⟨*token list*⟩ in the ⟨*polyomino specification*⟩ will be left empty.

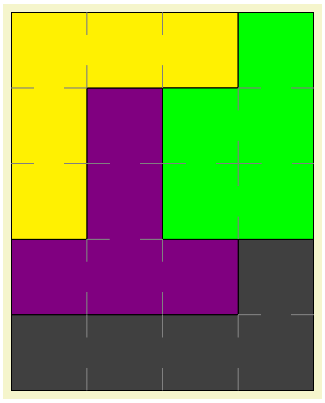
A cell corresponding to the empty token list will always be left empty.

`/polyomino/grid=`⟨*boolean*⟩ (default `true`, initially `false`)

If true then a grid is drawn. The grid does not apply to borders of polyominoes. The style of this grid is determined by the key `grid style`. A grid does not apply to a cell with a `pic`.

`/polyomino/grid style={`⟨*options*⟩`}` (style, no default, initially empty)

This key determines the style of the grid.



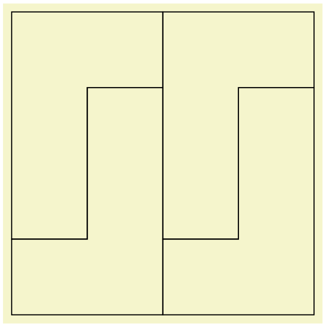
```
\begin{tikzpicture}[rotate=90]
\polyomino[
  grid,
  grid style={gray,dash pattern=on 3mm off 4mm on 3mm off 0mm},
  p={L}{style={darkgray,draw=black}},
  p={P}{style={green,draw=black}},
  p={T}{style={violet,draw=black}},
  p={V}{style={yellow,draw=black}}
]{
  LTVVV,
  LTTTV,
  LTPPV,
  LLPPP
}
\end{tikzpicture}
```

`/polyomino/p_2/p={`⟨*name*⟩`}{`⟨*options*⟩`}` (style, no default, initially empty)

This key determines the style of the polyomino with ⟨*name*⟩ in the ⟨*polyomino specification*⟩.

The ⟨*options*⟩ only accept keys from the key family `/polyomino/p_2`.

In the example below, the polyominoes have the same shape but are differentiated by using different names.

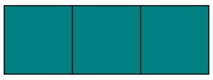


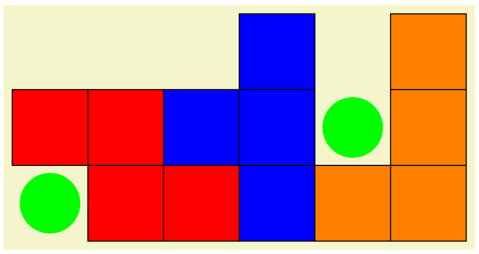
```
\begin{tikzpicture}
\pgfkeys{
  /polyomino,
  p={a}{},
  p={b}{},
  style={fill=none,draw}
}
\def\example{
  aa,
  ab,
  ab,
  bb
}
\polyomino{\example}
\polyomino[
  at={(2,0)}
]{\example}
\end{tikzpicture}
```

`/polyomino/p_2/pic={`⟨*code*⟩`}` (no default)

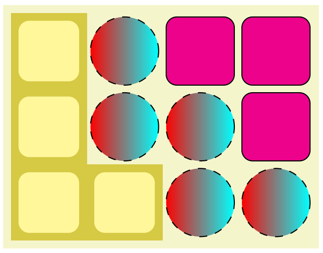
The ⟨*code*⟩ defines the `pic` which is used for each cell of the polyomino.

