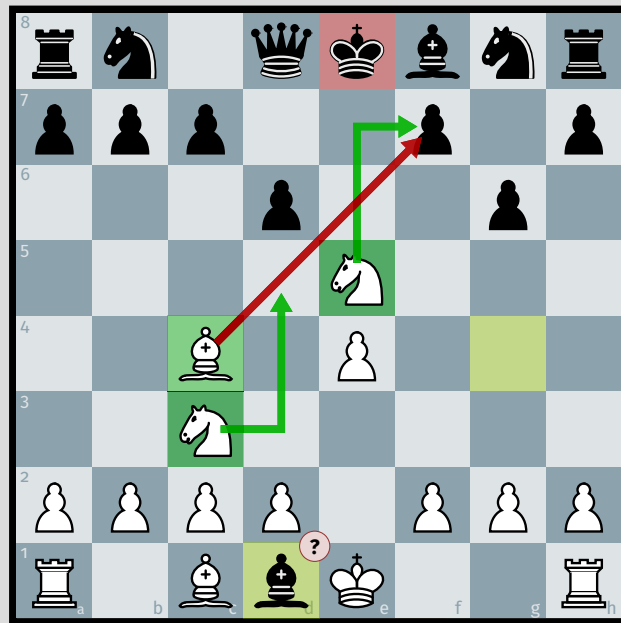


MPchess

drawing chess boards and positions with METAPOST



Contributor

Maxime CHUPIN

notezik@gmail.com

Version 0.3, 2023, April, 6th

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

Abstract

The *MPchess* package allows you to draw chess boards and positions. The appearance of its drawings is modern and largely inspired by what is offered by the excellent web site [Lichess.org](https://lichess.org). Relying on METAPOST probably allows more graphic flexibility than the excellent \LaTeX chess packages that already exist.

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

Contents

1	Installation	3
1.1	With \TeX live under Linux or MacOS	3
1.2	With Mik \TeX and Windows	4
1.3	Dependencies	4
2	Package Rationale and General Philosophy	4
3	Board	5
3.1	Size of the Board	5
3.2	Number of Squares	6
3.3	Dimension of a Square	6
3.4	Setting the Color Theme	6
3.4.1	Predefined Color Themes	6
3.4.2	Configuring a Personal Color Theme	7
3.5	Display Coordinates	8
3.6	White or Black Side	9
3.7	Players' Names	9
4	Pieces and Positions	10
4.1	Setting the Theme of the Pieces	11
4.2	Specifying to Move	11
4.3	Draw a Position	12
4.4	Build a position	12
4.4.1	Initialization	12
4.4.2	Adding Pieces	13
4.4.3	Deleting Pieces	13
4.5	Reading Data in FEN Format	14
4.6	Reading Data in PGN Format	15
4.6.1	Showing the Last Move	16
4.6.2	Getting the Number of Moves	17
5	Annotation	17
5.1	Arrows	17
5.2	Coloring Squares	19
5.3	Circles	20
5.4	Crosses	20
5.5	Move Comments	21
5.6	Main Lines	22
5.7	Possible Moves	23

6	Miscellaneous	23
6.1	Reset the chessboard	23
6.2	Global Reset	24
6.3	Clip the board	24
7	Use with \TeX	24
7.1	Use with pdf \TeX or X \TeX	24
7.1.1	With mpgraphics	25
7.1.2	With gmp	25
7.2	Use with Lua \TeX and luamplib	26
8	To Do	26
9	Complete Example	27
10	History	28
11	Acknowledgements	28

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

1 Installation

MPchess is on CTAN and can also be installed via the package manager of your distribution.

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

1.1 With \TeX Live under Linux or MacOS

To install MPchess with \TeX Live, 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/metapost/mpchess/

MPchess consists of 7 files METAPOST :

- mpchess.mp;
- mpchess-chessboard.mp;
- mpchess-pgn.mp;
- mpchess-fen.mp;
- mpchess-cburnett.mp;

- `mpchess-pieces.mp`;
- `mpchess-skak.mp`.

