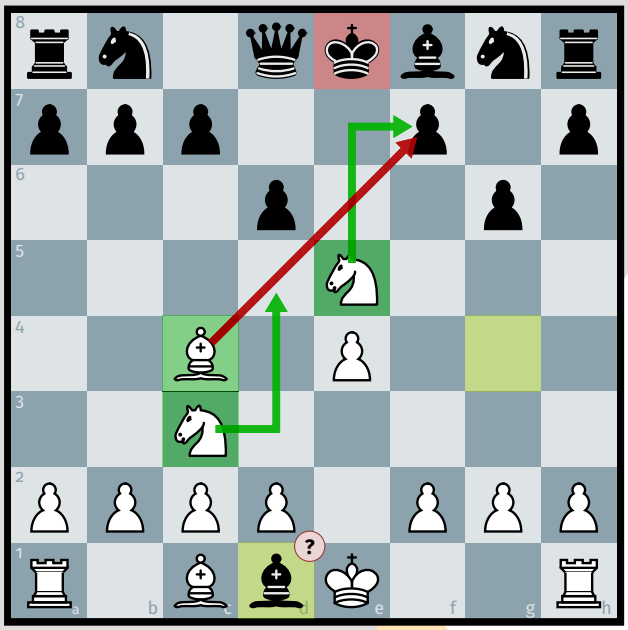
MPchess

drawing chess boards and positions with METAPOST



Contributor
Maxime CHUPIN
notezik@gmail.com

Version 0.1, 2023, March, 23th
https://plmlab.math.cnrs.fr/mchupin/mpchess

Abstract

This package allows you to draw chess boards and positions. The appearance of the drawings is modern and largely inspired by what is offered by the excellent web site Lichess.org. Relying on METAPOST probably allows more graphic flexibility than the excellent LaTeX packages.

https://plmlab.math.cnrs.fr/mchupin/mpchess

Contents

This package is in beta version, do not hesitate to report bugs, as well as requests for improvement.

1 Installation

MP*chess* is on the CTAN and can be installed via the package manager of your distribution.

https://www.ctan.org/pkg/mpchess

1.1 With TeXlive under Linux or MacOS

To install MP*chess* with TeXlive, you will have to create the directory `texmf` directory in your home.

```
user $> mkdir ~/texmf
```

Then, you will have to place the `.mp` files in the

~/texmf/tex/metapost/mpchess/

MP*chess* consists of 7 files METAPOST :

- `mpchess.mp`;
- `mpchess-chessboard.mp`;
- `mpchess-pgn.mp`;
- `mpchess-fen.mp`;
- `mpchess-cburnett.mp`;
- `mpchess-staunty.mp`;
- `mpchess-skak.mp`.

Once this is done, MP*chess* will be loaded with the classic

```
input mpchess
```

1.2 With MikTEX and Windows

These two systems are unknown to the author of MP*chess*, so we refer to their documentation to add local packages:

http://docs.miktex.org/manual/localadditions.html

1.3 Dependencies

MP*chess* depends on the packages METAPOST: hatching and, if MP*chess* is not used with LuaLATEX and luamplib, latexmp.

2 Why this package and general philosophy

There are already LATEX packages for drawing chess boards and positions, including the very good xskak [2]coupled with the package chessboard [1]. Ulrike Fisher has done there improvement, maintenance work, and provided us with excellent tools to make chess diagrams and to handle the different formats of game descriptions[1]. The documentations of these packages are very good.

Several things motivated the creation of MP*chess*. First of all, with chessboard the addition of a set of pieces is not very easy, because it relies on fonts. Moreover, I find that drawing chess game diagrams is something diagrams is a very graphical thing, and that using a dedicated drawing language offers more flexibility and what better than METAPOST [6].

With MP*chess*, we build the final image of the chess board with the pieces by successive layers. Thus, we begin by producing and drawing the board (backboard), which we can modify by coloring some squares for example, then we will add the pieces of the position (chessboard), and finally, we can annotate the whole with marks, colors, arrows, etc.

Moreover, MP*chess* produces images that are graphically close to what can be provided by excellent *open source* website https://lichess.org. You will see that the colors, the pieces and the general aspect are largely inspired of what this website offers.

3 Board

The board is called with MP*chess* backboard. You have to initialize the board before drawing it. This is done with the following command:

init_backboard

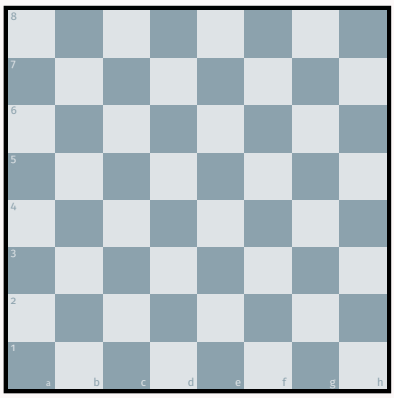
This command constructs a METAPOST picture named backboard. It should then be drawn as illustrated in the following example.

[1]She even developed the package to handle various chess fonts.

Exemple 1

```
input mpchess

beginfig(0);
init_backboard;
draw backboard;
endfig;
```



This initialization will make it possible to take into account the various options and features that we will describe in the following.

3.1 Setting the size

When creating the backboard, you can decide on the width of it. This is done with the following command:

`set_backboard_width(⟨dim⟩)`

⟨dim⟩: is the width of the board (with units). By default, this dimension is 5 cm.

The use of this command is illustrated in the example 2. This command should be used before `init_backboard` so that it is taken into account when creating the `picture`.

