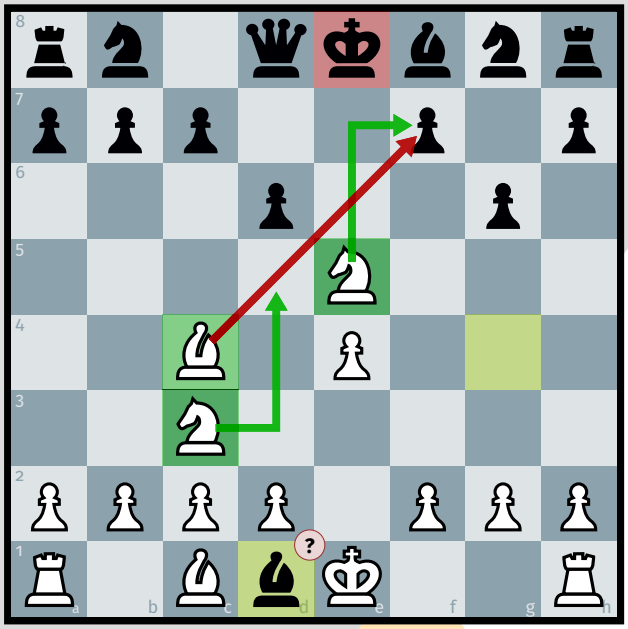
MPchess

dessiner des plateaux d'échecs et des positions avec METAPOST



Contributeur
Maxime CHUPIN
notezik@gmail.com

Version 0.8, 26 octobre 2024
https://plmlab.math.cnrs.fr/mchupin/mpchess

Résumé

Ce package METAPOST permet de dessiner des plateaux d'échecs et des positions. L'apparence des dessins se veut moderne et largement inspiré de ce que propose l'excellent site web Lichess.org. S'appuyer sur METAPOST permet sans doute plus de flexibilité graphique que les excellent packages LaTeX.

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

Table des matières

Ce package est en version beta, n'hésitez pas à faire remonter les bugs, ainsi que les demandes d'amélioration.

1 Installation

MP*chess* est sur le CTAN et peut être installé via le gestionnaire de package de votre distribution.

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

1.1 Avec la TeXlive sous Linux ou macOS

Pour installer MP*chess* avec TeXlive, il vous faudra créer le répertoire `texmf` dans votre home.

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

Ensuite, il faudra y placer les fichiers `.mp` dans le répertoire

`~/texmf/metapost/mpchess/`

MP*chess* est constitué de 7 fichiers METAPOST :

- `mpchess.mp`;
- `mpchess-chessboard.mp`;
- `mpchess-pgn.mp`;

— `mpchess-fen.mp`;

— `mpchess-cburnett.mp`;

— `mpchess-pieces.mp`;

— `mpchess-skak.mp`.

Une fois fait cela, MP*chess* sera chargé avec le classique

```
input mpchess
```

1.2 Avec MikTEX et Windows

Ces deux systèmes sont inconnus de l'auteur de MP*chess*, ainsi, nous renvoyons à leurs documentations pour y ajouter des packages locaux :

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

1.3 Dépendances

MP*chess* dépend des packages METAPOST : hatching et, si MP*chess* n'est pas utilisé avec LuaLATEX et luamplib, latexmp.

2 Pourquoi ce package et philosophie générale

Il existe déjà des packages LATEX pour dessiner des plateaux d'échecs et des positions dont le très bon xskak [2] couplé avec le package chessboard [1]. Ulrike Fisher a réalisé là un travail d'amélioration, de maintient, et nous a fourni d'excellents outils pour réaliser des diagrammes d'échecs et de traiter les différents formats de descriptions de parties[1]. Les documentations de ces packages sont de très bonnes qualités.

Plusieurs choses ont motivé la création de MP*chess*. Tout d'abord, avec chessboard l'ajout d'ensemble de pièces n'est pas très aisé, car cela repose sur des fontes. De plus, je trouve que le dessin de diagrammes de parties d'échec est quelque chose de très graphique, et que le passage par un langage dédié au dessin offre plus de souplesse et quoi de mieux que METAPOST [7].