A grid does not apply to a cell with a `pic`.





```
\begin{tikzpicture}
\polyomino[
  empty cell=*,
  grid,
  p={a}{style={red,draw=black}},
  p={b}{style={blue,draw=black}},
  p={c}{style={orange,draw=black}},
  p={circle}{pic={\fill[green] (0,0) circle[radius=0.4];}},
  row sep=;
]{
     {}     * * b    {}    c ;
     a      a b b {circle} c ;
  {circle} a a b     c     c
}
\end{tikzpicture}
```



```
\begin{tikzpicture}
\polyomino[
  p={circle}{
    pic={\path[pic actions] (0,0) circle[radius=0.45];},
    style={right color=cyan,left color=red,draw,dashed}
  },
  p={L}{
    pic={
      \fill[yellow!80!black] (-0.5,-0.5) rectangle +(1,1);
      \fill[yellow!50,rounded corners] (-0.4,-0.4) rectangle +(0.8,0.8);
    }
  },
  p={square}{
    pic={\path[pic actions] (-0.45,-0.45) rectangle +(0.9,0.9);},
    style={fill=magenta,draw,rounded corners}
  }
]{
  L {circle} {square} {square} ,
  L {circle} {circle} {square} ,
  L    L     {circle} {circle}
}
\end{tikzpicture}
```

`/polyomino/row sep={⟨token list⟩}` (no default, initially `,`)

The ⟨*token list*⟩ in the ⟨*polyomino specification*⟩ will start a new row.

`/polyomino/p_2/style={⟨options⟩}` (style, no default, initially empty)

This key determines the style of the polyomino.

References

[1] Till Tantau, *The TikZ and* PGF *Packages*, Manual for version 3.1.10, https://ctan.org/pkg/pgf, 2023.

Index

A The source code

```
%% polyomino.sty
%% Copyright 2024 Matthias Floré
%
% This work may be distributed and/or modified under the
% conditions of the LaTeX Project Public License, either version 1.3c
% of this license or (at your option) any later version.
% The latest version of this license is in
%   http://www.latex-project.org/lppl.txt
% and version 1.3c or later is part of all distributions of LaTeX
% version 2005/12/01 or later.
%
% This work has the LPPL maintenance status `maintained'.
%
% The Current Maintainer of this work is Matthias Floré.
%
% This work consists of the files polyomino.pdf, polyomino.sty,
% polyomino.tex and README.md.
\NeedsTeXFormat{LaTeX2e}
\RequirePackage{tikz}
\ProvidesExplPackage{polyomino}{2024/08/01}{1.0}{Polyominoes using TikZ and LaTeX3}
```

A.1 Variables and variants

```
\bool_new:N \l__polyomino_grid_bool
\bool_new:N \l__polyomino_pic_bool

\int_new:N \l__polyomino_col_int
\int_new:N \l__polyomino_dir_int
\int_new:N \l__polyomino_max_int
\int_new:N \l__polyomino_row_int
\int_new:N \l__polyomino_x_int
\int_new:N \l__polyomino_y_int

\seq_new:N \l__polyomino_add_seq
\seq_new:N \l__polyomino_cols_seq
```

```
\tl_new:N \l__polyomino_path_tl
\tl_new:N \l__polyomino_pic_tl

\cs_generate_variant:Nn \tl_map_inline:nn { en }
```

A.2 Pgfkeys

```
\pgfkeys
  {
    / polyomino /. is~family ,
    / polyomino /. search~also = / polyomino / p_2 ,
    / polyomino ,
    at /. initial = { ( 0 , 0 ) } ,
    empty~cell /. initial = . ,
    grid /. code = \bool_set:Nn \l__polyomino_grid_bool { \cs:w c_#1_bool \cs_end: } ,
    grid /. default = true ,
    grid = false ,
    grid~style /. style = { grid_style /. style = {#1} } ,
    grid_style /. style = {} ,
    row~sep /. initial = { , } ,
  }

\pgfkeys
  {%a separate key family so that the second argument of the key p only accepts keys which apply to a separate polyomino
    / polyomino / p_2 /. is~family ,
    / polyomino / p_2 ,
    connected /. code = \bool_set_false:N \l__polyomino_pic_bool ,
    connected /. value~forbidden ,
    discrete /. code = \bool_set_true:N \l__polyomino_pic_bool ,
    discrete /. value~forbidden ,
    p /. style~2~args = { #1__style /. style = {#2} } ,%2 underscores to avoid the same name as for example the key style_style
    pic /. code =
      {
        \bool_set_true:N \l__polyomino_pic_bool
        \tl_set:Nn \l__polyomino_pic_tl {#1}
      } ,
    style /. style = { style_style /. style = {#1} } ,
    style_style /. style = {} ,
  }
```