The size of the board can be retrieved with the following command:

`get_backboard_width`

This command returns a `numeric`.

3.2 Number of squares

By default, the game board contains 64 squares (8 × 8). You can change this with the following command:

`set_backboard_size(⟨nbr⟩)`

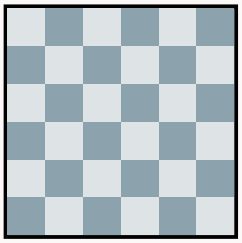
⟨nbr⟩: is the desired number of squares. The board will then be square of size ⟨*nbr*⟩×⟨*nbr*⟩. By default this number is 8.

Again, this command must be used before `init_backboard` for it to be taken into account as shown in the following example.

Exemple 2

```
input mpchess

beginfig(0);
set_backboard_width(3cm);
set_backboard_size(6);
init_backboard;
draw backboard;
endfig;
```



To obtain the size of the game board, you can use the following command the following command

`get_backboard_size`

This command returns a `numeric`.

3.3 Dimension of a square

Depending on the number of squares on the board and the prescribed width of the board, MP*chess* calculates the dimension (width or height) of a square. This serves as a general unit. To obtain it, use the following command.

`get_square_dim`

This command returns a `numeric`.

3.4 Setting the color theme

3.4.1 Predefined themes

Several color themes are available. To choose a color theme, use the following command:

`set_color_theme(⟨string⟩)`

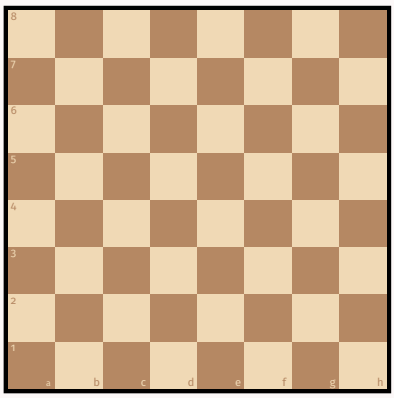
⟨***string***⟩ can be:

- `"BlueLichess"` (thème par défaut);
- `"BrownLichess"` ;
- or `"Classical"`.

The following examples show the results obtained.

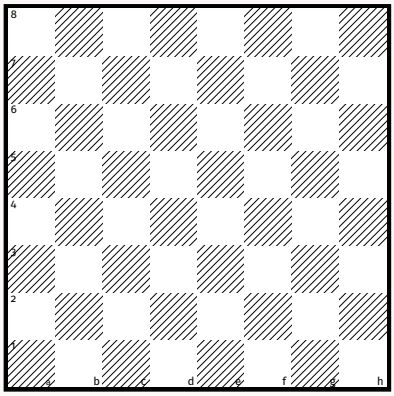
Exemple 3

```
input mpchess
beginfig(0);
set_color_theme("BrownLichess
      ");
init_backboard;
draw backboard;
endfig;
```



Exemple 4

```
input mpchess
beginfig(0);
set_color_theme("Classical");
init_backboard;
draw backboard;
endfig;
```



The two color themes provided borrow the colors of the Lichess themes.

3.4.2 Configuration of a personal theme

A color theme is really just the definition of two colors. These can be defined with the following commands.

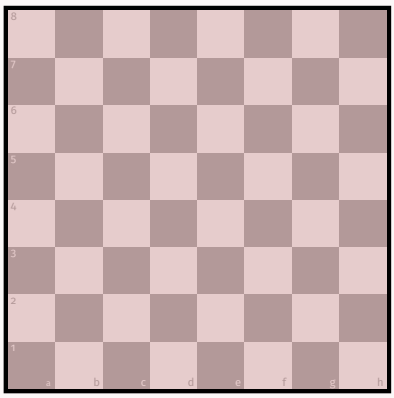
`set_white_color(`⟨*color*⟩`)`

`set_black_color(`⟨*color*⟩`)`

⟨*color*⟩ is a METAPOST `color`.

Exemple 5

```
input mpchess
beginfig(0);
set_white_color((0.9,0.8,0.8)
      );
set_black_color((0.7,0.6,0.6)
      );
init_backboard;
draw backboard;
endfig;
```



3.5 Display coordinates

You may have noticed in the various examples that by default, the coordinates are, as on the Lichess site, written in small letters inside the boxes.

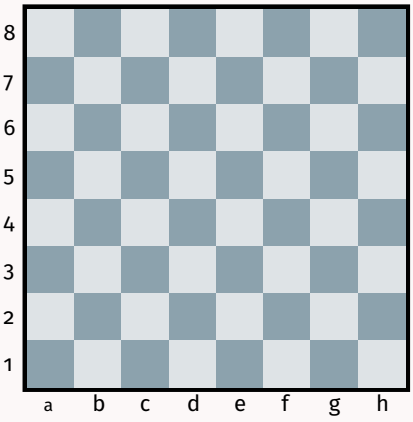
MP*chess* allows to position these coordinates outside the board with the following command.

`set_coords_outside`

The result is as follows.

Exemple 6

```
input mpchess
beginfig(0);
set_coords_outside;
init_backboard;
draw backboard;
endfig;
```



There is also a command to position the coordinates inside the board.

`set_coords_inside`

You can see in this documentation that with luamplib and LaTeX, the font is the font of the current document. To draw these letters and these numbers, MP*chess* uses the METAPOST operator `infont` and the font is set to `defaultfont` by default[2], so it is not possible to modify the composition font of the coordinates. This font can be changed with the following command.

