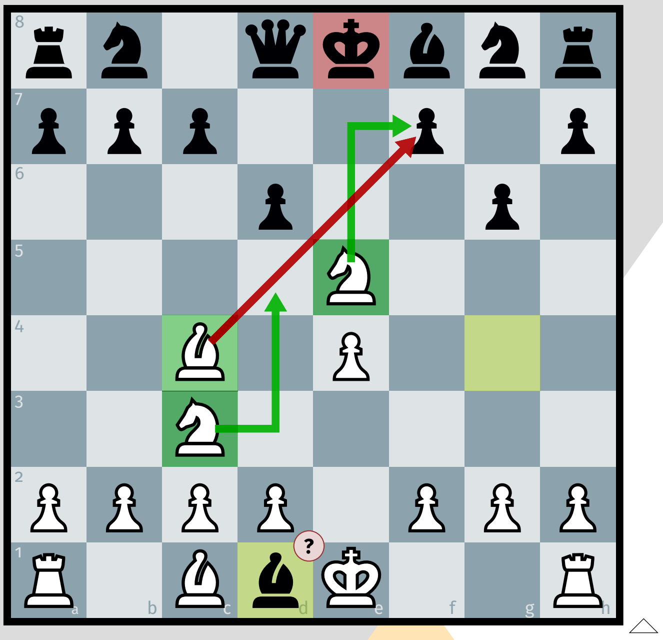
MPchess

drawing chess boards and positions with METAPOST



Contributor
Maxime Chupin
notezik@gmail.com

Version 0.9, 2024, Decembre, 16th
https://gitlab.gutenberg-asso.fr/mchupin/mpchess

Abstract

The MP*chess* 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`. Relying on METAPOST probably allows more graphic flexibility than the excellent LaTeX chess packages that already exist.

`https://plmlab.math.cnrs.fr/mchupin/mpchess`
`https://github.com/chupinmaxime/mpchess`

Contents

1 Installation . . . **3**
1.1 With TeXlive under Linux or macOS . . . 3
1.2 With MikTeX 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 . . . 9
3.5 Display Coordinates . . . 10
3.6 White or Black Side . . . 11
3.7 Players' Names . . . 11

4 Pieces and Positions . . . **12**
4.1 Setting the Theme of the Pieces . . . 13
4.2 Specifying to Move . . . 14
4.3 Draw a Position . . . 15
4.4 Build a position . . . 15
4.4.1 Initialization . . . 15
4.4.2 Adding Pieces . . . 16
4.4.3 Deleting Pieces . . . 16
4.5 Reading Data in FEN Format . . . 17
4.6 Reading Data in PGN Format . . . 18
4.6.1 Showing the Last Move . . . 20
4.6.2 Getting the Number of Moves . . . 20

5 Annotation . . . **21**
5.1 Arrows . . . 21
5.2 Coloring Squares . . . 22
5.3 Circles . . . 23
5.4 Crosses . . . 23
5.5 Move Comments . . . 24

5.6 Main Lines 25
5.7 Possible Moves 26

6 Miscellaneous 26
6.1 Reset the chessboard 26
6.2 Global Reset 27
6.3 Clip the board 27

7 Use with LaTeX 27
7.1 Use with pdfLaTeX or XeLaTeX 27
7.1.1 With mpgraphics 28
7.1.2 With gmp 28
7.2 Use with LuaLaTeX and luamplib 29
7.3 TrueType Font 29

8 To Do 30

9 Complete Example 31

10 History 32

11 Acknowledgements 32

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

1 Installation

MP*chess* is on CTAN and can also 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/metapost/mpchess/`

MP*chess* 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, MP*chess* will be loaded with the classic METAPOSTinput code

```
input mpchess
```

1.2 With MikTEX and Windows

These two systems are unknown to the author of MP*chess*, so we refer you to the MikTEXdocumentation concerning the addition of local packages:

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

1.3 Dependencies

MP*chess* depends, of course on METAPOST, as well as the packages hatching and—if MP*chess* is not used with LuaLATEX and the luamplib package—the latexmp package.

2 Package Rationale and General Philosophy

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

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

Moreover, MP*chess* 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 MP*chess* are largely inspired by 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:

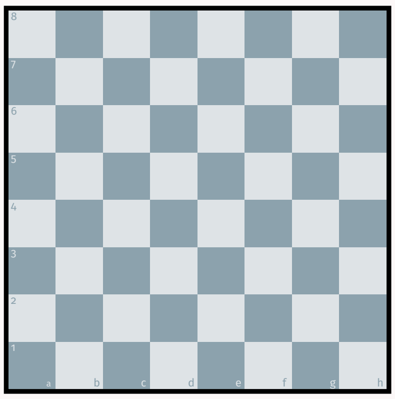
`init_backboard`

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

Exemple 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⟩)`

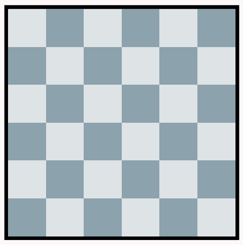
⟨nbr⟩: is the desired number of squares. The board will then be square of size ⟨*nbr*⟩×⟨*nbr*⟩. By default this number is 8.