A.3 The command \polyomino

```
\NewDocumentCommand \polyomino { O {} m }
  {
    {%note the double braces {{...}} so that the contents is in a group and in particular, \pgfkeys is applied locally
      \pgfkeys { / polyomino , #1 }
      \int_zero:N \l__polyomino_col_int
      \int_set:Nn \l__polyomino_row_int { 1 }
      \seq_clear:N \l__polyomino_cols_seq
      \tl_map_inline:en {#2}
      %it is convenient that this ignores spaces in #2
      %e argument specifier for the case that #2 is given by a command or contains a command
        {
          \tl_if_eq:neTF {##1} { \pgfkeysvalueof { / polyomino / row~sep } }
            {
              \seq_put_right:NV \l__polyomino_cols_seq \l__polyomino_col_int
              \int_incr:N \l__polyomino_row_int
              \int_zero:N \l__polyomino_col_int
            }
            {
              \int_incr:N \l__polyomino_col_int
              \tl_clear_new:c { l__polyomino_\int_use:N \l__polyomino_row_int _\int_use:N \l__polyomino_col_int _tl }
              \tl_if_eq:neF {##1} { \pgfkeysvalueof { / polyomino / empty~cell } }
                { \tl_set:cn { l__polyomino_\int_use:N \l__polyomino_row_int _\int_use:N \l__polyomino_col_int _tl } {##1} }
              \tl_gclear_new:c { g__polyomino_\int_use:N \l__polyomino_row_int _\int_use:N \l__polyomino_col_int _tl }
            }
        }
      \seq_put_right:NV \l__polyomino_cols_seq \l__polyomino_col_int
      \int_set:Nn \l__polyomino_max_int { \fp_eval:n { max ( \seq_use:Nn \l__polyomino_cols_seq { , } ) } }
      \seq_map_indexed_inline:Nn \l__polyomino_cols_seq
        {
          \tl_clear_new:c { l__polyomino_##1_0_tl }
          \int_step_inline:nnn { ##2 + 1 } { \l__polyomino_max_int + 1 }
            { \tl_clear_new:c { l__polyomino_##1_####1_tl } }
        }
```

```
\int_step_inline:nnn { 0 } { \l__polyomino_max_int + 1 }
  {
    \tl_clear_new:c { l__polyomino_0_##1_tl }
    \tl_clear_new:c { l__polyomino_\int_eval:n { \l__polyomino_row_int + 1 }_##1_tl }
  }
\pgfkeys
  {
    / tikz ,
    shift /. expanded = { \pgfkeysvalueof { / polyomino / at } } ,
    shift = { ( 0 , \seq_count:N \l__polyomino_cols_seq ) }
  }
\seq_map_indexed_inline:Nn \l__polyomino_cols_seq
  {
    \int_step_inline:nn {##2}
      {
        \tl_if_empty:cF { l__polyomino_##1_####1_tl }
          {
            {%note the double braces {{...}} so that \pgfkeys is applied locally
              \pgfkeys { / polyomino / p_2 , \cs:w l__polyomino_##1_####1_tl \cs_end: __style }
              \bool_if:NTF \l__polyomino_pic_bool
                { \pic [ / polyomino / p_2 / style_style ] at ( ####1 - 0.5 , 0.5 - ##1 ) { code = { \l__polyomino_pic_tl } } ; }
                {
                  \seq_clear:N \l__polyomino_add_seq
                  \tl_if_eq:ccF { l__polyomino_##1_####1_tl } { l__polyomino_##1_\int_eval:n { ####1 - 1 }_tl }
                    {
                      \tl_if_empty:cT { g__polyomino_##1_####1_tl }
                        {
                          \int_set:Nn \l__polyomino_dir_int { 1 }
                          \int_set:Nn \l__polyomino_col_int {####1}
                          \int_set:Nn \l__polyomino_row_int {##1}
                          \int_set:Nn \l__polyomino_x_int {####1}
                          \int_set:Nn \l__polyomino_y_int { 1 - ##1 }
                          \tl_build_begin:N \l__polyomino_path_tl