[2]With luamplib the `infont` operator is redefined and its argument is simply ignored

`set_coords_font(⟨font⟩)`

It will then be necessary to use the naming conventions specific to the METAPOST operator `infont` and we refer to the documentation [6] for more details.

You can also delete the coordinates with the following command.

`set_no_coords`

And the reverse command also exists.

`set_coords`

3.6 White or black side

To choose if you want to see the tray on the white or black side, MP*chess* provides two commands.

`set_white_view`

`set_black_view`

By default, we see the board on the white side.

3.7 Players' names

You can fill in the names of the players so that they are noted around the chessboard. This is done with the following commands.

`set_white_player(⟨string⟩)`

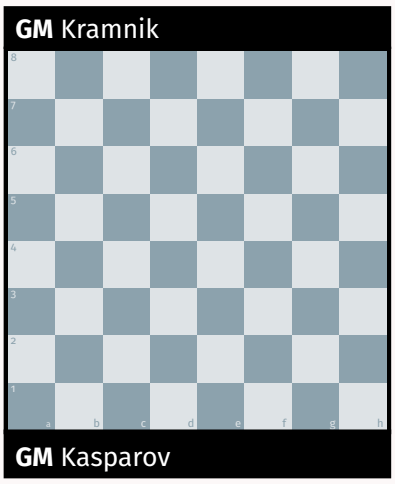
`set_black_player(⟨string⟩)`

⟨***string***⟩ **:** is the string interpreted by LaTeXto display.

Exemple 7

```
input mpchess
beginfig(0);
set_white_player("\textbf{GM}
      Kasparov");
set_black_player("\textbf{GM}
      Kramnik");

init_backboard;
draw backboard;
endfig;
```

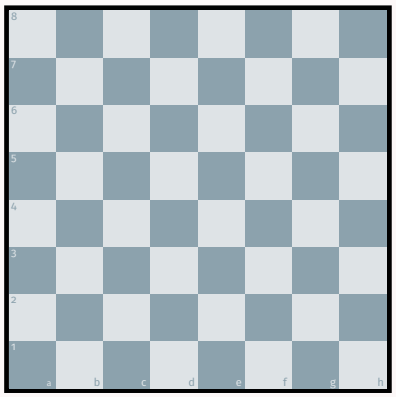


It is possible to place the names on the right side of the board without the black bands present by default. This happens either if the coordinates are printed outside the tray, or if the following command is used.

`set_players_side`

Exemple 8

```
input mpchess
beginfig(0);
set_white_player("\
      textbf{GM}
      Kasparov");
set_black_player("\
      textbf{GM}
      Kramnik");
set_players_side;
init_backboard;
draw backboard;
endfig;
```



GM Kramnik

GM Kasparov

4 Pieces et positions

MP*chess*, as described above, builds the picture of a chess position layer by layer. This part is dedicated to the configuration of pieces and positions.

Internally, MP*chess* builds a table on the board grid. Then, some macros allow to generate a `picture` to be drawn *over* the board (`backboard`).

4.1 Setting the theme of the pieces

MP*chess* provides for the moment three themes of pieces, two borrowed from Lichess, and the other borrowed from the package skak citectan-skak[3].

To choose the theme we will use the following command.

`set_pieces_theme(⟨string⟩)`

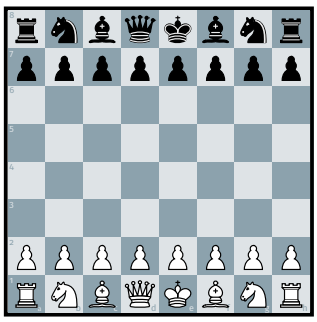
⟨***string***⟩**:** can be:

- `"cburnett"` (default), to get the Lichess *cburnett* pieces set;
- `"staunty"`, to get the Lichess *staunty* pieces set;
- `"skak"`, to get the skak pieces set.

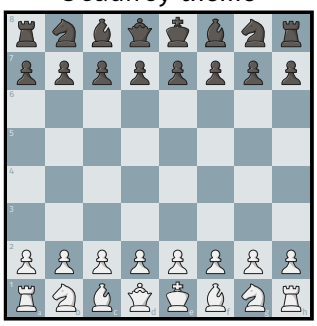
The table 1 shows the result of the three sets of pieces.

[3]Which provides the METAFONT code for a chess font, which has been easily adapted to METAPOST for MP*chess*.

cburnett theme



staunty theme



skak theme

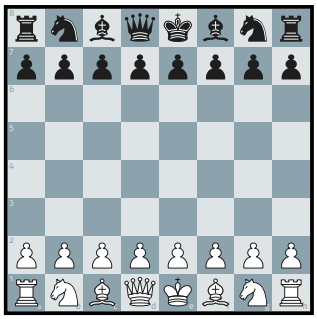


Table 1: The different themes of pieces provided by MP*chess*.

4.2 Who's to move

MP*chess* indicates who has the move between white and black. This is done by a small colored triangle (white or black) at the end of the board (which you can you can see in the following examples).

To specify who has to move we use the following commands.

```
set_white_to_move
```

```
set_black_to_move
```

By default, white has to move, and this information is displayed.

To activate or deactivate this display, use one of the following two commands.

```
set_whos_to_move
```

```
unset_whos_to_move
```

4.3 Draw position

The commands described below allow you to build a position in several ways (adding pieces one by one, reading a FEN file, etc.). Once a position has been constructed, it can be plotted using the following command which generates a METAPOST `picture`.