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 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, 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 Color Themes

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

`set_color_theme(⟨string⟩)`

⟨string⟩ can be:

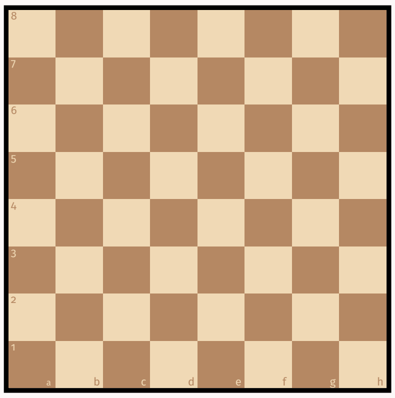
- `"BlueLichess"` (default);

- "BrownLichess";
- "GreenLichess";
- "PinkPyramidalLichess";
- "Wood";
- "Classical";
- "Dotted";
- or "User";

The following example shows the results obtained from "BrownLichess":

Exemple 3

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



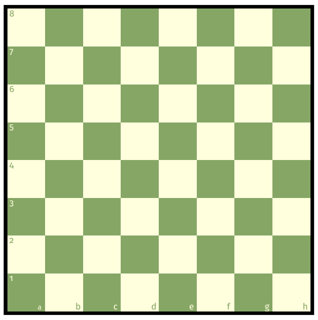
The table 1 shows the different results of the different themes.

The use of "User" is a special case, as it allows you to define black and white squares. For this to work a another macro buildUserSquare should be difined. It must build the images (picture) _blackSquarePic and _whiteSquarePic. Here is a prototype of such a command.

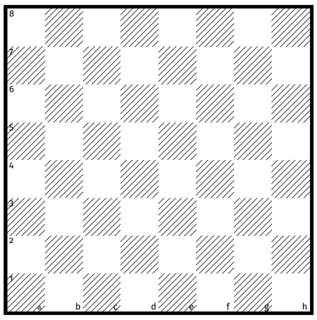
```
def buildUserSquare(expr _SquareUnit)=
  _blackSquarePic:=image(
      fill (unitsquare scaled _SquareUnit) withcolor
           red;
  );
  _whiteSquarePic:=image(
      fill unitsquare scaled _SquareUnit withcolor
           green;
  );
enddef;
```

The macro takes a unit as argument, and must construct two squares with length equal to this unit.

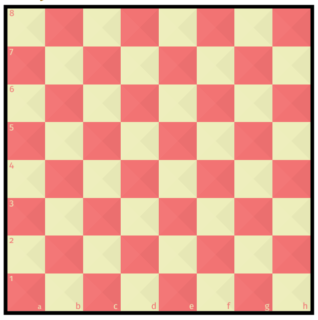
`"GreenLichess"` theme



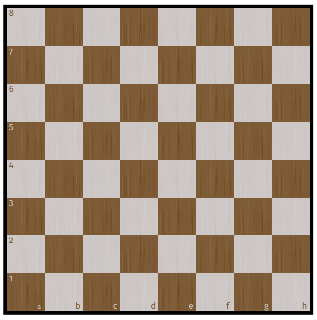
`"Classical"` theme



`"PinkPyramidalLichess"` theme



`"Wood"` theme



`"Dotted"` theme

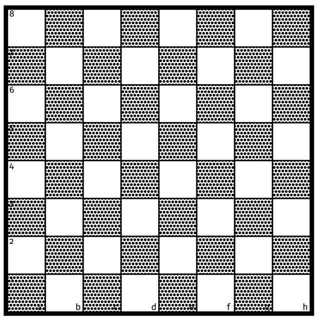


Table 1: The different color themes provided by MP*chess*.

3.4.2 Configuring a Personal Color Theme

Colors. A color theme is the definition of two colors. These can be defined with the following commands[2]:

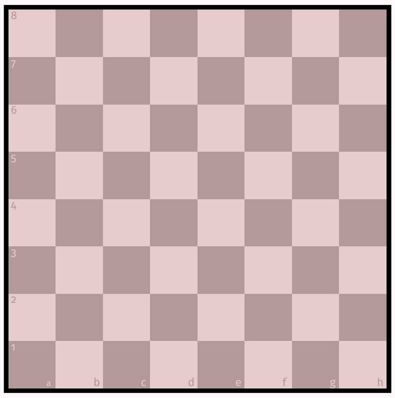
`set_white_squares_color(⟨color⟩)`

`set_black_squares_color(⟨color⟩)`

⟨*color*⟩ is a METAPOST `color`.