                          \fp_do_until:nn { ####1 - 1 = \l__polyomino_x_int && 1 - ##1 = \l__polyomino_y_int }
                            {
                              %concerning \tl_build_put_right:Ne \l__polyomino_path_tl,
                              %for example (0,0)--(0,1)--(0,2) results in a larger file size than (0,0)--(0,2)
                              \tl_if_eq:ccTF
                                { l__polyomino_##1_####1_tl }
```

```
{
  l__polyomino
  _\int_eval:n
    { \l__polyomino_row_int + \clist_item:nn { 0 , 1 , 0 , -1 } { \l__polyomino_dir_int } }
  _\int_eval:n
    { \l__polyomino_col_int + \clist_item:nn { 1 , 0 , -1 , 0 } { \l__polyomino_dir_int } }
  _tl
}
{
  \tl_if_eq:ccTF
    { l__polyomino_##1_####1_tl }
    {
      l__polyomino
      _\int_eval:n
        { \l__polyomino_row_int + \clist_item:nn { -1 , 1 , 1 , -1 } { \l__polyomino_dir_int } }
      _\int_eval:n
        { \l__polyomino_col_int + \clist_item:nn { 1 , 1 , -1 , -1 } { \l__polyomino_dir_int } }
      _tl
    }
    {
      \tl_build_put_right:Ne \l__polyomino_path_tl
        { -- ( \int_use:N \l__polyomino_x_int , \int_use:N \l__polyomino_y_int ) }
      \int_add:Nn \l__polyomino_row_int
        { \clist_item:nn { -1 , 1 , 1 , -1 } { \l__polyomino_dir_int } }
      \int_add:Nn \l__polyomino_col_int
        { \clist_item:nn { 1 , 1 , -1 , -1 } { \l__polyomino_dir_int } }
      \int_compare:nNnTF { \l__polyomino_dir_int } = { 1 }
        { \int_set:Nn \l__polyomino_dir_int { 4 } }
        { \int_decr:N \l__polyomino_dir_int }
    }
    {
      \int_add:Nn \l__polyomino_row_int
        { \clist_item:nn { 0 , 1 , 0 , -1 } { \l__polyomino_dir_int } }
      \int_add:Nn \l__polyomino_col_int
        { \clist_item:nn { 1 , 0 , -1 , 0 } { \l__polyomino_dir_int } }
    }
  \tl_if_empty:cTF
    { g__polyomino_\int_use:N \l__polyomino_row_int _\int_use:N \l__polyomino_col_int _tl }
    {
```

```
                                      \seq_put_right:Ne \l__polyomino_add_seq
                                        { \int_use:N \l__polyomino_row_int _\int_use:N \l__polyomino_col_int }
                                    }
                                    {
                                      \bool_set_true:N \l__polyomino_pic_bool
                                      \int_set:Nn \l__polyomino_x_int { ####1 - 1 }
                                      \int_set:Nn \l__polyomino_y_int { 1 - ##1 }
                                    }
                                }
                                {
                                  \tl_build_put_right:Ne \l__polyomino_path_tl
                                    { -- ( \int_use:N \l__polyomino_x_int , \int_use:N \l__polyomino_y_int ) }
                                  \int_compare:nNnTF { \l__polyomino_dir_int } = { 4 }
                                    { \int_set:Nn \l__polyomino_dir_int { 1 } }
                                    { \int_incr:N \l__polyomino_dir_int }
                                }
                              \bool_if:NF \l__polyomino_pic_bool
                                {
                                  \int_add:Nn \l__polyomino_x_int { \clist_item:nn { 1 , 0 , -1 , 0 } { \l__polyomino_dir_int } }
                                  \int_add:Nn \l__polyomino_y_int { \clist_item:nn { 0 , -1 , 0 , 1 } { \l__polyomino_dir_int } }
                                }