`chessboard`

The use of this command will be widely illustrated in the following examples.

4.4 Build a position

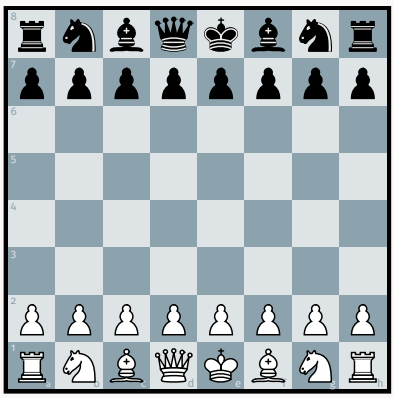
4.4.1 Initialization

To obtain the initial position of a game, simply use the following command.

`init_chessboard`

Exemple 9

```
input mpchess
beginfig(0);
init_backboard;
draw backboard;
init_chessboard;
draw chessboard;
endfig;
```



You can also initialize an empty `chessboard` with the following command.

`set_empty_chessboard`

4.4.2 Adding pieces

You can add pieces to build a position with the following two commands.

`add_white_pieces(⟨piece1⟩,⟨piece2⟩,etc.)`

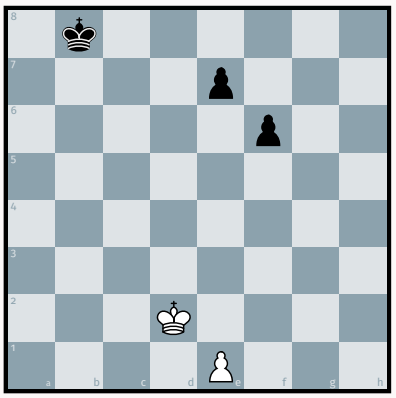
`add_black_pieces(⟨piece1⟩,⟨piece2⟩,etc.)`

These commands take lists of ⟨***piece***⟩ which are strings that describe the piece and the position using the algebraic notation. There is no limitation on the number of pieces in the list.

The following example illustrates the use of these commands.

Exemple 10

```
input mpchess
beginfig(0);
init_backboard;
draw backboard;
set_empty_chessboard;
add_white_pieces("e1","Kd2");
add_black_pieces("e7","f6","
      Kb8");
draw chessboard;
endfig;
```



4.4.3 Delete piece

MP*chess* provides several commands to remove items from a position.

The first command allows you to delete an item from a square. This command takes a list of squares, using algebraic notation.

`clear_squares(`⟨*square1*⟩`,`⟨*square2*⟩`,etc.)`

The ⟨***square1***⟩, ⟨***square2***⟩, etc., are strings, for example `"a3"`.

The following command allows to delete a set of squares in a region determined by two coordinates on the board. This command allows to take a list of regions.

`clear_areas(`⟨*area1*⟩`,`⟨*area2*⟩`,etc.)`

The ⟨***area1***⟩, ⟨***area2***⟩, etc., are strings consisting of two coordinates separated by a dash, for example `"a3-g7"`.

The following command deletes all the cells in a file (column) determined by a letter on the board. This command allows to take a list of files.

`clear_files(`⟨*file1*⟩`,`⟨*file2*⟩`,etc.)`

The ⟨***file1***⟩, ⟨***file2***⟩, etc., are strings of characters consisting of a letter, for example `"a"`.

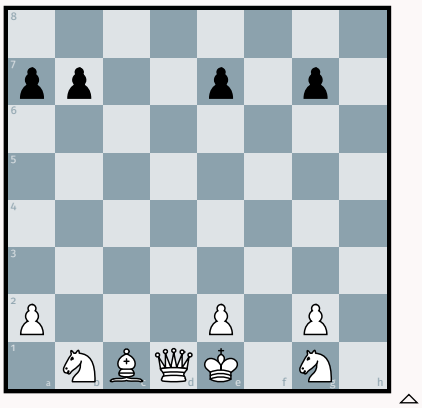
The following command deletes all the cells in a rank (line) determined by a number on the board. This command allows to take a list of ranks.

`clear_ranks(`⟨*rank1*⟩`,`⟨*rank2*⟩`,etc.)`

The ⟨***rank1***⟩, ⟨***rank2***⟩, etc., are strings made up of a number, for example `"4"`.
The use of all these commands is illustrated in the following example.

Exemple 11

```
input mpchess
beginfig(0);
init_backboard;
draw backboard;
init_chessboard;
clear_squares("a1","b2");
clear_areas("c2-d7");
clear_files("f","h");
clear_ranks("8");
draw chessboard;
endfig;
```



4.5 Reading data in FEN format

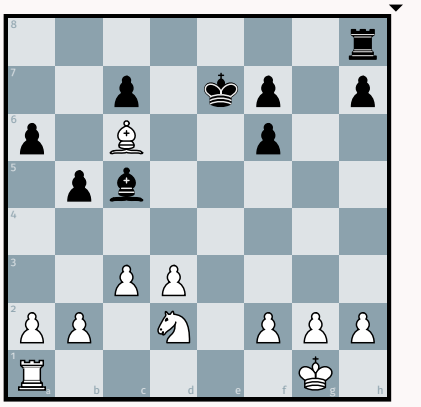
MP*chess* allows you to read a position in the FEN format thanks to the following command.

`build_chessboard_from_fen(`⟨*string*⟩`)`

⟨***string***⟩**:** is a string describing a position in FEN format. Note that all information after the w or b character is ignored.