Exemple 4

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



Color Types. To choose a type, you can use the following command:

`set_board_type(⟨string⟩)`

Three types of coloring are available:

⟨***string***⟩ can be:

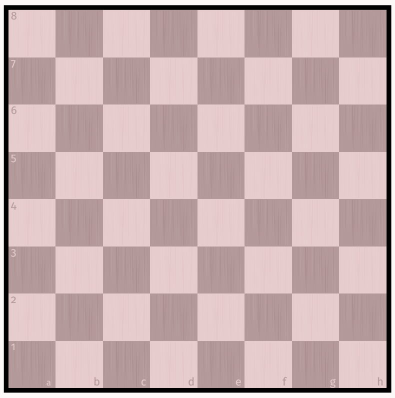
- `"flat"`, simple flat coloring (default);
- `"pyramidal"`, Lichess *pyramidal* coloring;
- `"wood"`, wood imitation.

Here is an example coupling color and type definitions.

[2]Attention, in version 0.6, `set_white_color` became `set_white_squares_color` and `set_black_color` became `set_black_squares_color`.

Exemple 5

```
input mpchess
beginfig(0);
set_white_squares_color
      ((0.9,0.8,0.8));
set_black_squares_color
      ((0.7,0.6,0.6));
set_board_type("wood");
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 you to choose the coordinates position inside or outside the board with the following command[3]:

```
set_coordinates_position(⟨string⟩)
```

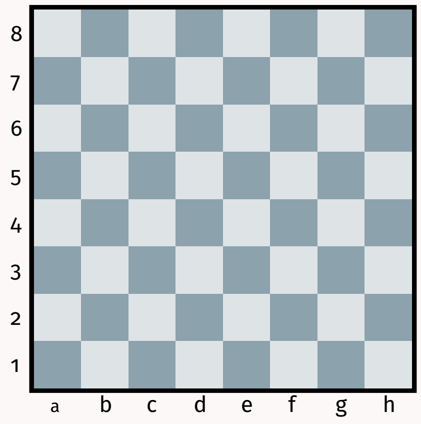
⟨***string***⟩ can be:

- `"inside"` (default);
- `"outside"`.

The result is as follows.

Exemple 6

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



[3]Attention, in version 0.6, `set_coords_inside` and `set_coords_outside` became `set_coordinates_position`.

You can see in the previous examples that with luamplib and LaTeX, the font used for the coordinates 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[4], so it is not possible to modify the composition font of the coordinates. This font can be changed with the following command[5].

`set_coordinates_font(⟨font⟩)`

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

You can also delete the coordinates with the following command[6]:

`hide_coordinates`

And the reverse command also exists:

`show_coordinates`

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 LaTeX to display.

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

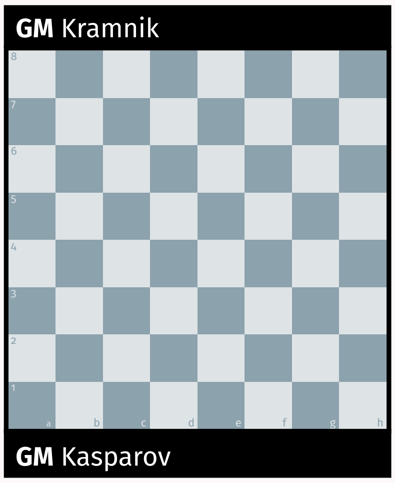
[5]Attention, in version 0.6, `set_coords_font` became `set_coordinates_font`.

[6]Attention, in version 0.6, `set_no_coords` became `hide_coordinates` and `set_coords` became `show_coordinates`.

Exemple 7

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

init_backboard;
draw backboard;
endfig;
```

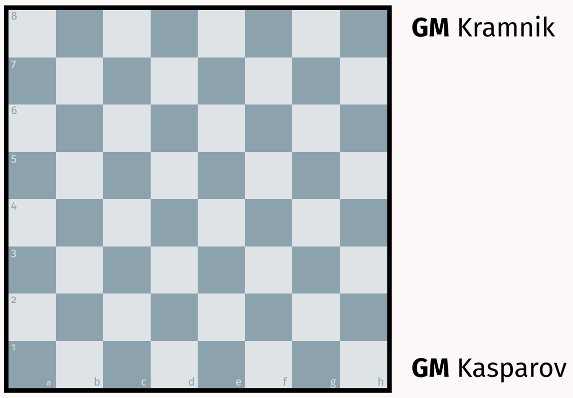


It is possible to place the names on the right side of the board without the black bands present by default. This happens either if the coordinates are printed outside the tray, or if the following command is used:

```
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;
```