Once this is done, `MPchess` will be loaded with the classic `METAPOST` input code

```
input mpchess
```

1.2 With MikTeX and Windows

These two systems are unknown to the author of `MPchess`, so we refer you to the MikTeX documentation concerning the addition of local packages:

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

1.3 Dependencies

`MPchess` depends, of course on `METAPOST`, as well as the packages `hatching` and—if `MPchess` is not used with Lua \TeX and the `luamplib` package—the `latexmp` package.

2 Package Rationale and General Philosophy

There are already \TeX packages for drawing chess boards and positions, including the very good package `xskak` [2] coupled with the package `chessboard` [1]. There, Ulrike Fisher made improvements, undertaken maintenance work, and provided us with excellent tools to make chess diagrams and to handle the different formats of game descriptions¹. The documentation of each of these packages is very good.

Several things motivated the creation of `MPchess`. First of all, with `chessboard` the addition of a set of pieces is not very easy, because it relies on fonts. Moreover, the author finds that drawing chess game diagrams is by its nature a very graphical task, and that using a dedicated drawing language offers increased flexibility. In that case, what better than `METAPOST` [6]?

With `MPchess`, the final image of the chess board is built with the pieces by successive layers. Thus, we begin by producing and drawing the board (`backboard`), which we can modify—for example, by coloring some squares. We then can add the pieces of the position (`chessboard`), and finally, we can annotate the whole thing with marks, colors, arrows, and so forth.

Moreover, `MPchess` produces images that are graphically close to what is provided by excellent *open source* website <https://lichess.org>. The colors, the pieces and the general aspect of `MPchess` are largely inspired by what this website offers.

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

3 Board

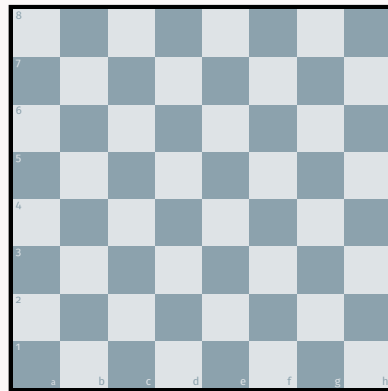
The board is called with MPchess `backboard`. You have to initialize the board before drawing it. This is done with the following command:

`init_backboard`

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

Example 1

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



This initialization will make it possible to use the various options and features that are described in the following sections.

3.1 Size of the Board

When creating the `backboard`, you can further specify its width. 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:

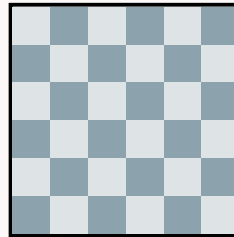
`set_backboard_size(<nbr>)`

<nbr>: is the desired number of squares. The board will then be square of size $\langle nbr \rangle \times \langle nbr \rangle$. 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 specify size of the game board, you can use 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, `MPchess` 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 Color Themes

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

`set_color_theme(<string>)`

<string> can be:

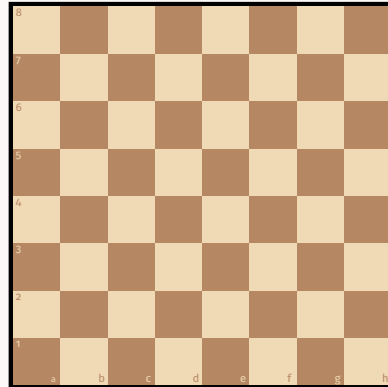
- `"BlueLichess"` (thème par défaut);

- "BrownLichess";
- or "Classical".

The following examples show the results obtained from each theme:

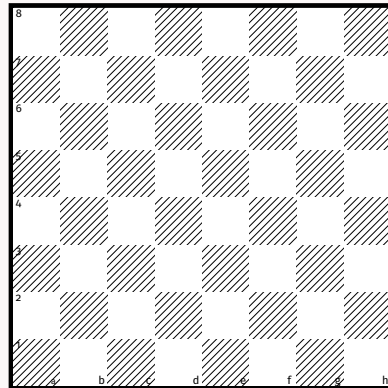
Example 3

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



Example 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 Configuring a Personal Color Theme