Avec MP*chess*, on construit l'image finale du plateau d'échec avec les pièces par couches successives. Ainsi, on commencera par produire et dessiner le plateau (`backboard`), que l'on pourra modifier en colorant par exemple certaines cases, ensuite on ajoutera les pièces de la position (`chessboard`), et enfin, on pourra annoter le tout avec des marques, des couleurs, des flèches, etc.

Par ailleurs, MP*chess* produit des images proches graphiquement de ce que peut fournir l'excellent site *open source* `https://lichess.org`. Vous verrez que les couleurs, les pièces et l'aspect général sont largement inspirés de ce que propose ce site.

1. Elle a même développé le package chessfss pour gérer diverses fontes d'échec.

3 Plateau

Le plateau est appelé avec MP*chess* `backboard`. Il faudra initialiser le plateau avant de le dessiner. Cela se fait avec la commande suivante :

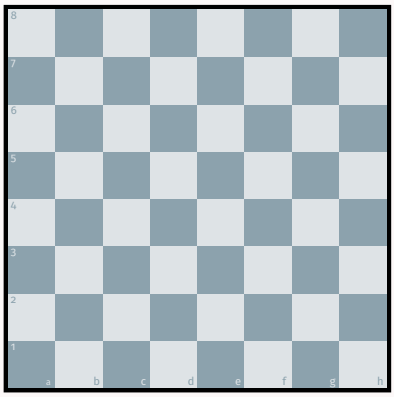
`init_backboard`

Cette commande construit une `picture` de METAPOST nommée `backboard`. Il faudra ensuite la tracer comme l'illustre l'exemple suivant.

Exemple 1

```
input mpchess

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



Cette initialisation permettra de prendre en compte les différentes options et fonctionnalités que nous allons décrire dans la suite.

3.1 Réglage des tailles

Lors de la création du `backboard`, on peut décider de la largeur de celui-ci. Cela se fait grâce à la commande suivante :

`set_backboard_width(`⟨*dim*⟩`)`

⟨***dim***⟩ **:** est la largeur de plateau de jeu souhaitée (avec l'unité). Par défaut, cette dimension est à 5 cm.

L'utilisation de cette commande est illustré à l'exemple 2. Cette commande est à utilisée avant `init_backboard` pour qu'elle soit prise en compte à la création de l'image.

On peut récupérer la dimension du plateau de jeu par la commande suivante.

`get_backboard_width`

Cette commande retourne un type `numeric`.

3.2 Nombre de case

Par défaut, le plateau de jeu contient 64 cases (8 × 8). On peut modifier cela avec la commande suivante :

`set_backboard_size(⟨nbr⟩)`

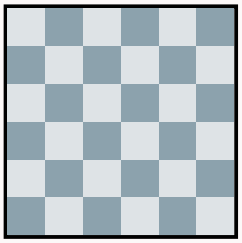
⟨***nbr***⟩ **:** est le nombre de cases souhaité. Le plateau sera alors carré de taille ⟨*nbr*⟩×⟨*nbr*⟩. Par défaut ce nombre est à 8.

Encore une fois, cette commande est à utilisée avant `init_backboard` pour qu'elle soit prise en compte comme le montre l'exemple suivant.

Exemple 2

```
input mpchess

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



Pour obtenir la taille du plateau de jeu, on pourra utiliser la commande suivante.

`get_backboard_size`

Cette commande retourne un type `numeric`.

3.3 Dimension d'une case

En fonction du nombre de cases sur le plateau et la largeur prescrite pour le plateau, MP*chess* calcule la dimension (largeur ou hauteur) d'une case. Cela sert d'unité générale. Pour l'obtenir, on utilisera la commande suivante.

`get_square_dim`

Cette commande retourne un `numeric`.

3.4 Réglage du thème de couleur

3.4.1 Thèmes prédéfinis

Plusieurs thèmes de couleurs sont accessibles. Pour choisir un thème de couleur, on utilisera la commande suivante :

`set_color_theme(⟨string⟩)`

⟨***string***⟩ peut valoir :

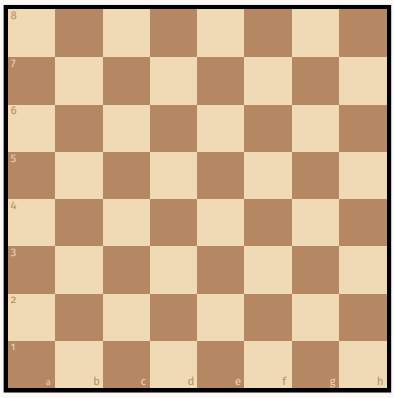
- `"BlueLichess"` (thème par défaut);
- `"BrownLichess"`;

— "GreenLichess";
— "PinkPyramidalLichess";
— "Wood";
— "Classical";
— "Dotted";
— ou "User".