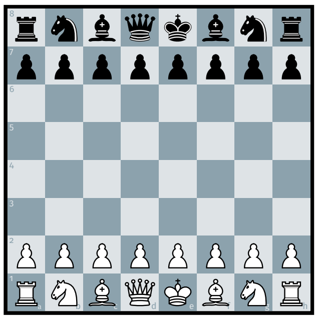


4 Pieces and Positions

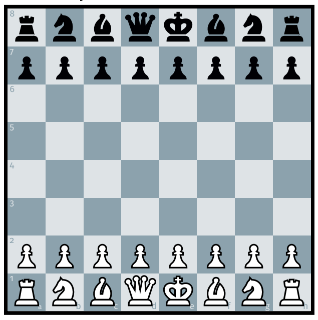
MPchess, as described above, builds the picture of a chess position layer by layer. This section describes the configuration of pieces and positions.

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

cburnett theme



mpchess theme



skak theme

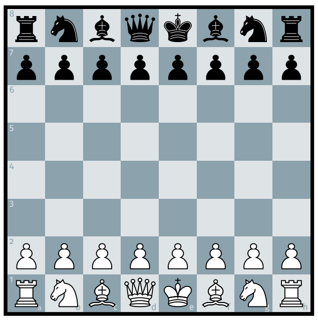


Table 2: The different themes of pieces provided by MP*chess*.

4.1 Setting the Theme of the Pieces

MP*chess* provides for the moment three themes of pieces. The default theme is called `mpchess`. It has been designed for this package METAPOST. It has been proposed to the Lichess project, and has been accepted. Thus, you will also have access to the `mpchess` piece set with Lichess[7]

Another theme is borrowed from Lichess (`cburnett`) and the last one is borrowed from the package skak citectan-skak[8].

To choose the theme, use the following command:

```
set_pieces_theme(⟨string⟩)
```

⟨*string*⟩: can be:

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

The table 2 shows the result of the three sets of pieces.

[7]The open-source projects feed each other! Even if obviously, this package has borrowed much more from Lichess than the other way around.

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

4.2 Specifying to Move

MP*chess* 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:

`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[9]:

`show_whos_to_move`

`hide_whos_to_move`

It is now possible to change the appearance of the marker (since version 0.9[10]. For this purpose, MP*chess* provides three styles: `"triangle"` (default), `"square"` and `"disk"`. To select the marker type, use the following command:

`set_marker_type(⟨string⟩)`

⟨*string*⟩ may be

- `"triangle"` (default value);
- `"square"`;
- `"disk"`;
- `"custom"`.

You can adjust the size (by a factor of the unit dimension) with the following command:

`set_marker_scale(⟨numeric⟩)`

⟨*numeric*⟩: is a unit dimension factor for enlarging or shrink the mark (default value is 0.35).

The `"custom"` marker type value lets you define the marker's appearance. When this option is called, a macro METAPOST `custom_maker(expr s,r)` must be defined. The `s` parameter is the scale factor setting: the markers supplied by MP*chess* are drawings contained in a square of side 1 (METAPOST unit) and scaled by `s*_chessSquareU`. The `r` parameter is a boolean that handles the case where the marker is rotated 180 degrees when placed at the top of the chessboard.

For example, a `custom_marker` macro reproducing the triangle symbol might be as follows:

[9]Attention, in version 0.6, `set_whos_to_move` and `unset_whos_to_move` became `show_whos_to_move` and `hide_whos_to_move`.

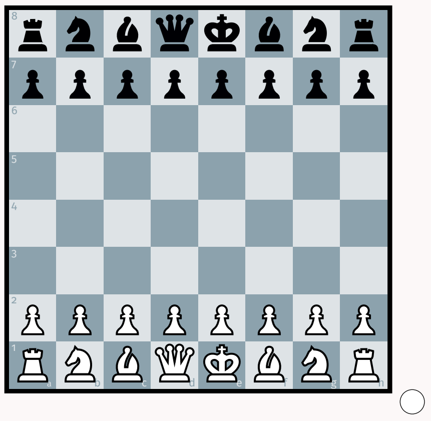
[10]Thanks to Udi Fogiel for fixing a placement bug and for suggesting improvements.

```
def custom_marker(expr s, r)=
  ((0,0)--(1,0)--(0.5,0.5)--cycle) if r: rotatedaround
       ((0.5, 0.5/3), 180) fi scaled (s*_chessSquareU)
enddef;
```

Only the outline is defined, which will be filled by either white or black.

