

ffn2tex v1.00 - Alain Brobecker, 2012-2024

What is ffn2tex? page 2

How to use it: pre-processing page 2

Quick %CUBES reference and examples page 3

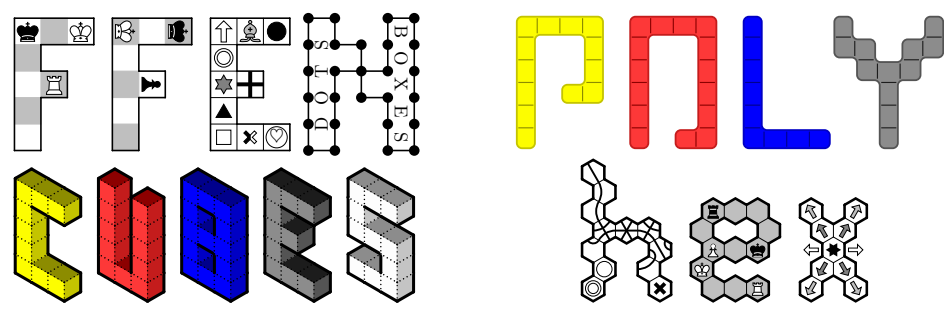
Quick %POLY reference and examples page 4

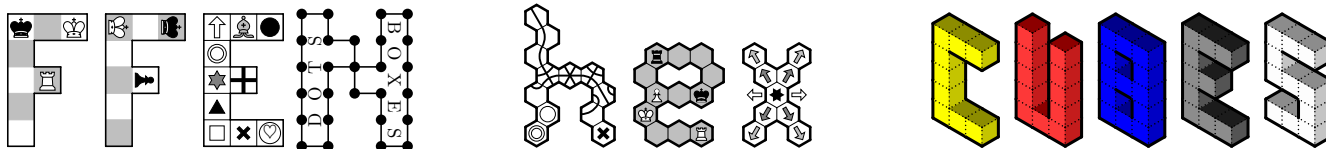
Quick %FFEN reference page 5

Quick %HEX reference page 6

%FFEN examples page 7-8

%HEX examples page 9-10





ffn2tex v1.00 - Alain Brobecker, 2012-2024

What is ffn2tex?

ffn2tex is a pre-processor to create **diagrams with square grids**, **diagrams with hexagonal grids** or **cubes setups** in the $\text{T}_{\text{E}}\text{X}$ typesetting system. For example you can make diagrams for Chess, fairy Chess, Checkers, Draughts, Go, and many other games. I also personally use it to create puzzle games...

The name comes from FEN (Forsyth-Edwards Notation) which was created to describe Chess positions (position=diagram+who has the move+castling rights+etc...). In 2002, Joost de Heer designed the **Fairy FEN** extension for fairy Chess diagrams (not positions! In fact François Labelle points out that it should have been called Fairy Forsyth, not Fairy FEN). I added some more possibilities for my own purpose. You will have more informations in the examples or by searching "Forsyth-Edwards Notation" on the internet.

How to use it: pre-processing

The following explanations were written for someone using a $\text{m}\$do\$$ system. I hope this example is enough to understand how to use **ffn2tex** and for people using another operating system.

Instead of adding commands to $\text{T}_{\text{E}}\text{X}$ i chose to make a program that must be executed on your **.tex** file before you process it with $\text{T}_{\text{E}}\text{X}$ (so it must be pre-processed by **ffn2tex**). It converts the **%FFEN** instructions into **pstricks** command, so at first you need to have the standard **pstricks** package installed. You must also **copy ffn2tex.prg** in your $\text{T}_{\text{E}}\text{X}$ work directory.

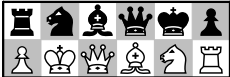
Now create a file named "sample.tex" in your $\text{T}_{\text{E}}\text{X}$ work directory and write the following in it:

```
\documentclass{article}
\usepackage{pstricks}
\begin{document}
Sample diagram:\\
%FFEN rnbqkp/PKQBNR
\end{document}
```

Then under the $\text{m}\$do\$$ prompt (or the command line of your OS) type the following:

```
c:\tex>ffn2tex sample.tex temporary.tex
c:\tex>latex temporary.tex
c:\tex>dvips temporary.ps
c:\tex>del sample.pdf
c:\tex>ps2pdf temporary.ps
c:\tex>rename temporary.pdf sample.pdf
c:\tex>del temporary.*
```

The resulting "sample.pdf" file contains the following:

```
Sample diagram:

```

Of course we don't want to type this sequence every time we process a **.tex** file, so we will use a batch (shell?) file to do it. I saved the following batch sequence in "goffen.bat" which is in my $\text{T}_{\text{E}}\text{X}$ work directory:

```
ffn2tex %1.tex temporary.tex
latex temporary.tex
dvips temporary.dvi
ps2pdf temporary.ps
del %1.pdf
rename temporary.pdf %1.pdf
del temporary.*
```

Then i only have to type this under the $\text{m}\$do\$$ prompt:

```
c:\tex>goffen sample
```

Some examples are following. The list of all variables, modifiers and pieces is in the Quick reference. Happy diagramming...

ffen2tex v1.00 - Quick %POLY reference and examples
Alain Brobecker, 2012-2024

Poly Modifiers: (default values are given, not reinitialised between two diagrams)

```
%POLY_SquareSize 0.7          (in centimeter)
%POLY_LineWidth 0.08
%POLY_LineArc 0.07
%POLY_Edge 0.03
%POLY_SeparationWidth 0.05
%POLY_SeparationEdge 0.05
%POLY_GridStyle 1             (0=no grid / 1=normal grid)
%POLY_BorderWidth 0.2
%POLY_InnerWidth 0.02
%POLY_BorderAll 1             (0=off / 1=on, affects all 4 borders variables below, and evaluated before them)
%POLY_BorderUp 1              (0=off / 1=on)
%POLY_BorderLeft 1            (0=off / 1=on)
%POLY_BorderDown 1           (0=off / 1=on)
%POLY_BorderRight 1          (0=off / 1=on)
%POLY_BoardRotation 0        (rotates board N*90° anticlockwise)
%POLY_ObjectWidth 0.1        (size for tick and cross)
```

Important: If %POLY_GridStyle is set to 1, you must define **boardcolor** beforehand with:

```
\definecolor{boardcolor0}{rgb}{1,1,1} %background (white)
\definecolor{boardcolor1}{rgb}{0.3,0.3,0.3} %lines (dark grey)
```

Draw polygons:

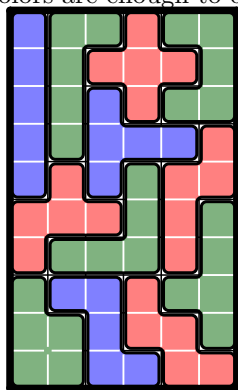
```
%POLY UpperRow/.../LowerRow
. empty
a-z for connex polygons with triples of colors defined beforehand in the file with:
\definecolor{a0}{rgb}{.55,.00,.00} %inner polygon
\definecolor{a1}{rgb}{.99,.33,.33} %border
\definecolor{a2}{rgb}{.99,.99,.99} %separation between squares
```

Piece modifiers:

```
!   ✓ draw a tick on a square, empty or inside a polygon (you must have defined tickcolor beforehand)
%   ✗ draw a cross, empty or inside a polygon (you must have defined crosscolor beforehand)
```

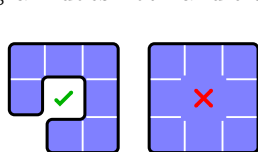
Examples:

First example is one of the many pentamino cutouts of a 6 × 10 rectangle in which no pieces of same color are adjacent. Three colors are enough to do this, but two colors are not.



```
\definecolor{boardcolor0}{rgb}{1,1,1} %white
\definecolor{boardcolor1}{rgb}{0,0,0} %black
\definecolor{r0}{rgb}{.5,.7,.5} %red
\definecolor{r1}{rgb}{0,0,0}
\definecolor{r2}{rgb}{1,1,1}
\definecolor{g0}{rgb}{1,.5,.5} %green
\definecolor{g1}{rgb}{0,0,0}
\definecolor{g2}{rgb}{1,1,1}
\definecolor{b0}{rgb}{.5,.5,1} %blue
\definecolor{b1}{rgb}{0,0,0}
\definecolor{b2}{rgb}{1,1,1}
%POLY bggrrg/bgrrrg/bgbrgg/bgbbbr/brbrrr/rrrg/rgggrg/gbbrrg/ggbrg/ggbbrr
```

Second example shows the use of ticks and crosses to illustrate that your polygon must be without closed holes, since the program does not handle them the expected way:



```
\definecolor{tickcolor}{rgb}{0,.7,0}
\definecolor{crosscolor}{rgb}{1,0,0}
%POLY_GridStyle 0
%POLY bbb/b!.b/.bb
~~%POLY bbb/b%.b/bbb
```