Exemple 12

```
input mpchess;
beginfig(0);
init_backboard;
draw backboard;
string fenstr;
fenstr := "7r/2p1kp1p/p1B2p
      2/1pb5/8/2PP4/PP1N1PPP/
      R5K1 b - - 2 19";
build_chessboard_from_fen(
      fenstr);
draw chessboard;
endfig;
```



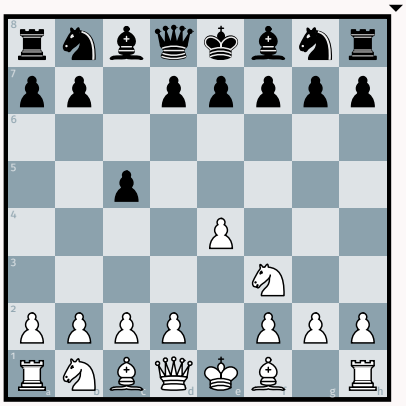
It is also possible to read an external file containing on the first line a string in the format FEN with the following command.

`build_chessboard_from_fen_file(`⟨*string*⟩`)`

⟨***string***⟩**:** is a string of characters (between double quotes) indicating the name of the file to read.

Exemple 13

```
input mpchess;
beginfig(0);
init_backboard;
draw backboard;
build_chessboard_from_fen_file
      ("test.fen");
draw chessboard;
endfig;
```



4.6 Reading data in PGN format

MP*chess* also allows to read a string in the format PGN. Attention, this is a partial management of the format, in particular MP*chess* does not manage the *tags* of the format. In reality, MP*chess* only handles the string describing the moves played. In the same way, the accepted format by MP*chess* does not allow variants or comments.

When such a functionality is used, MP*chess* stores all the intermediate positions and thus allows to represent them.

To construct the positions, we use the following command.

`build_chessboards_from_pgn(⟨string⟩)`

Once the positions are built, we can represent them with the following command.

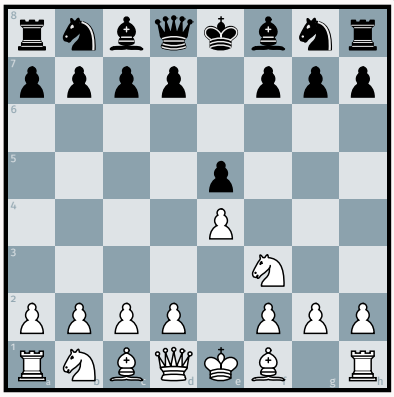
`chessboard_step(⟨int⟩)`

⟨***int***⟩**:** is the number of the step. The initial configuration is numbered 0, and then each move, white or black, is numbered.

This command, like `chessboard` (see page 12), returns a `picture`.
The following example illustrates the use of these commands.

Exemple 14

```
input mpchess;
string pgnstr;
pgnstr := "1. e4 e5 2. Nf3 Nc
      6 3. Nxe5 Nxe5 4. Bb5 c
      6";
build_chessboards_from_pgn(
      pgnstr);
beginfig(0);
init_backboard;
draw backboard;
draw chessboard_step(3); % Nf
      3
endfig;
```



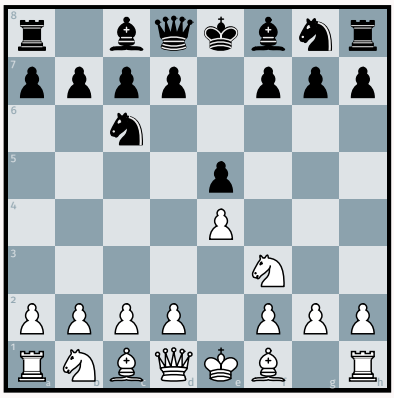
It is also possible to read an external file containing on the first line a string in the format PGN with the following command.

`build_chessboard_from_pgn_file(⟨string⟩)`

⟨***string***⟩**:** is a string of characters (between double quotes) indicating the name of the file to read.

Exemple 15

```
input mpchess;
build_chessboards_from_pgn_file
      ("test.pgn");
beginfig(0);
init_backboard;
draw backboard;
draw chessboard_step(4); %
      Nxe5
endfig;
```



4.6.1 Show last move

The last move can be displayed automatically with the following command.

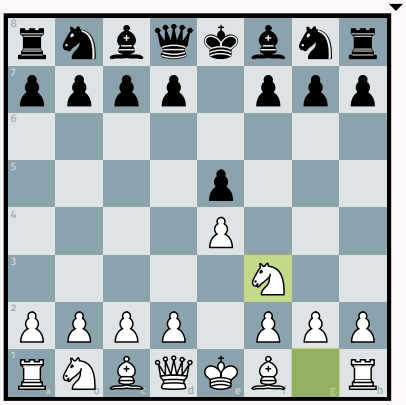
`show_last_move(⟨int⟩)`

⟨***int***⟩**:** is the number of the step. The initial setup is numbered 0, and then each move, white or black, is numbered.

This command colors in transparency the two squares where the last move starts and ends. Thus, it must be used between the drawing of the board (`draw backboard`) and the drawing of the pieces (`draw chessboard_step(i)`).

Exemple 16

```
input mpchess;
string pgnstr;
pgnstr := "1. e4 e5 2. Nf3 Nc
      6 3. Nxe5 Nxe5 4. Bb5 c
      6";
build_chessboards_from_pgn(
      pgnstr);
beginfig(0);
init_backboard;
draw backboard;
show_last_move(3);
draw chessboard_step(3); % Nf
      3
endfig;
```



You can configure the color used to color the squares of the last move with the following command.