Exemple 9

```
input mpchess
beginfig(0);
set_marker_type("disk");
set_marker_scale(0.5);
init_backboard;
draw backboard;
init_chessboard;
draw chessboard;
endfig;
```



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

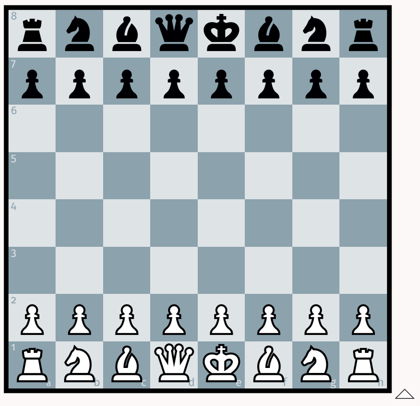
4.4.1 Initialization

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

`init_chessboard`

Exemple 10

```
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.)

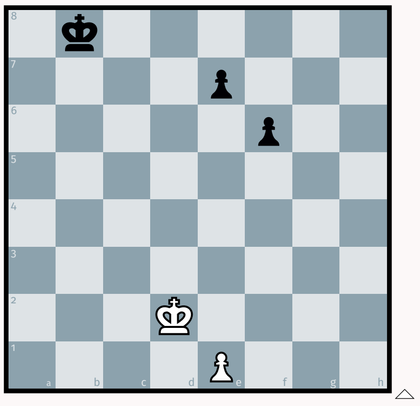
add_black_pieces(⟨*piece1*⟩,⟨*piece2*⟩,etc.)

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:

Exemple 11

```
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

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 variables ⟨***square1***⟩, ⟨***square2***⟩, and so forth are strings; for example, `"a 3"`.

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.

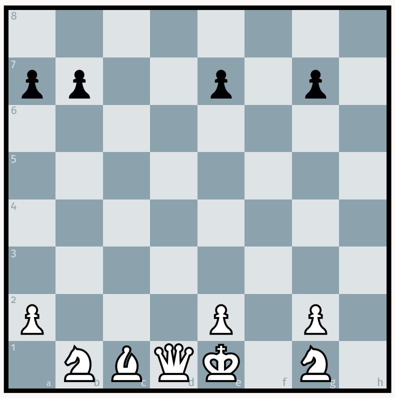
`clear_ranks(⟨rank1⟩,⟨rank2⟩,etc.)`

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:

Exemple 12

```
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

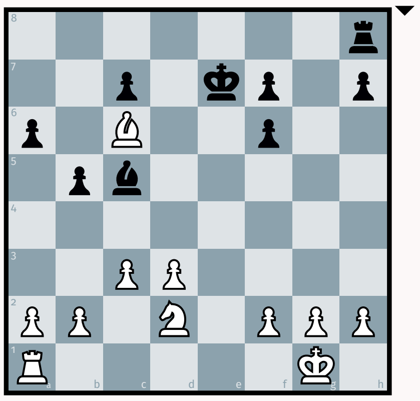
MP*chess* allows you to read a position in the FEN format thanks to the following command:

`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 13

```
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;
```



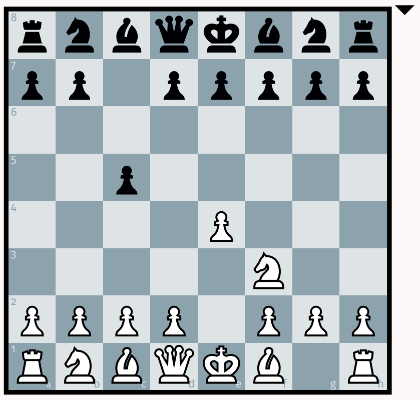
It is also possible to read an external file containing on its first line a string in FEN format with the following command:

`build_chessboard_from_fen_file(⟨string⟩)`

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

Exemple 14

```
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 reads strings in the PGN format. Please note, this is a partial implementation of the format—-in particular, MP*chess* does not manage the *tags* of the PGN format. Rather, MP*chess* handles only 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 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:

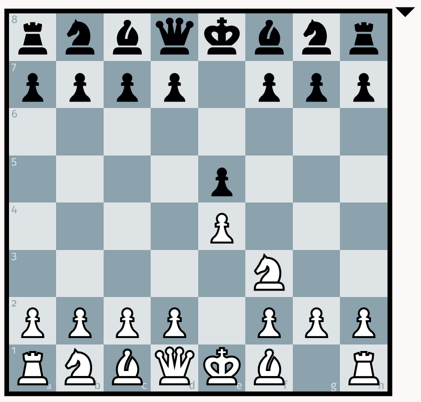
`chessboard_step(⟨int⟩)`

⟨*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 15), returns a `picture`.
The following example illustrates the use of these commands:

Exemple 15

```
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;
```



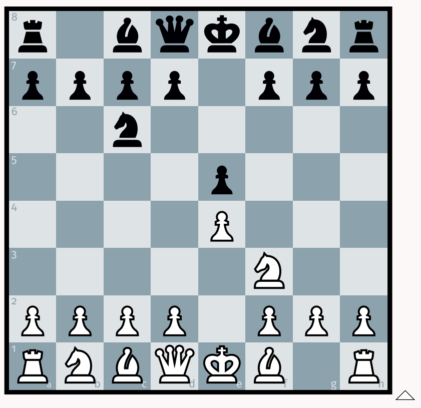
It is also possible to read an external file containing on its first line a string in PGN format with the following command:

`build_chessboard_from_pgn_file(⟨string⟩)`

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

Exemple 16

```
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:

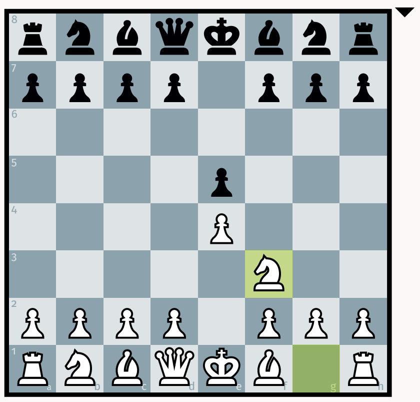
`show_last_move(⟨int⟩)`

⟨*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)`).

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
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`.

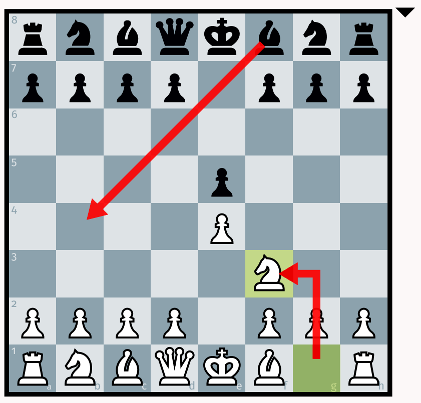
⟨string1⟩: is a string (between double-quotes) consisting of two coordinates (letter and number) separated by two characters that can be

- `--` 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 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);
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:

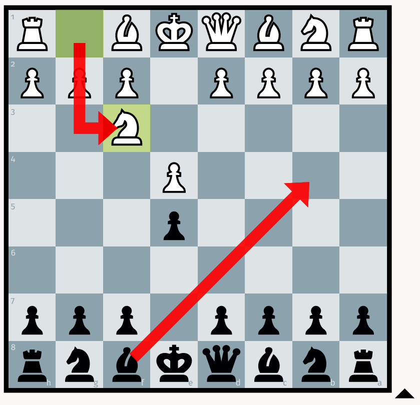
`set_arrow_width(⟨coeff⟩)`

⟨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 19

```
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

MP*chess* also allows you to color squares with the following command:

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

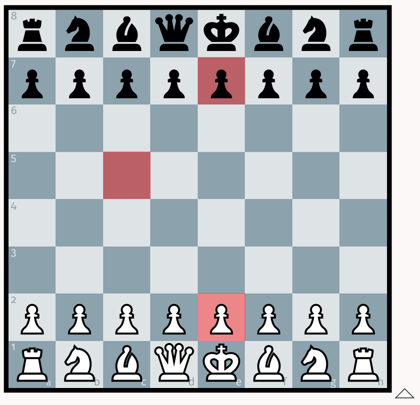
⟨*color*⟩: is a METAPOST `color`.

⟨*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 20

```
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

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

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

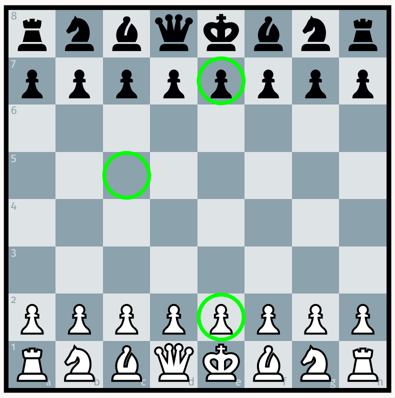
⟨*color*⟩: is a METAPOST `color`.

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

The following example illustrates the use of this command.

Exemple 21

```
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.)`

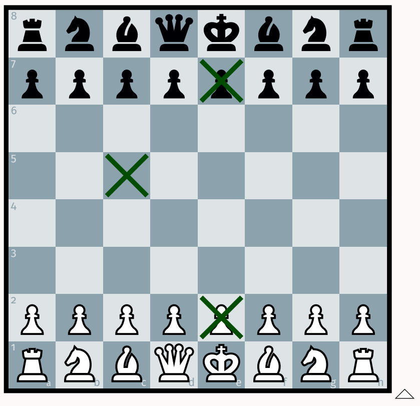
⟨*color*⟩: is a METAPOST `color`.

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

The following example illustrates the use of this command.