Some more remarks:

- POLY_LineWidth, POLY_SeparationWidth, POLY_SeparationEdge, POLY_InnerWidth, POLY_BorderWidth and POLY_ObjectWidth are relative to POLY_SquareSize, so the real width for POLY_LineWidth is $0.08 \times 0.7 = 0.56$ cm.
- To have polygons with special rendering in the **Polyssimo Challenge** documents, i used my program MSR.exe (Multiple Search and Replace) after ffen2tex.exe but before latex.exe with the following kind of commands:

```
#,fillstyle=solid,fillcolor=m0#,fillstyle=vlines*,hatchsep=2pt,fillcolor=white#
```

ffn2tex v1.00 - Quick %FFEN reference
Alain Brobecker, 2012-2024

Draw a diagram: %FFEN UpperRow/.../LowerRow

Board Modifiers: (default values are given, not reinitialised between two diagrams)

- %FFEN_SquareSize 0.5
- %FFEN_ThinLineWidth 0.03
- %FFEN_BoldLineWidth 0.15
- %FFEN_Fill 2 (0=none / 1=odd / 2=even (Chess) / 3=full)
- %FFEN_FillStyle 0 (0=lightgray / 1=gray / 2=black / 3=vlines / 4=hlines / 5=crosshatch / 6=centered dot)
- %FFEN_InnerStyle 0 (0=none / 1=solid / 2=dashed / 3=dotted)
- %FFEN_InnerWidth 0.03
- %FFEN_BorderStyle 1 (0=none / 1=solid / 2=dashed / 3=dotted)
- %FFEN_BorderWidth 0.06
- %FFEN_BorderAll 1 (0=off / 1=on, affects all 4 borders variables below, and evaluated before them)
- %FFEN_BorderUp 1 (0=off / 1=on)
- %FFEN_BorderLeft 1 (0=off / 1=on)
- %FFEN_BorderDown 1 (0=off / 1=on)
- %FFEN_BorderRight 1 (0=off / 1=on)
- %FFEN_PieceCount 0 (0=off / 1=W+B / 2=?+?=W+B)
- %FFEN_CornersRadius 0 (0=off / ?=radius)
- %FFEN_BoardRotation 0 (rotates board N*90° anticlockwise, if N> 4 pieces are not rotated, > and ^ won't work)

I also added the %FFEN_HBorderStyle, %FFEN_VBorderStyle, %FFEN_HBorderWidth, %FFEN_VBorderWidth, %FFEN_HInnerStyle, %FFEN_VInnerStyle, %FFEN_HInnerWidth and %FFEN_VInnerWidth modifiers.

Warnings: If you change more than once a value before using %FFEN only the first one is taken in account. The values are not reinitialised after a diagram. %FFEN_BorderAll is evaluated before the other %FFEN_Border modifiers.

Pieces:

char	piece	output	char	piece	output	char	piece	output
K / k	White / Black King		&	Heart		u	Up Bar	
Q / q	W/B Queen		"	Diamond			Vertical Bar	
R / r	W / B Rook		!	Spade		,	Small Vertical Bar	
B / b	W / B Bishop		?	Club		U	Up Left Bar	
N / n	W / B kNight		J	Up Left Arc		y	Vertical Left Bar	
P / p	W / B Pawn		j	Truchet Arcs		+	Cross Bars	
C / c	W / B Circle		.	Bold Circle		\	Diagonal	
D / d	W / B Dame					%	Cross Diagonals	
O / o	W / B Small Circle					G	Greater	
T / t	W / B Triangle							
S / s	W / B Square					L	V Line Fill	
X / x	W / B Cross		:	Thin Cross		l	H Line Fill	
E / e	W / B Star		'a	Symbol		F	CrossHatch Fill	
A / a	W / B Up Arrow		"42	Double Symbol		f	White Fill	

Valid symbols are !"#\$%&()*+,-./0..9;=<=>?@A..Z[\]^_`a..z

Double symbols are inserted using two ' characters, not the double quote!

TeX instructions and macros can be inserted as a piece using the { and } delimiters (no space!).