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

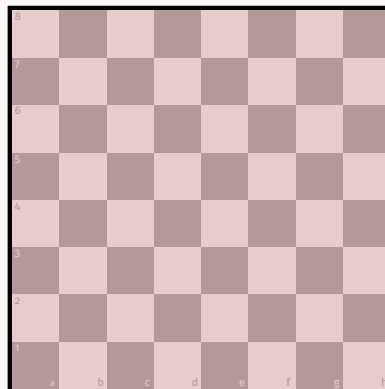
```
set_white_color(<color>)
```

```
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.

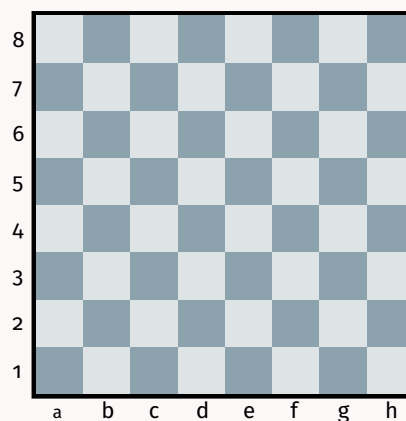
MPchess allows you to position these coordinates outside the board with the following command:

`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 the previous examples that with `luamplib` and `TeX`, the font used for the coordinates is the font of the current document. To draw these letters and these numbers, MPchess uses the METAPOST operator `infont` and the font is set to `defaultfont` by default², so it is not possible to modify the

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

composition font of the coordinates. This font can be changed with the following command.

```
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 METAPOST 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, *MPchess* 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 chess-board. This is done with the following commands:

```
set_white_player(<string>)
```

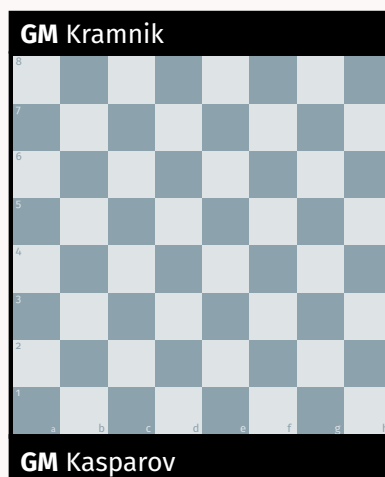
```
set_black_player(<string>)
```

<string>: is the string interpreted by \TeX to display.

Exemple 7

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

init_backboard;
draw backboard;
endfig;
```

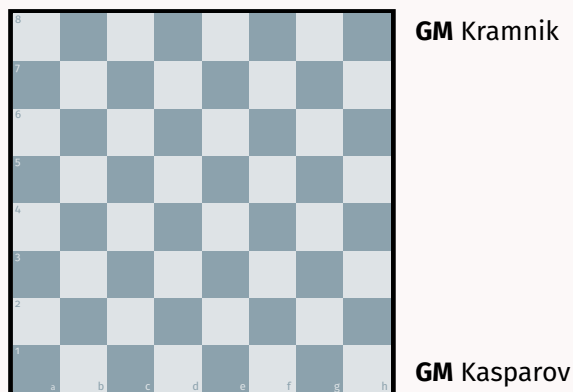


It is possible to place the names on the right side of the board without the black bands present by default. This happens either if the coordinates are printed outside the tray, or if the following command is used:

`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;
```



4 Pieces and Positions

`MPchess`, as described above, builds the picture of a chess position layer by layer. This section describes the configuration of pieces and positions.