Exemple 22

```
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:

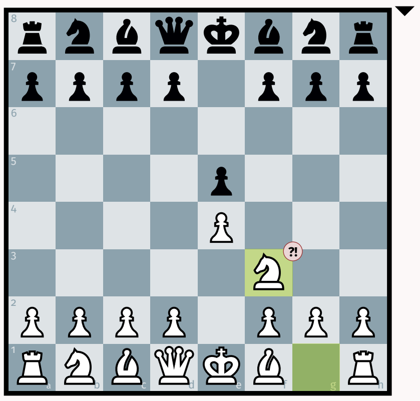
```
draw_comment(⟨str⟩,⟨pos⟩)
```

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

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

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);
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

MP*chess* 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 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.)`

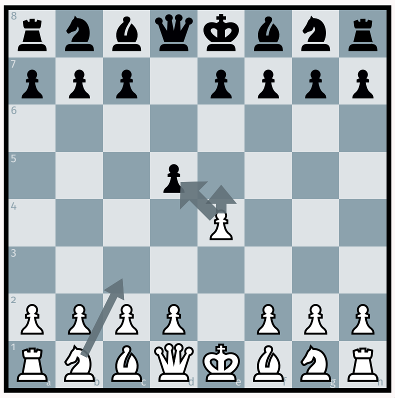
⟨*step*⟩: is the step of the game you want to annote;

⟨*move1*⟩, ⟨*move2*⟩, etc.: are the moves to be illustrated using PGN notation.

The following example illustrates the use of this command:

Exemple 24

```
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

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

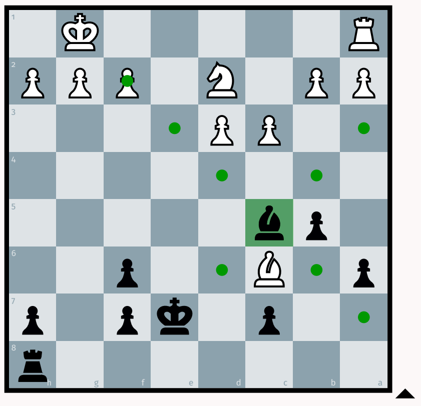
show_possible_moves(⟨*square*⟩)

⟨***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 25

```
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);
  set_black_view;
  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 annote;

⟨***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 MP*chess*, use the following command:

```
reset_mpchess
```

6.3 Clip the board

We can clip the chessboard with the following command.

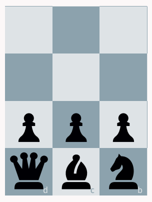
```
clip_chessboard(⟨string⟩)
```

⟨*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 26

```
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[11].

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

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; 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 LaTeXdocument.

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 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:

```
\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 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 then suffices 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, refer to the +luamplib [4] documentation.

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

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

Here is a complete example (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}
```

7.3 TrueType Font

The MP*chess* package provides a very simple font composed of only the 12 glyphs corresponding to the black and white pieces in the Unicode table. To access it you just have to use LuaLaTeX or XeLaTeX and the package fontspec. To facilitate its use, we propose some definitions.

```
  \documentclass{article}
  \usepackage{fontspec}
  \newfontfamily{\chessfont}{mpchess font}
  \newcommand\bP{{\chessfont \char"265F}} % black Pawn
  \newcommand\bN{{\chessfont \char"265E}} % black Knight
  \newcommand\bB{{\chessfont \char"265D}} % black Bishop
  \newcommand\bR{{\chessfont \char"265C}} % black Rook
  \newcommand\bQ{{\chessfont \char"265B}} % black Queen
  \newcommand\bK{{\chessfont \char"265A}} % black King
  \newcommand\wP{{\chessfont \char"2659}} % white Pawn
  \newcommand\wN{{\chessfont \char"2658}} % white Knight
  \newcommand\wB{{\chessfont \char"2657}} % white Bishop
  \newcommand\wR{{\chessfont \char"2656}} % white Rook
  \newcommand\wQ{{\chessfont \char"2655}} % white Queen
  \newcommand\wK{{\chessfont \char"2654}} % white King

  \begin{document}

 1. e4 e5 2. \wB c4 d6 3. \wN f3 \bB g4 4. \wN
  c3 g6 5. \wN xe5 \bB xd1.

  \end{document}
```

This code will produce:
”1. e4 e5 2. ♗c4 d6 3. ♘f3 ♝g4 4. ♘c3 g6 5. ♘xe5 ♝xd1.”