Piece Modifiers:

- *N rotated N*90° anticlockwise when N ∈ [0;3], and rotated 45°+(N-4)*90° when N ∈ [4;7]
- or = lightgray fill or gray fill for white pieces (♡ and ◇ are not white), B&W fill on some black pieces (eg: -x)
- < or > piece is between this square and left square or between this square and right square
- ^ or v piece is between this square and up square or between this square and down square
- @ piece is circled (same size as White Circle, but transparent)

Square Modifiers: (can apply on many empty squares)

- \$ change square(s) fill property
- # square(s) outside board (coloring affected by \$ but not by %FFEN_Fill)
- [and/or] border on the left and/or right of next square(s)
- ~ and/or - border above and/or below next square(s)

Draw a diagram: %HEX UpperRow/.../LowerRow

Board Modifiers: (default values are given, not reinitialised between two diagrams)

- %HEX_HexRadius 0.5
- %HEX_FirstShift 0 (0=upper line start on left / 1=upper line is shifted one hexagon right)
- %HEX_ThinLineWidth 0.05
- %HEX_BoldLineWidth 0.15
- %HEX_InnerStyle 1 (0=none / 1=solid / 2=dashed / 3=dotted)
- %HEX_InnerWidth 0.03
- %HEX_BorderStyle 1 (0=none / 1=solid / 2=dashed / 3=dotted)
- %HEX_BorderWidth 0.06
- %HEX_BorderAll 1 (0=off / 1=on, affects all 6 borders variables, and evaluated before them)
- %HEX_BorderLeft 1 (0=off / 1=on)
- %HEX_BorderDownLeft 1 (0=off / 1=on)
- %HEX_BorderDownRight 1 (0=off / 1=on)
- %HEX_BorderRight 1 (0=off / 1=on)
- %HEX_BorderUpRight 1 (0=off / 1=on)
- %HEX_BorderUpLeft 1 (0=off / 1=on)
- %HEX_PieceCount 0 (0=off / 1=W+B / 2=?+?=W+B)
- %HEX_CornersRadius 0 (0=off / ?=radius)

Warnings: If you change more than once a value before using %HEX only the first one is taken in account. The values are not reinitialised after a diagram. %HEX_BorderAll is evaluated before the other %HEX_Border modifiers.

Pieces:

char	piece	output	char	piece	output	char	piece	output
K / k	White / Black King		&	Heart		N	PathN	
R / r	W / B Rook		”	Diamond		U	PathU	
P / p	W / B Pawn		!	Spade		V	PathV	
S / s	W / B Square		?	Club		W	PathW	
X / x	W / B Cross		:	Thin Cross		H	PathH	
C / c	W / B Circle		.	Bold Circle		n/m	Path n/m	
D / d	W / B Dame					u	Path u	
O / o	W / B Small Circle		j	Small Arc		v	Path v	
T / t	W / B Triangle		J	Big Arc		w/y/z	Path w/y/z	
A / a	W / B Up Arrow		\	Diagonal		h/i	Path h/i	
E / e	W / B Star		'a	Symbol		"42	Double Symbol	

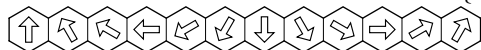
Valid symbols are !"#\$%&()*+,-./0..9:;<=>?@A..Z[\]^_`a..z

Double symbols are inserted using two ' characters, not the double quote!

TeX instructions and macros can be inserted as a piece using the { and } delimiters (no space!).

Piece Modifiers:

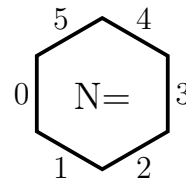
*N rotated N*30° anticlockwise with N ∈ {0;1;2;3;4;5;6;7;8;9;A;B} with A=10 and B=11, see example below:



- or = lightgray fill or gray fill (only for white pieces, ♥ and ◇ are not white)
- @ piece is circled (same size as White Circle, but transparent)

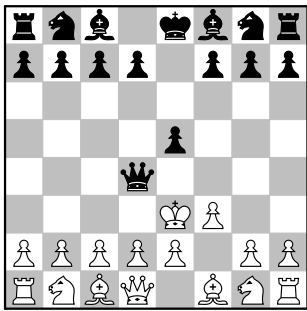
Hexagon Modifiers: (can apply on many empty hexagons)

- \$ or % change hexagon(s) fill property to lightgray or darkgray
- # hexagon(s) outside board
- [N a borderline is added on border N of the hexagon (see aside for values of N)



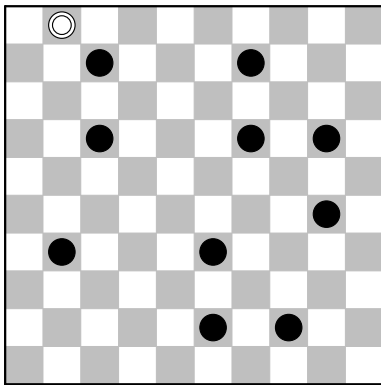
ffn2tex v1.00 - %FFEN examples
Alain Brobecker, 2012-2024

Since **ffn2tex** is based upon the Forsyth-Edwards Notation, the default settings are for Chess diagrams. The one below has been reached after black's third move, how did the game went?