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

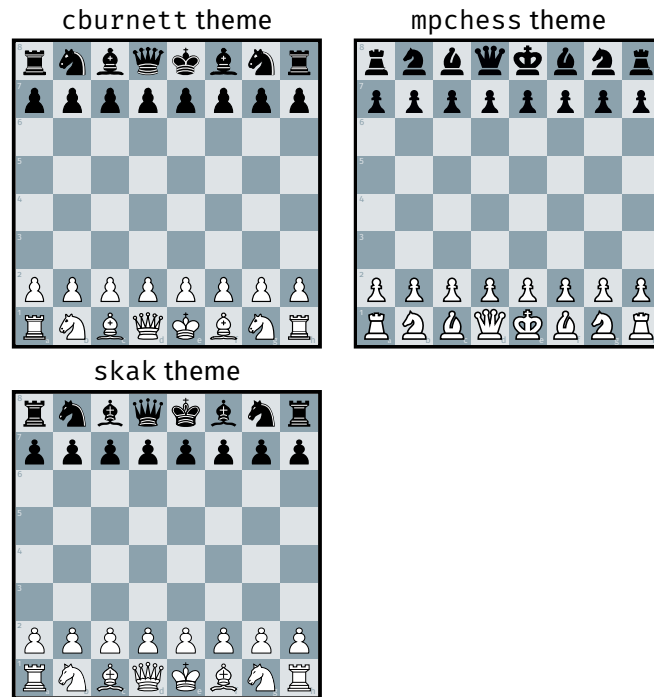


Table 1: The different themes of pieces provided by *MPchess*.

4.1 Setting the Theme of the Pieces

MPchess provides for the moment three themes of pieces: two borrowed from Lichess, and one borrowed from the package *skak* citectan-skak³.

To choose the theme, use the following command:

```
set_pieces_theme(<string>)
```

<string>: can be:

- "**cburnett**", (default), to use the Lichess *cburnett* pieces set;
- "**mpchess**", to use the set specially designed for this package;
- "**skak**", to use the *skak* pieces set.

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

4.2 Specifying to Move

MPchess indicates which player has the current move. This is done by a small colored triangle (white or black) at the end of the board (which you can see in the following examples).

To specify which side is to move, use the following commands:

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

`set_white_to_move`

`set_black_to_move`

By default, white is 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 a 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

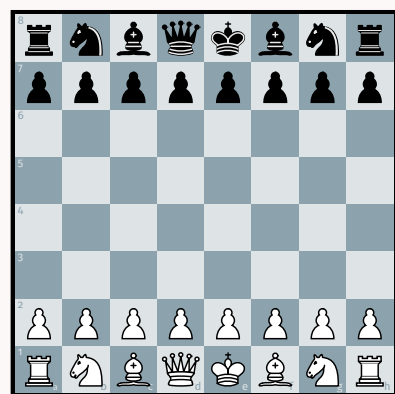
4.4.1 Initialization

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.)
```

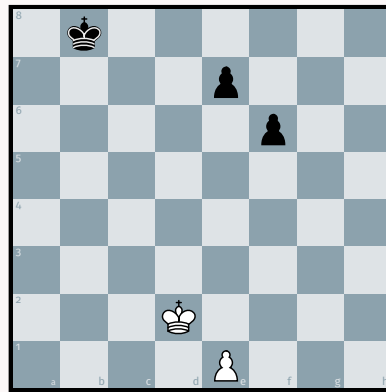
```
add_black_pieces(<piece1>,<piece2>,etc.)
```

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

The following example illustrates the use of these commands:

Example 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 Deleting Pieces

MPchess 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 variables **<square1>**, **<square2>**, and so forth are strings; for example, "a3".

The following command deletes a set of squares in a region determined by two coordinates on the board. This command may take a list of regions:

```
clear_areas(<area1>,<area2>,etc.)
```

The variables **<area1>**, **<area2>**, and so forth are strings consisting of two coordinates separated by a hyphen; for example, "a3-g7".

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

```
clear_files(<file1>,<file2>,etc.)
```

The variables **<file1>**, **<file2>**, and so forth 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 may take a list of ranks.

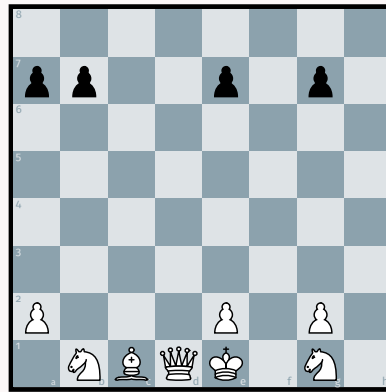
`clear_ranks(<rank1>,<rank2>,etc.)`

The variables **<rank1>**, **<rank2>**, and so forth are strings made up of a number; for example, "4".

The use of all these commands is illustrated in the following example:

Example 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

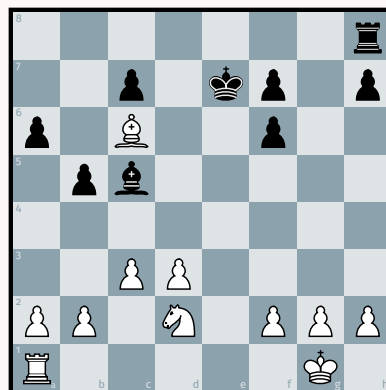
MPchess allows you to read a position in the FEN format thanks to the following command:

`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.

Example 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;
```