`set_last_move_color(⟨color⟩)`

⟨***color***⟩: is a METAPOST `color`.

4.6.2 Get the number of moves

You can get the number of half moves with the following command.

`get_halfmove_number`

This command returns a `numeric`.

You can also get the total number of moves in the sense that they are numbered in the PGN format, with the following command:

`get_totalmove_number`

This command returns a `numeric`.

5 Annotation

Numerous commands allow to annotate the chessboard (arrow, color, circle, cross, etc.).

5.1 Arrows

The command for drawing arrows on the chessboard is the following.

`draw_arrows(⟨color⟩)(⟨string1⟩,⟨string2⟩, etc.)`

⟨***color***⟩: is a METAPOST `color`.

⟨***string1***⟩**:** is a string (between double-quotes) consisting of two coordinates (letter and number) separated by two characters which can be

`--` to connect the two squares in a straight line;

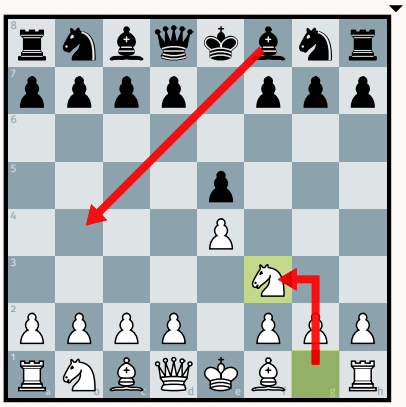
`-|` to connect the two squares in a broken line, first horizontally then vertically;

`|-` to connect the two squares in a broken line, first vertically then horizontally.

The following example illustrates the use of this command.

Exemple 17

```
input mpchess;
string pgnstr;
pgnstr := "1. e4 e5 2. Nf3 Nc
      6 3. Nxe5 Nxe5 4. Bb5 c
      6";
build_chessboards_from_pgn(
      pgnstr);
beginfig(0);
init_backboard;
draw backboard;
show_last_move(3);
draw chessboard_step(3); % Nf
      3
draw_arrows(red)("f8--b4","g
      1|-f3");
endfig;
```



The thickness of the arrows can be changed with the following command.

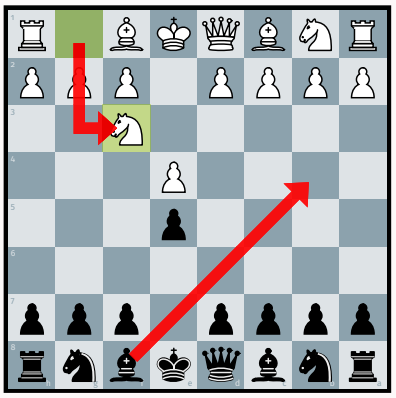
`set_arrow_width(`⟨*coeff*⟩`)`

⟨***coeff***⟩**:** is a coefficient (`numeric`) which allows you to adjust the width of the arrows in proportion to the width of a square on the chessboard. By default, this coefficient is `0.08`.

The following example illustrates the use of this command.

Exemple 18

```
input mpchess;
string pgnstr;
pgnstr := "1. e4 e5 2. Nf3 Nc
      6 3. Nxe5 Nxe5 4. Bb5 c
      6";
build_chessboards_from_pgn(
      pgnstr);
beginfig(0);
set_black_view;
init_backboard;
draw backboard;
show_last_move(3);
draw chessboard_step(3); % Nf
      3
set_arrow_width(0.12);
draw_arrows(red)("f8--b4","g
      1|-f3");
endfig;
```



5.2 Coloring squares

MPchess also allows you to color squares with the following command.

`color_square(`⟨*color*⟩`)(`⟨*coord1*⟩`,`⟨*coord2*⟩`, etc.)`

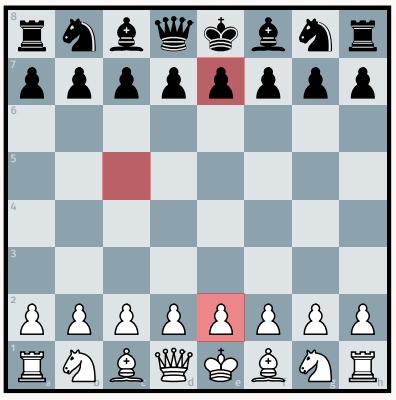
⟨***color***⟩: is a METAPOST `color`.

⟨***coord1***⟩: is a string (between double quotes) consisting of two coordinates (letter and number).

The following example illustrates the use of this command.

Exemple 19

```
input mpchess
beginfig(0);
init_backboard;
draw backboard;
color_square(red)("e2","e7","
      c5");
init_chessboard;
draw chessboard;
endfig;
```



This command colors the squares with a certain transparency to fit to the white and black squares.

5.3 Circles

MP*chess* allows you to surround squares with circles using the following command below.

`draw_circles(⟨color⟩)(⟨coord1⟩,⟨coord2⟩, etc.)`

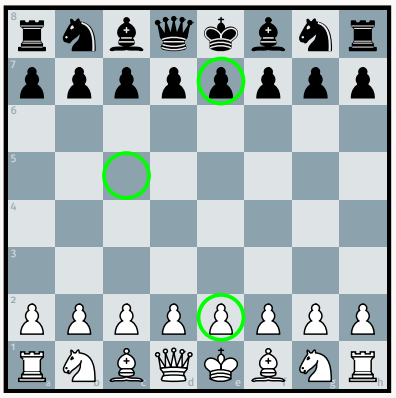
⟨***color***⟩: is a METAPOST `color`.