`%FFEN rnb1kbnr/pppp1ppp/8/4p3/3q4/4KP2/PPPPP1PP/RNBQ1BNR`

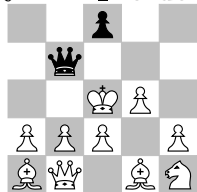
Diagrams can have any size between 1×1 and 99×99 , but for empty areas of width bigger than 9 squares you'll have to use two digits or more. Here a width of 10 empty squares is given as two empty areas of width 5, ie 55, but it could have been 64 or 73... Below is an international Draughts game with `%FFEN_PieceCount` set.



`%FFEN_PieceCount 1`
`%FFEN 1D8/2c3c3/55/2c3c1c1/55/8c1/1c3c4/55/5c1c2/55`
 White plays and wins.

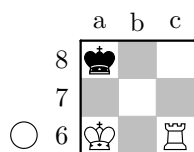
1+10: White plays and wins.

We show here a 5×5 part of the first example with only the down border. The `%FFEN_Fill` variable was changed to `1=odd` (ie upper left is filled) to have the correct square coloring. **Note that %FFEN_BorderAll is evaluated before the other %FFEN_Border modifiers.**



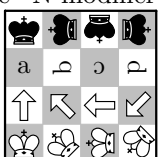
`%FFEN_PieceCount 0`
`%FFEN_Fill 1`
`%FFEN_BorderAll 0`
`%FFEN_BorderDown 1`
`%FFEN 2p2/1q3/2KP1/PPP1P/BQ1BN`

To put coordinates we insert letters and figures with the `'` escape character (double `'` for text with two characters), and put them outside the board using the `#` modifier. It's tedious but flexible. **Note that %FFEN_Fill is still 1, because the values are not reinitialised between two diagrams.** Upper left corner would be colored if it were not outside the board. I added a white circle indicating who has to move, and yes, it's mate in 1.



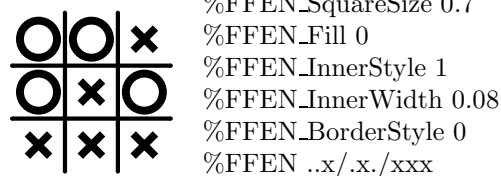
`%FFEN_BorderAll 0`
`%FFEN_BorderUp 1`
`%FFEN_BorderLeft 1`
`%FFEN #2#'a#'b#'#1#'8k2/#1#'73/#C#'6K1R`

With the `*N` modifier we can rotate the pieces, this is often used for fairy pieces in Chess problems.

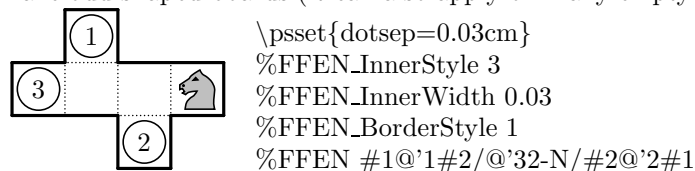


`%FFEN_Fill 2`
`%FFEN_BorderAll 1`
`%FFEN k*1k*2k*3k/'a*1'b*2'c*3'd/A*4A*1A*5A/K*4K*1K*5K`

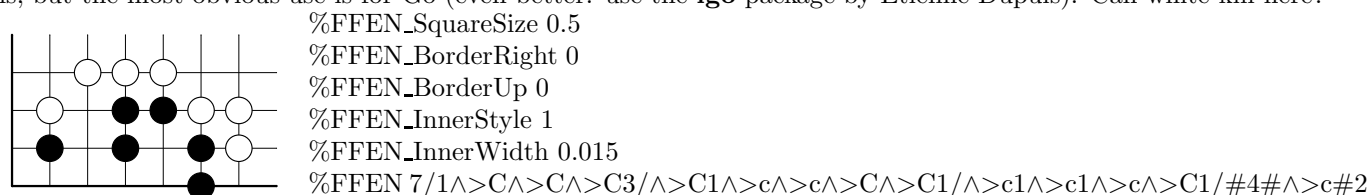
By modifying the size, the inner lines, the borders and the filling, we leave the world of Chess/Draughts/Checkers for Tic-tac-toe. What was the last move here?