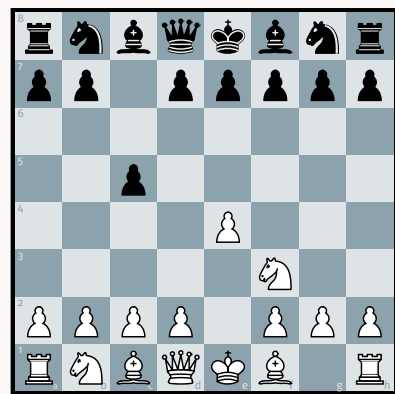
It is also possible to read an external file containing on its first line a string in FEN format with the following command:

`build_chessboard_from_fen_file(<string>)`

<string>: is a string of characters (between double quotes) indicating the name of the file to be read.

Example 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

MPchess also reads strings in the PGN format. Please note, this is a partial implementation of the format—in particular, MPchess does not manage the *tags* of the PGN format. Rather, MPchess handles only the string describing the moves played. In the same way, the accepted format by MPchess does not allow variants or comments.

When such a functionality is used, MPchess stores all the intermediate positions and thus represents them.

To construct the positions, use the following command:

`build_chessboards_from_pgn(<string>)`

Once the positions are built, they can be represented with the following command:

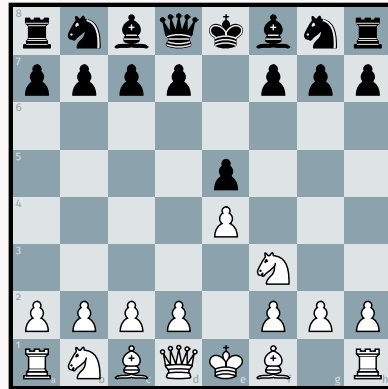
`chessboard_step(<int>)`

<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;
```



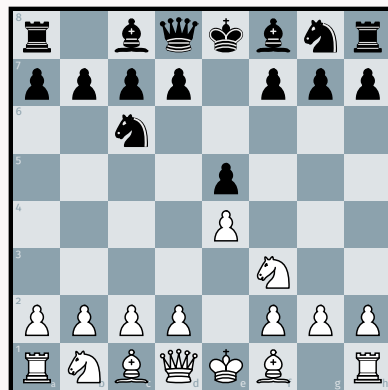
It is also possible to read an external file containing on its first line a string in PGN format with the following command:

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); % Nc
    6
endfig;
```



4.6.1 Showing the Last Move

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

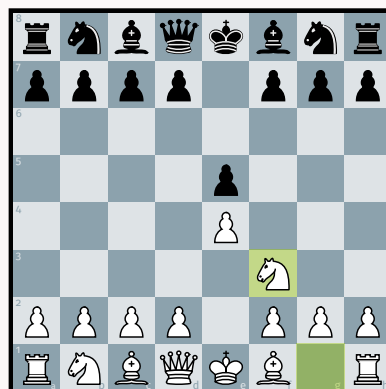
show_last_move(*<int>*)

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

This command uses transparent colors to show the two squares where the last move started and ended. Thus, it must be used between the drawing of the board (**draw backboard**) and the drawing of the pieces (**draw chessboard_step(i)**).

Example 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 Getting 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 you 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`.