⟨***coord1***⟩: is a string (between double quotes) consisting of two coordinates (letter and number).

The following example illustrates the use of this command.

Exemple 20

```
input mpchess
beginfig(0);
init_backboard;
draw backboard;
init_chessboard;
draw chessboard;
draw_circles(green)("e2","e7"
      ,"c5");
endfig;
```



5.4 Crosses

MP*chess* allows you to draw crosses on squares with the following command.

`draw_crosses(⟨color⟩)(⟨coord1⟩,⟨coord2⟩, etc.)`

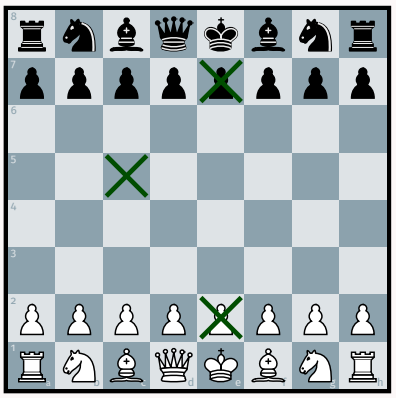
⟨***color***⟩: is a METAPOST `color`.

⟨***coord1***⟩: is a string (between double quotes) consisting of two coordinates (letter and number).

The following example illustrates the use of this command.

Exemple 21

```
input mpchess
beginfig(0);
init_backboard;
draw backboard;
init_chessboard;
draw chessboard;
draw_crosses(0.7[green,black
      ])("e2","e7","c5");
endfig;
```



5.5 Move comments

MP*chess* allows you to display the classic move comments of the type "!?" with the following command.

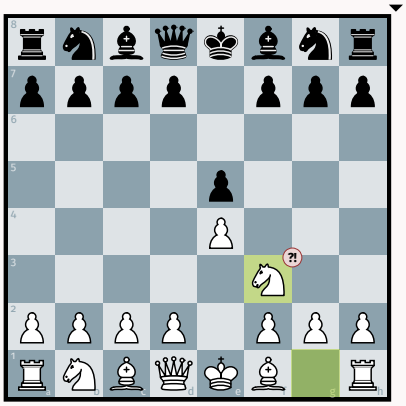
`draw_comment(⟨str⟩,⟨pos⟩)`

⟨***str***⟩: is a string (between double-quotes) to display.

⟨***pos***⟩: is a string (between double-quotes) consisting of a coordinate (letter and number).

Exemple 22

```
input mpchess;
string pgnstr;
pgnstr := "1. e4 e5 2. Nf3 Nc
      6 3. Nxe5 Nxe5 4. Bb5 c
      6";
build_chessboards_from_pgn(
      pgnstr);
beginfig(0);
init_backboard;
draw backboard;
show_last_move(3);
draw chessboard_step(3); % Nf
      3
draw_comment("?!","f3");
endfig;
```



The color of the comment annotation can be changed with the following command following command.

`set_comment_color(⟨color⟩)`

6 Miscellaneous

6.1 Reset the chessboard

To reset the internal structure storing the positions of the parts, you can use the following command.

```
clear_chessboard
```

6.2 Global reset

To reinitialize the values of the different parameters of MP*chess*, you can use the following command.

```
reset_mpchess
```

6.3 Clip the board

We can clip the chessboard with the following command.

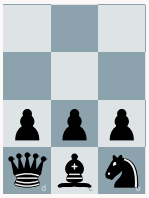
```
clip_chessboard(⟨string⟩)
```

⟨***string***⟩**:** is a string (between double quotes) composed of two coordinates (letter and number) separated by a dash, for example `"a1-c6"`.

Here is an example of an illustration.

Exemple 23

```
input mpchess;
string pgnstr;
pgnstr := "1. e4 e5 2. Nf3 Nc
      6 3. Nxe5 Nxe5 4. Bb5 c
      6";
build_chessboards_from_pgn(
      pgnstr);
beginfig(0);
set_black_view;
init_backboard;
draw backboard;
show_last_move(3);
draw chessboard_step(3); % Nf
      3
draw_comment("?!","f3");
clip_chessboard("e1-g4");
endfig;
```



7 Use with LaTeX

7.1 Use with pdfLaTeX or XeLaTeX

There are several ways to include the images produced by MP*chess* in a LaTeXdocument. The first is to generate pdf files with METAPOST and then to include them with `\includegraphics`. This solution works with all engines.

You can also use the packages gmp or mpgraphics with pdfLaTeX or XeLaTeX[4].

7.1.1 With mpgraphics

With mpgraphics [5], we load MP*chess* with the `mpdefs` environment and we can produce images with METAPOST code but without using `beginfig` and `endfig`, the code to generate a figure is in the `mpdisplay` environment. It will be necessary to use the option `-shell-escape` option when compiling the LaTeXdocument.

Here is a complete example of illustration.

```
\documentclass{article}
\usepackage{mpgraphics}
\begin{document}
\begin{mpdefs}
input mpchess
\end{mpdefs}
\begin{mpdisplay}
init_backboard;
draw backboard;
init_chessboard;
draw chessboard;
draw_arrows(red)("e7--e5","g1|-f3");
\end{mpdisplay}
\end{figure}
\end{document}
```

7.1.2 With gmp

The use of gmp [3] is quite similar to that of mpgraphics. Some commands are different, but as with mpgraphics, we do not use `beginfig` and `endfig`. The loading of MP*chess* can be done when loading the package, and the METAPOST code is in a `mpost` environment. Here again it will be necessary to compile the LaTeX document with the `-shell-escape` option.