8 To Do

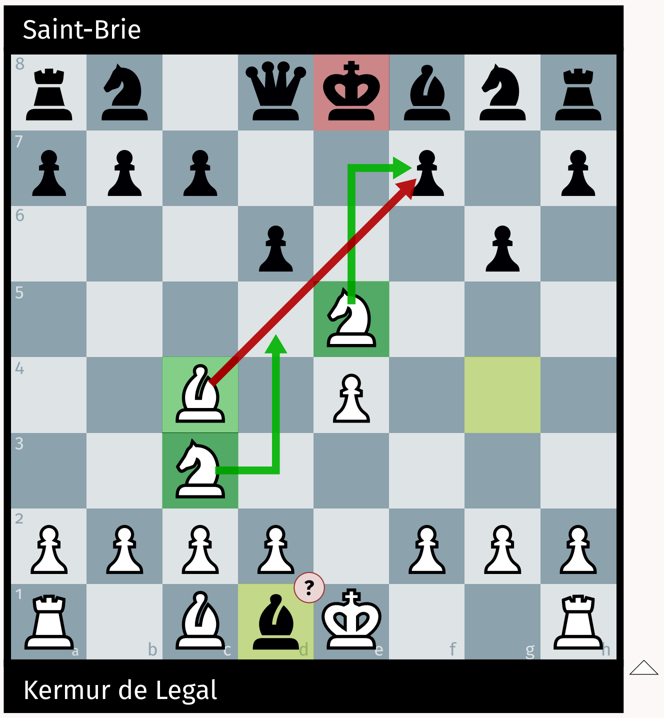
Many things can be added to MP*chess*. Among these, we can think of:

- display the captured pieces, or the differential of the captured pieces;
- 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 27

```
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

v0.9, December 16, 2024: Correction of a bug in the placement of the trait marker (by Udi Fogiel), added different marker features (`"triangle"`, `"square"`,`"disk"` and `"custom"`). Filling of the white marker.

v0.8, October 26, 2024: Adding `"Dotted"` board theme, and `"User"` board theme that allows user to define black and white squares.

v0.7, July, 2023: Black knight adjustment, adding color board themes (`GreenLichess`, `PinkPyramidalLichess`, `Wood` with color type (`set_board_type`)).

v0.6, April 26, 2023: Fixed bugs concerning castling management, and moves ambiguities in PGN format. Changed `set_white_color` to `set_white_squares_color` and `set_black_color` to `set_black_squares_color`. Change `set_no_coords` to `hide_coordinates` and `set_coords` to `show_coordinates`. Change `set_whos_to_move` to `show_whos_to_move` and `unset_whos_to_move` to `hide_whos_to_move`. Change `set_coords_inside` and `set_coords_outside` to `set_coordinates_position`. Change `set_coords_font` to `set_coordinates_font`.

v0.5, April 20, 2023: Bug fixed, **changing the default piece set** for the `mpchess` set (which has been added to Lichess), added TrueType font, and updated documentation.

v0.4, April 6, 2023: Corrections in the documentation, especially the English version; added commands to visualize the possible movements for a piece (section 5.7).

v0.3, March 29, 2023: Small bug.

v0.2, 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.

v0.1, March 23, 2023: First publication on the CTAN.

11 Acknowledgements

The author would like to thank Quark67 for his feedback and corrections, Douglas Johnson for correcting the English version of the documentation and Hans Nieuwenhuis for his advices. 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

_blackSquarePic, 7
_whiteSquarePic, 7

add_black_pieces, 16
add_white_pieces, 16

build_chessboard_from_fen, 17
build_chessboard_from_fen_file, 18
build_chessboard_from_pgn_file, 19
build_chessboards_from_pgn, 19
buildUserSquare, 7

chessboard, 15
chessboard_step, 19
clear_areas, 17
clear_chessboard, 26
clear_files, 17
clear_ranks, 17
clear_squares, 17
clip_chessboard, 27
color_square, 22

draw_arrows, 21
draw_black_main_lines, 25
draw_black_main_lines_step, 25
draw_circles, 23
draw_comment, 24
draw_crosses, 23
draw_white_main_lines, 25
draw_white_main_lines_step, 25

get_backboard_size, 6
get_backboard_width, 5
get_halfmove_number, 20
get_square_dim, 6
get_totalmove_number, 20

hide_coordinates, 11
hide_whos_to_move, 14

init_backboard, 5
init_chessboard, 15

reset_mpchess, 27

set_arrow_width, 21
set_backboard_size, 6
set_backboard_width, 5
set_black_player, 11
set_black_squares_color, 9
set_black_to_move, 14
set_black_view, 11
set_board_type, 9
set_color_theme, 6
set_comment_color, 24
set_coordinates_font, 11
set_coordinates_position, 10
set_empty_chessboard, 16
set_last_move_color, 20
set_main_lines_color, 25
set_marker_scale, 14
set_marker_type, 14
set_pieces_theme, 13
set_players_side, 12
set_possible_moves_color, 26
set_white_player, 11
set_white_squares_color, 9
set_white_to_move, 14
set_white_view, 11
show_coordinates, 11
show_last_move, 20
show_possible_moves, 26
show_possible_moves_step, 26
show_whos_to_move, 14