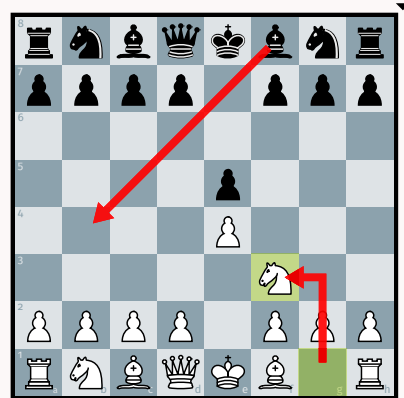
<string1>: is a string (between double-quotes) consisting of two coordinates (letter and number) separated by two characters that can be

- 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:

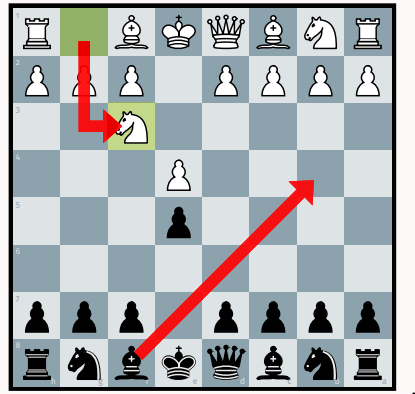
set_arrow_width(<coeff>)

<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.)`

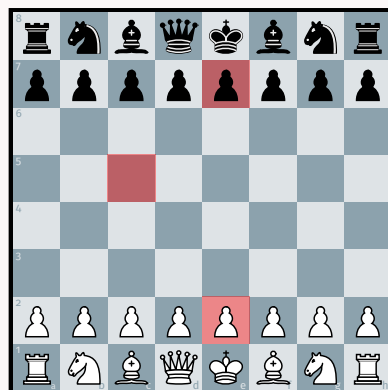
<color>: is a METAPOST `color`.

<coord1>: is a string (between double quotes) consisting of two coordinates (a letter and a 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 transparently colors the specified squares.

5.3 Circles

MPchess allows you to surround squares with circles using the following command below:

`draw_circles(<color>)(<coord1>,<coord2>, etc.)`

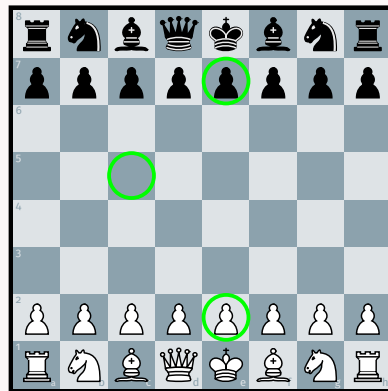
<color>: is a METAPOST `color`.

<coord1>: is a string (between double quotes) consisting of two coordinates (letter and number).

The following example illustrates the use of this command.

Example 20

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



5.4 Crosses

MPchess allows you to draw crosses on squares with the following command:

`draw_crosses(<color>)(<coord1>,<coord2>, etc.)`

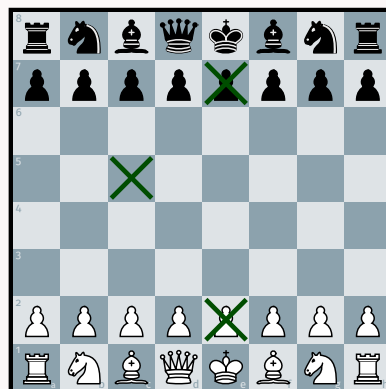
<color>: is a METAPOST `color`.

<coord1>: is a string (between double quotes) consisting of two coordinates (a letter and a number).

The following example illustrates the use of this command.

Example 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

MPchess allows you to display the classic move comments of the type “!?” with the following command:

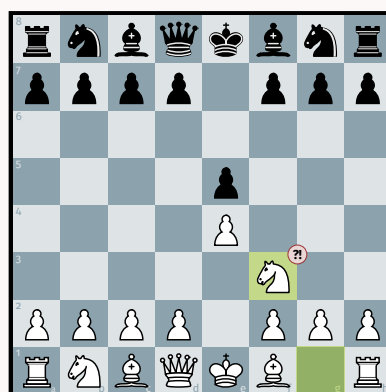
`draw_comment(<str>,<pos>)`

<str>: is a string (between double quotes) to display.

<pos>: is a string (between double quotes) consisting of a coordinate (a letter and a number).

Example 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:

`set_comment_color(<color>)`

5.6 Main Lines

MPchess provides a command to display the arrows of the moves of the main lines of analysis. There are commands for both colors.

```
draw_white_main_lines(<move1>,<move2>,etc.)
```

```
draw_black_main_lines(<move1>,<move2>,etc.)
```

<move1>, <move2>, etc.: are the moves to be illustrated using PGN notation.