Here is a complete example of illustration.

```
\documentclass{article}
\usepackage[shellescape, everymp={input mpchess;}]{gmp}
```

[4] We would like to thank Quark67 for the questions and advice

```
\begin{document}

\begin{mpost}
init_backboard;
draw backboard;
init_chessboard;
draw chessboard;
draw_arrows(red)("e7--e5","g1|-f3");
\end{mpost}
\end{document}
```

7.2 Use with LuaLaTeX and luamplib

It is possible to use MP*chess* directly in a LaTeX file with LuaLaTeX and the package luamplib. This is what is done to write this documentation.

It will then suffice to put the METAPOST code in the environment `mplibcode`.

For certain functionalities, MP*chess* uses the METAPOST operator `infont`. Thus, in order for the content of these features to be composed in the current font of the document, one must add the command :

```
\mplibtextextlabel{enable}
```

For more details on these mechanisms, we refer to the documentation of the package luamplib [4].

We can load globally MP*chess* with the following command.

```
\everymplib{input mpchess;}
```

Here is a complete example of illustration (to be compiled with LuaLaTeX).

```
\documentclass{article}
\usepackage{luamplib}

\everymplib{input mpchess;}

\begin{document}

\begin{mplibcode}
beginfig(0);
init_backboard;
draw backboard;
init_chessboard;
draw chessboard;
draw_arrows(red)("e7--e5","g1|-f3");
```

```
endfig;
\end{mplibcode}
\end{document}
```

8 To do

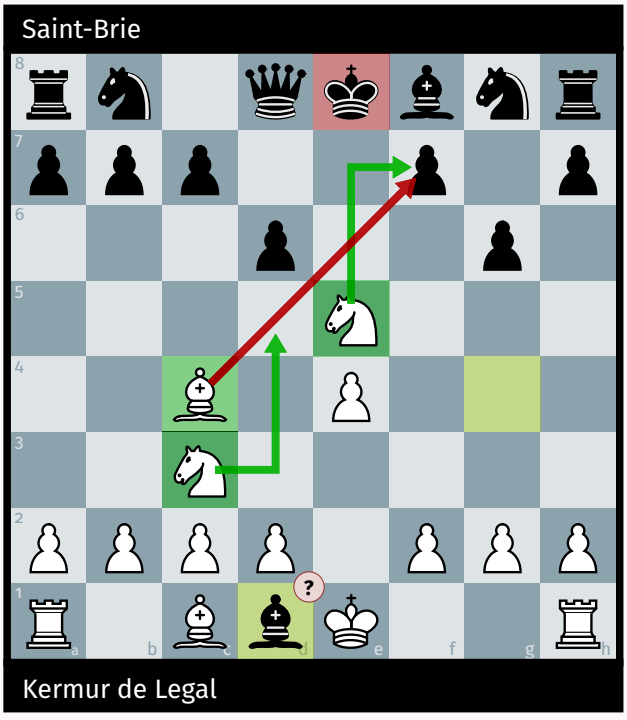
Many things can be added to MP*chess*. Among these, we can think of:

- display the captured pieces, or the differential of the captured pieces;
- afficher le temps restant pour chaque joueur ;
- show the accessible squares for a chosen piece (the Lichess points);
- display the *n* arrows indicating the *n* moves of the first lines of a position (with a decreasing thickness of the arrows);
- adding the coordinates outside the board when it is clipped;
- add piece themes.

9 Complete example

Exemple 24

```
input mpchess
string pgnstr;
pgnstr:="1. e4 e5 2. Bc4 d6 3. Nf3 Bg4 4. Nc3 g6 5. Nxe5 Bxd1";
build_chessboards_from_pgn(pgnstr);
beginfig(0);
set_white_player("Kermur de Legal");
set_black_player("Saint-Brie");
set_backboard_width(8cm);
init_backboard;
draw backboard;
show_last_move(10);
draw_comment("?","d1");
color_square(0.3[green,black])("c4","c3","e5");
color_square(0.3[red,black])("e8");
draw chessboard_step(10);
draw_arrows(0.3[green,black])("e5|-f7","c3-|d5");
draw_arrows(0.3[red,black])("c4--f7");
endfig;
```



References

[1] Ulrike Fischer. *The `chessboard` package. Print chess boards*. Version 1.9. Mar. 9, 2023. URL: https://ctan.org/pkg/chessboard.

[2] Ulrike Fischer. *The `xskak` package. An extension to the skak package for chess typesetting*. Version 1.5. Mar. 9, 2023. URL: https://ctan.org/pkg/xskak.

[3] Enrico Gregorio. *The `gmp` package. Enable integration between MetaPost pictures and LaTeX*. Version 1.0. June 24, 2016. URL: https://ctan.org/pkg/gmp.

[4] Hans Hagen et al. *The `luamplib` package. Use LuaTeX's built-in MetaPost interpreter*. Version 2.23.0. Jan. 12, 2022. URL: https://ctan.org/pkg/luamplib.

[5] Vafa Khalighi and bidi-tex GitHub Organisation. *The `mpgraphics` package. Process and display MetaPost figures inline*. Version 0.3. Sept. 8, 2019. URL: https://ctan.org/pkg/mpgraphics.

[6] The MetaPost Team and John Hobby. *The `metapost` package. A development of Metafont for creating graphics*. Aug. 26, 2021. URL: https://ctan.org/pkg/metapost.

Command Index