                            }
                          \tl_build_end:N \l__polyomino_path_tl
                          \bool_if:NF \l__polyomino_pic_bool
                            { \fill [ / polyomino / p_2 / style_style ] ( ####1 - 1 , 1 - ##1 ) \l__polyomino_path_tl -- cycle ; }
                        }
                    }
                  \tl_gset:cn { g__polyomino_##1_####1_tl } { c }
                  \seq_map_inline:Nn \l__polyomino_add_seq
                    { \tl_gset:cn { g__polyomino_########1_tl } { c } }
                }
            }
          }
      }
  }
\bool_if:NT \l__polyomino_grid_bool
  {
    \int_step_inline:nn { \seq_count:N \l__polyomino_cols_seq - 1 }
      {
        \int_zero:N \l__polyomino_col_int
        \int_zero:N \l__polyomino_x_int
        \int_set:Nn \l__polyomino_y_int
          { \int_min:nn { \seq_item:Nn \l__polyomino_cols_seq {##1} } { \seq_item:Nn \l__polyomino_cols_seq { ##1 + 1 } } }
        \int_while_do:nNnn { \l__polyomino_x_int } < { \l__polyomino_y_int }
          {
            \bool_do_while:nn
              {
                \tl_if_eq_p:cc
                  { l__polyomino_##1_\int_use:N \l__polyomino_x_int _tl }
                  { l__polyomino_\int_eval:n { ##1 + 1 }_\int_use:N \l__polyomino_x_int _tl }
                &&
                ! \tl_if_empty_p:c { g__polyomino_##1_\int_use:N \l__polyomino_x_int _tl }
                &&
                \int_compare_p:nNn { \l__polyomino_x_int } < { \l__polyomino_y_int + 1 }
              }
              { \int_incr:N \l__polyomino_x_int }
            \int_compare:nNnT { \l__polyomino_x_int } > { \l__polyomino_col_int + 1 }
              {
                \draw [ / polyomino / grid_style ]
                  ( \int_use:N \l__polyomino_col_int , -##1 ) -- ( \int_use:N \l__polyomino_x_int - 1 , -##1 ) ;
              }
            \int_set_eq:NN \l__polyomino_col_int \l__polyomino_x_int
          }
      }
    \int_set:Nn \l__polyomino_x_int { \seq_count:N \l__polyomino_cols_seq }
    \int_step_inline:nn { \l__polyomino_max_int - 1 }
      {
        \int_zero:N \l__polyomino_row_int
        \int_zero:N \l__polyomino_y_int
        \int_while_do:nNnn { \l__polyomino_y_int } < { \l__polyomino_x_int }
          {
            \bool_do_while:nn
              {
                \tl_if_eq_p:cc
                  { l__polyomino_\int_use:N \l__polyomino_y_int _##1_tl }
                  { l__polyomino_\int_use:N \l__polyomino_y_int _\int_eval:n { ##1 + 1 }_tl }
                &&
                ! \tl_if_empty_p:c { g__polyomino_\int_use:N \l__polyomino_y_int _##1_tl }
```

```
                      &&
                      \int_compare_p:nNn { \l__polyomino_y_int } < { \l__polyomino_x_int + 1 }
                      &&
                      \int_compare_p:nNn {##1} < { \seq_item:Nn \l__polyomino_cols_seq { \l__polyomino_y_int } + 0 }
                    }
                    { \int_incr:N \l__polyomino_y_int }
                  \int_compare:nNnT { \l__polyomino_y_int } > { \l__polyomino_row_int + 1 }
                    {
                      \draw [ / polyomino / grid_style ]
                        ( ##1 , -\int_use:N \l__polyomino_row_int ) -- ( ##1 , 1 - \int_use:N \l__polyomino_y_int ) ;
                    }
                  \int_set_eq:NN \l__polyomino_row_int \l__polyomino_y_int
                }
            }
        }
    }
  }

\endinput
```