When using the format PGN for the construction of the positions to be displayed, the following commands can be used to specify which move of the game is being commented on:

```
draw_white_main_lines_step(<step>)(<move1>,<move2>,etc.)
```

```
draw_black_main_lines_step(<step>)(<move1>,<move2>,etc.)
```

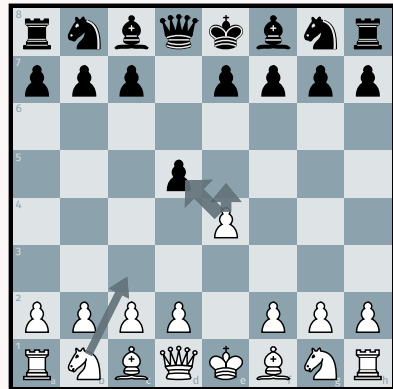
<step>: is the step of the game you want to annotate;

<move1>, <move2>, etc.: are the moves to be illustrated using PGN notation.

The following example illustrates the use of this command:

Example 23

```
input mpchess
string pgnstr;
pgnstr:="1. e4 d5";
build_chessboards_from_pgn(
    pgnstr);
beginfig(0);
init_backboard;
draw backboard;
draw chessboard_step(2);
draw_white_main_lines_step(2)
    ("exd5", "e5", "Nc3");
endfig;
```



To change the color (by default 0.3[_blackColorSquare,black]), use the following command:

```
set_main_lines_color(<color>)
```

5.7 Possible Moves

MPchess allows, by imitating <https://lichess.org>, to show possible moves for a piece. To do that, use the following command:

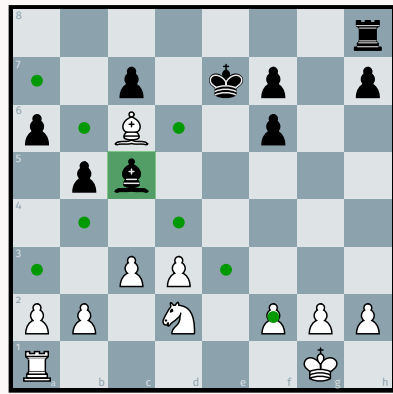
`show_possible_moves(<square>)`

<square>: is a string (between double quotes) consisting of a coordinate (a letter and a number).

This command must be used after having drawn the **chessboard**.

Exemple 24

```
input mpchess
string fenstr;
fenstr:="7r/2p1kp1p/p1B2p
2/1pb5/8/2PP4/PP1N1
PPP/R5K1 b - - 2 19";
build_chessboard_from_fen(
fenstr);
beginfig(0);
init_backboard;
draw backboard;
draw chessboard;
show_possible_moves("c5");
endfig;
```



In the case where several positions have been constructed with PGN format, the following command should be used to access the chosen position.

`show_possible_moves_step(<step>)(<square>)`

<step>: is the step of the game you want to annotate;

<square>: is a string (between double quotes) consisting of a coordinate (a letter and a number).

By default, the color is set to 0.4[green,black], but this can be changed with the can be changed with the following command:

`set_possible_moves_color(<color>)`

6 Miscellaneous

6.1 Reset the chessboard

To reset the internal structure storing the positions of the parts, use the following command:

`clear_chessboard`

6.2 Global Reset

To reinitialize the values of the different parameters of `MPchess`, use the following command:

```
reset_mpchess
```

6.3 Clip the board

We can clip the chessboard with the following command.

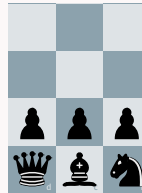
```
clip_chessboard(<string>)
```

<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 25

```
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 pdf \LaTeX or X \LaTeX

There are several ways to include the images produced by `MPchess` in a \LaTeX document. 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 pdf \LaTeX or X \LaTeX ⁴.

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

7.1.1 With **mpgraphics**

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

Here is a complete example:

```
\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{document}
```

7.1.2 With **gmp**

The use of the package **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 **MPchess** 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:

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

\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 Lua \TeX and **luamplib**

It is possible to use *MPchess* directly in a \TeX file with Lua \TeX and the package **luamplib**. (This is what is done to write this documentation.)

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

For certain functionalities, *MPchess* 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

```
\mplibtexttextlabel{enable}
```

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

We can load globally *MPchess* with the following command:

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

Here is a complete example (to be compiled with Lua \TeX).