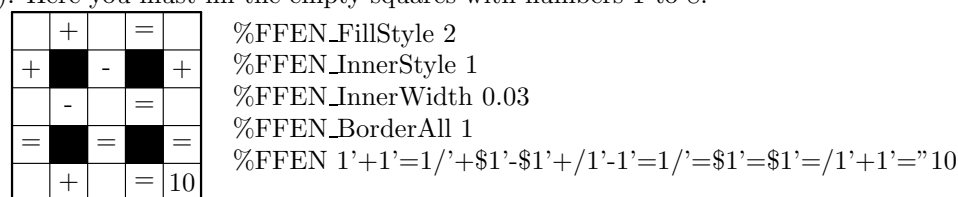
White pieces can be colored with the - or = modifiers. Pieces can be circled with the @ modifier. And using the # modifier we can make odd-shaped boards (it can also apply on many empty squares).



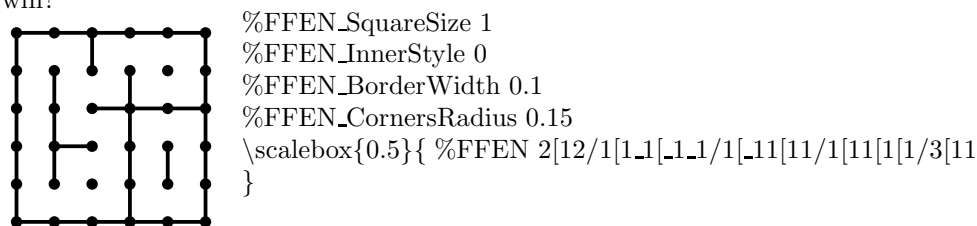
With the ^ and > modifiers it's possible to put pieces inbetween squares. This is used in Chess for some retrograde analysis problems, but the most obvious use is for Go (even better: use the **igo** package by Étienne Dupuis). Can white kill here?



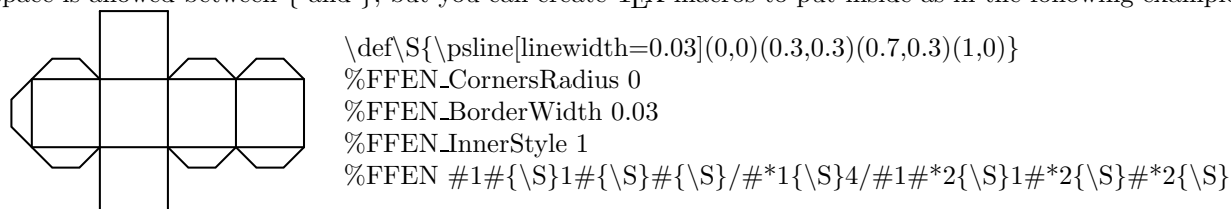
The \$ modifier changes the fill property of a particular square (it can also apply on many empty squares and to outside squares). Here you must fill the empty squares with numbers 1 to 8.



With the %FFEN_CornersRadius and the [,], ~ and _ modifiers you can make "Dots and boxes" diagrams. What to play here to win?



Finally you can use the { and } delimiters and directly insert psscript code to create whatever symbol which is not yet present. No space is allowed between { and }, but you can create T_EX macros to put inside as in the following example.



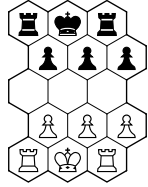
Some more remarks:

- When FFEN_Fill is set to 2 (even), the upper left corner is empty, when it's set to 1 (odd) the upper left corner is filled.
- FFEN_BoldLineWidth is used for Bold Circle (.), Bars, Diagonals and Arcs.
- FFEN_ThinLineWidth is used for circling pieces, White pieces and Thin Cross.
- FFEN_ThinLineWidth, FFEN_BoldLineWidth, FFEN_InnerWidth and FFEN_BorderWidth and everything related to them are relative to FFEN_SquareSize, so by default the real width for FFEN_ThinLineWidth is $0.03 \times 0.5 = 0.015$ cm.
- The centered dots (FFEN_FillStyle 6) have diameter of 3 times FFEN_BorderWidth.
- Symbols, figures and letters are inserted using `"\rput[B](0.5*FFEN_SquareSize,0.25*FFEN_SquareSize){42}"`, so they look best with a FFEN_SquareSize of 0.5, size being changed afterward with a `\scalebox`.

ffn2tex v1.00 - %HEX examples
Alain Brobecker, 2012-2024

For %HEX I tried to keep as many similarities with %FFEN as possible. The main differences are: the cells can now be colored in lightgray or darkgray, the borderline handling is a bit different, the pieces can no more be put between two hexagons, some pieces were removed, others were added.