Les exemples suivants montrent les résultats obtenus pour les thèmes prédéfinis.

Exemple 3

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



Le tableau 1 montre les résultats des différents thèmes fournis par MP*chess*.

L'utilisation de "User" est un cas particulier, car il permet de définir soit même les casses noires et blanches. Pour que cela fonctionne, il faut définir en plus une macro buildUserSquare qui construit les images (picture) internes _blackSquarePic et _whiteSquarePic dont voici un prototype :

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

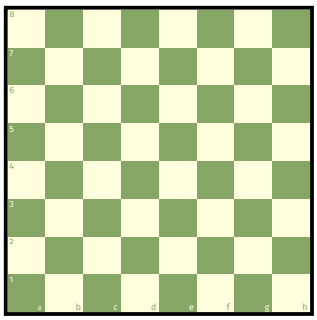
La macro prend en argument une unité, et doit construire deux carrés de côté cette unité.

3.4.2 Configuration d'un thème personnel

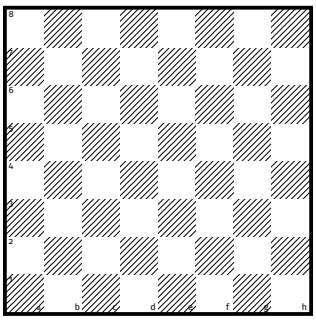
Couleurs. Un thème de couleur conciste en la définition de deux couleurs. Celles-ci peuvent se définir avec les commandes suivantes[2].

2. Attention, lors du passage à la version 0.6, set_white_color est devenu set_white_squares_color et set_black_color est devenu set_black_squares_color.

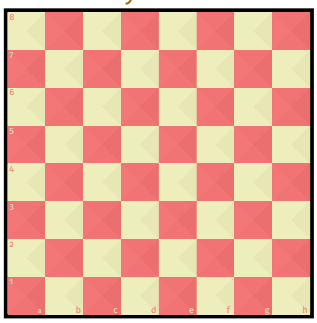
Thème "GreenLichess"



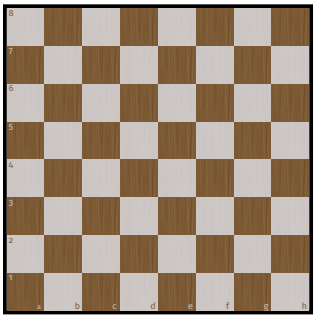
Thème "Classical"



Thème "PinkPyramidalLichess"



Thème "Wood"



Thème "Dotted"

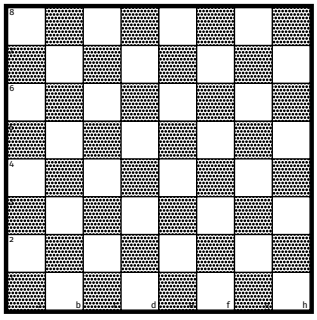


TABLE 1 – Les différents thèmes de couleur fournis par MP*chess*.

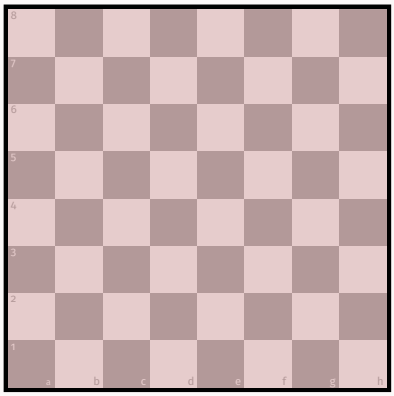
`set_white_squares_color(⟨color⟩)`

`set_black_squares_color(⟨color⟩)`

⟨*color*⟩ est une `color` METAPOST.