```
\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 *MPchess*. Among these, we can think of:

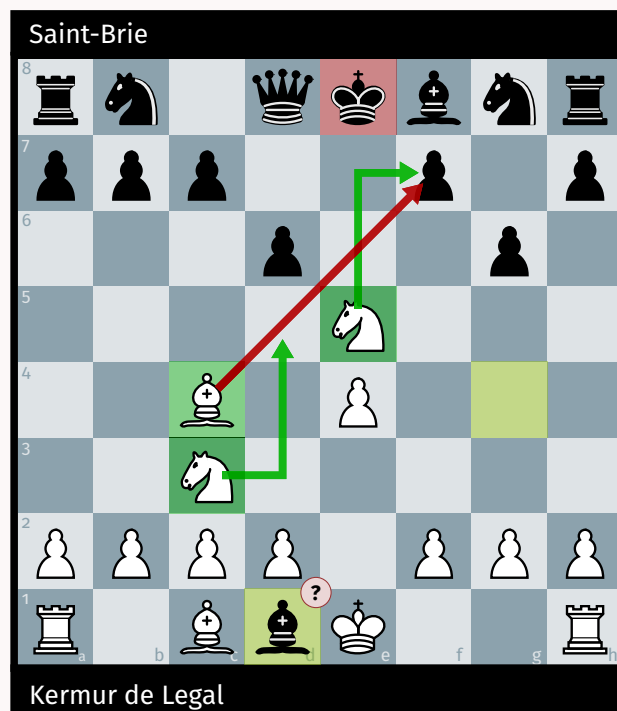
- display the captured pieces, or the differential of the captured pieces;
- display the remaining time for each player;
- parsing the complete PGN format with the optional tags ;

- adding the coordinates outside the board when it is clipped;
- add piece themes.

9 Complete Example

Exemple 26

```
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;
```



10 History

vo.3, April 6, 2023: Corrections in the documentation, especially the English version; added commands to visualize the possible movements for a piece (section 5.7).

vo.2, March 28, 2023: Added commands for PGN and FEN file reading; added commands for displaying the main lines of analysis; removal of the staunty theme (because of license) and creation of the mpchess parts theme.

vo.1, March 23, 2023: First publication on the CTAN.

11 Acknowledgements

The author would like to thank Quark67 for his feedback and corrections, and Douglas Johnson for correcting the English version of the documentation. This feedback and encouragement is extremely appreciated!

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

add_black_pieces, 13
add_white_pieces, 13

build_chessboard_from_fen, 14
build_chessboard_from_fen_file, 15
build_chessboard_from_pgn_file, 16
build_chessboards_from_pgn, 15

chessboard, 12
chessboard_step, 15
clear_areas, 13
clear_chessboard, 23
clear_files, 13
clear_ranks, 14
clear_squares, 13
clip_chessboard, 24
color_square, 19

draw_arrows, 17
draw_black_main_lines, 22
draw_black_main_lines_step, 22
draw_circles, 20
draw_comment, 21
draw_crosses, 20
draw_white_main_lines, 22
draw_white_main_lines_step, 22

get_backboard_size, 6
get_backboard_width, 5
get_halfmove_number, 17
get_square_dim, 6

get_totalmove_number, 17

init_backboard, 5
init_chessboard, 12

reset_mpchess, 24

set_arrow_width, 18
set_backboard_size, 6
set_backboard_width, 5
set_black_color, 7
set_black_player, 9
set_black_to_move, 12
set_black_view, 9
set_color_theme, 6
set_comment_color, 21
set_coords, 9
set_coords_font, 9
set_coords_inside, 8
set_coords_outside, 8
set_empty_chessboard, 12
set_last_move_color, 17
set_main_lines_color, 22
set_no_coords, 9
set_pieces_theme, 11
set_players_side, 10
set_possible_moves_color, 23
set_white_color, 7
set_white_player, 9
set_white_to_move, 12
set_white_view, 9
set_whos_to_move, 12
show_last_move, 16
set_possible_moves, 23
set_possible_moves_step, 23

unset_whos_to_move, 12