Hexagons and chess pieces are not going that well together, thus i only included Kings, Rooks and Pawns (the movements of the latter will be different from standard chess), and there's no coloring with default settings. One very important parameter is the %HEX_FirstShift that affects the shift of the upper line as can be seen in the two examples below.

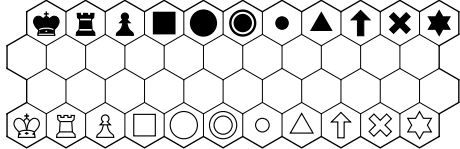


%HEX_HexRadius 0.3
 %HEX_FirstShift 0
 %HEX rkr/ppp/3/PPP/RKR



%HEX_FirstShift 1
 %HEX rkr/ppp/3/PPP/RKR

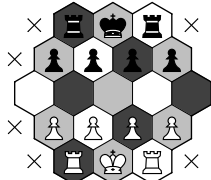
Diagrams can have any size between 1×1 and 99×99 , but for empty areas of width bigger than 9 hexagons you'll have to use two digits or more. Here a width of 11 empty squares is given as 65 since $6+5=11$. It can also be given as 74, 83... With the %HEX_PieceCount variable set, white and black pieces are counted, and the result is put below the diagram. Please note that %HEX_FirstShift is still set to 1, as in previous example since I didn't modified it.



%HEX_PieceCount 1
 %HEX krpscdotaxe/65/74/KRPSCDOTAXE
 \$\leftarrow\$ white+black pieces

11+11: \leftarrow white+black pieces

The # hexagon modifier puts an hexagon outside the borders, the inner lines will not be drawn between outside hexagons. Here i put Thin Crosses in the outside hexagons so that you can see them. I also used the \$ and % hexagon modifiers to display an important coloration of hexagonal boards, in which two adjacent hexagons have different colors. Some chess variants then say that bishops are pieces that do move on same colored hexagons, thus we need three of them and they are feeble pieces. I decided not to include them, I never was satisfied with chess variants playing on hexagons...

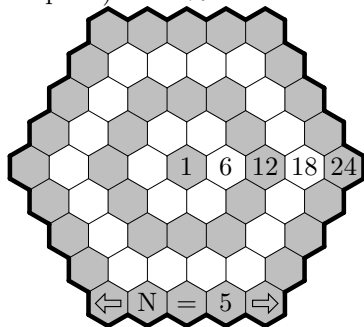


%HEX_FirstShift 1
 %HEX_PieceCount 0
 %HEX #:%1\$11#:/#:\$11\$1\$1/1%1\$11%1/#:\$11\$1\$1/#:%1\$11#:

If you want an hexagonal board made of hexagonal cells, the following one is made of $1+6+12+18+24=61$ hexagons. You can wipe the outer ring to have a board made of $1+6+12+18=37$ hexagons, or wipe two rings to have one made of $1+6+12=19$ hexagons... And in general, for a board with outer length N we will have $1 + 3 \times (N - 1) \times N$ hexagons.

(because $1 + 6 \times 1 + 6 \times 2 + \dots + 6 \times (N - 1) = 1 + 6 \times \sum_{k=1}^{N-1} k = 1 + 6 \times \frac{(N - 1) \times N}{2}$)

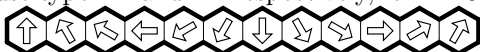
I took the opportunity to increase the %HEX_BorderWidth parameter, and i inserted text with ' or " (two ' character, not the double quote). The %HEX command must be on one line only, but it had to be separated in three to be shown.



%HEX_FirstShift 0
 %HEX_BorderWidth 0.2
 %HEX #2\$5#2/#1\$14\$1#2/#1\$11\$31\$1#1/\$11\$12\$11\$1#1/...
 ...\$11\$11\$1'6\$'12"18\$'24/\$11\$12\$11\$1#1/#1\$11\$31\$1#1/...
 ...#2\$*3-A\$'N\$'=\$'5\$*9-A#2

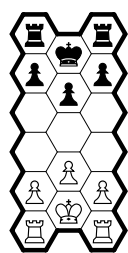
Here $N = 5$, thus we have $1 + 3 \times (5 - 1) \times 5 = 61$ hexagons.