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;
```



Type de coloriage. On peut choisir un type de coloriage avec la commande suivante :

`set_board_type(⟨string⟩)`

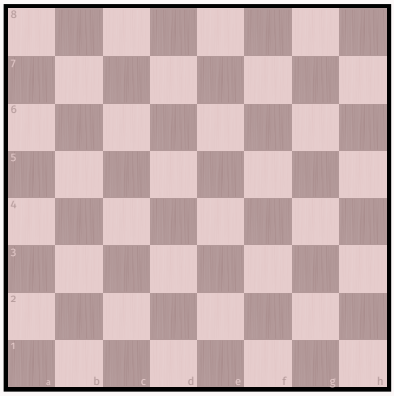
Les trois types de coloriage fournis par MP*chess* se choisissent avec ⟨***string***⟩ qui peut valoir :

- `"flat"`, coloriage à plat (défaut);
- `"pyramidal"`, coloriage *pyramidal* à la Lichess;
- `"wood"`, imitation bois.

Voici un exemple de paramétrage de couleur et de type de coloriage.

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 Affichage des coordonnées

Vous avez pu constater dans les divers exemples que par défaut, les coordonnées sont, comme le fait le site Lichess, inscrites en petit à l'intérieur des cases.

MP*chess* permet de choisir le positionnement de ces coordonnées à l'extérieur ou à l'intérieur du plateau avec la commande suivante[3].

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

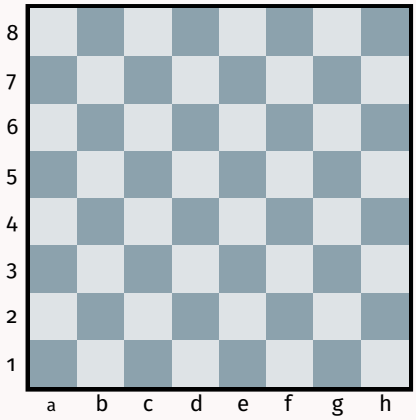
⟨***string***⟩ peut valoir :

- `"inside"` (par défaut);
- `"outside"`.

Le résultat est alors le suivant.

Exemple 6

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



Vous pouvez constater dans cette documentation qu'avec luamplib et LaTeX, la fonte est la fonte du document courant. Pour tracer ces lettres et ces chiffres, MP*chess* utilise l'opérateur METAPOST `infont` et la fonte est réglée à `defaultfont` par défaut[4]. On peut modifier cette fonte avec la commande suivante[5].

`set_coordinates_font(`⟨*font*⟩`)`

Il faudra alors utiliser les conventions de nommage propres à l'opérateur `infont` de METAPOST et nous renvoyons à la documentation [7] pour plus de détails.

On pourra aussi supprimer les coordonnées avec la commande suivante[6].

`hide_coordinates`

3. Attention, en version 0.6, `set_coords_inside` et `set_coords_outside` ont été remplacés par `set_coordinates_position`.
4. Avec luamplib l'opérateur `infont` est redéfini et son argument est simplement ignoré, ainsi, il n'est pas possible de modifier la fonte de composition des coordonnées.
5. Attention, en version 0.6, `set_coords_font` est devenue `set_coordinates_font`.
6. Attention, dans la version 0.6, `set_no_coords` est devenu `hide_coordinates` et `set_coords` est devenu `show_coordinates`

Et la commande inverse aussi existe.

`show_coordinates`

3.6 Vue blanche ou noire

Pour choisir si l'on souhaite voir le plateau du côté blanc ou noir, MP*chess* fournit deux commandes.

`set_white_view`

`set_black_view`

Par défaut, on voit l'échiquier côté blanc.

3.7 Noms des joueurs

On peut renseigner les noms des joueuses ou des joueurs pour qu'ils soient notés autour de l'échiquier. Ceci se fait avec les commandes suivantes.

`set_white_player(⟨string⟩)`

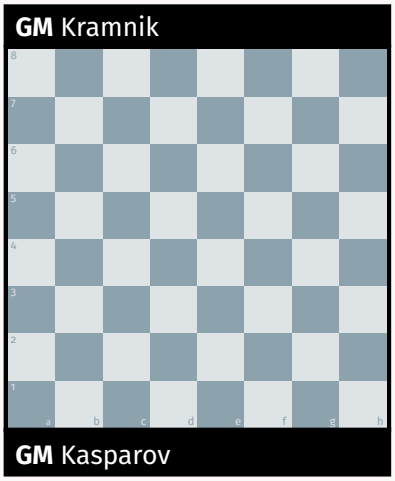
`set_black_player(⟨string⟩)`

⟨***string***⟩ **:** est la chaîne de caractères interprétée par LaTeX à afficher.

Exemple 7

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

init_backboard;
draw backboard;
endfig;
```