The pieces can be rotated anticlockwise by steps of 30 degrees, with the *N piece modifier. For 300 and 330 degrees rotations you must type *A and *B respectively, ie A=10 and B=11 as in hexadecimal notation.



%HEX A*1A*2A*3A*4A*5A*6A*7A*8A*9A*AA*BA

Sometimes you will need the hexagons oriented in another way. Sadly i didn't include this feature, but you will be able to do that with the `\rotatebox` command of the `graphics` package. In the following example the Pawns can behave as in standard chess: move forward and capture sideways.

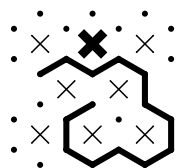
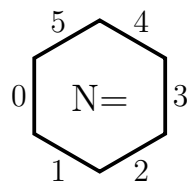


```
\usepackage{graphics}
...
\rotatebox{90}{
%HEX_FirstShift 0
%HEX *9R*9P2*9p*9r/*9K*9P1*9p*9k#1/*9R*9P2*9p*9r
}
```

You can modify the borders and inner lines definition with the `%HEX_Border` and `%HEX_Inner` modifiers. Note that `%HEX_BorderAll` is evaluated before the other `%HEX_Border` modifiers.

You can also add borderlines to an individual hexagon with the `[N` modifier, even if it's inside the board. You can add more than one borderline if you put a succession of `[N` modifiers, for example `[1[2x` puts both borders on the bottom of the hexagon marked with a big cross. Also note that as all modifiers it can apply to more than one hexagon.

`%HEX_CornersRadius` is used to put dots on all corners of all inside hexagons (not of outside hexagons).



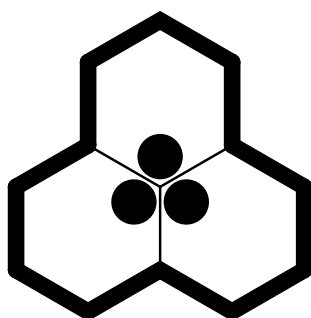
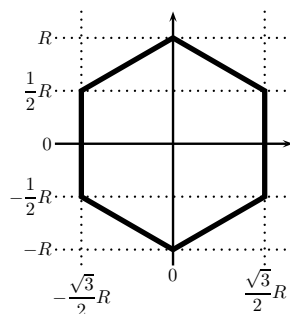
```
%HEX_HexRadius 0.4
%HEX_InnerStyle 0
%HEX_BorderAll 0
%HEX_CornersRadius 0.1
%HEX [2:[1[2x[1:/:[3:#1:/:[0[1[2[5:[1[2[3[4:
```

With the `-` and `=` modifiers, you can change the color of a white piece to lightgray or darkgray. Please note that `♡` and `◇` are not white. You can also circle a piece with the `@` modifier.



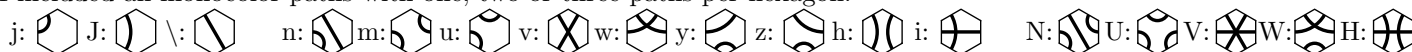
```
%HEX_BorderAll 1
%HEX_InnerStyle 1
%HEX #@S@-T@=E/@&@-@"*7=A
```

Finally you can use the `{` and `}` delimiters and directly insert pscript code to create whatever symbol which is not yet present. No space is allowed between `{` and `}`, but you can create \TeX macros to put inside as in the following example. The coordinate system of the hexagon is given on the left, with $R = \%HEX_HexRadius$.



```
\def\eye{\pscircle*(0.7,0){0.3}}
%HEX_HexRadius 1.1
%HEX #1*6{\eye}/*A{\eye}*2{\eye}
```

I included all monocolour paths with one, two or three paths per hexagon:



Some more remarks:

- `%HEX_BoldLineWidth` is used for Bold Circle (`.`), Diagonals, Arcs and Paths.
- `%HEX_ThinLineWidth` is used for circling pieces, White pieces and Thin Cross.
- `%HEX_ThinLineWidth`, `%HEX_BoldLineWidth`, `%HEX_InnerWidth` and `%HEX_BorderWidth` and everything related to them are relative to `%HEX_HexRadius`, so by default the real width for `%HEX_ThinLineWidth` is $0.03 \times 0.5 = 0.015$ cm.
- Symbols, figures and letters are inserted using `"\rput[B](0,-0.1){42}"`, so they look best with a `%HEX_HexRadius` of 0.3, size being changed afterward with a `\scalebox`.