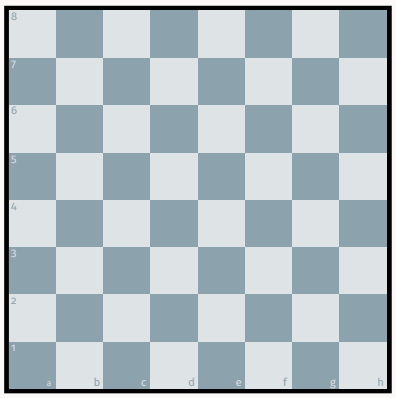


Il est possible de placer les noms sur le côté droit du plateau sans les bandeaux noirs présents par défaut. Cela se produit soit si les coordonnées sont imprimées à l'extérieur du plateau, soit si la commande suivante est utilisée.

`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 Pièces et positions

MP*chess*, comme décrit plus haut, construit le graphique d'une position d'échec couche par couche. Cette partie est dédiée à la configuration des pièces et des positions.

En interne, MP*chess* construit un tableau sur la grille du plateau. Ensuite, des macros permettent de générer une `picture` à dessiner *par dessus* le plateau (`backboard`).

4.1 Réglage du thème des pièces

MP*chess* fournit pour l'instant trois thèmes de pièces. Le thème par défaut est appelé `mpchess`. Il a été designé pour ce package METAPOST. Il a été proposé au projet Lichess, et a été accepté. Ainsi, vous aurez aussi accès à l'ensemble de pièces `mpchess` avec Lichess[7] ! Un autre thème est emprunté à Lichess (`cburnett`) et l'autre est emprunté au package skak [5][8].

Pour choisir le thème on utilisera la commande suivante.

`set_pieces_theme(`⟨*string*⟩`)`

⟨***string***⟩ **:** peut valoir :

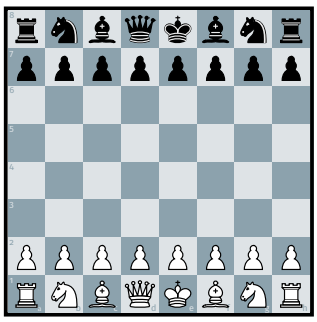
- `"mpchess"` (valeur par défaut), pour obtenir l'ensemble de pièces spécialement conçu pour ce package ;
- `"cburnett"`, pour obtenir l'ensemble de pièces nommé *cburnett* de Lichess ;
- `"skak"`, pour obtenir l'ensemble de pièces du package skak.

Le tableau 2 montre le résultat des trois ensembles de pièces.

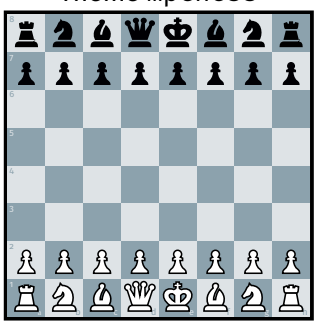
7. Les projets libres se nourrissent mutuellement ! Même si évidemment, ce package a bien plus emprunté à Lichess que le contraire.

8. Qui fournit le code METAFONT pour la fonte de pièces d'échec, code qui a été facilement adapté en METAPOST pour MP*chess*.

Thème cburnett



Thème mpchess



Thème skak

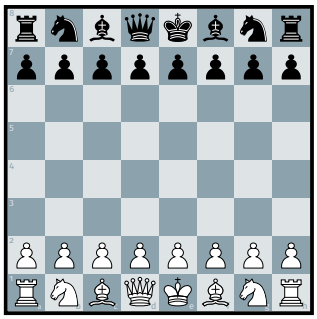


TABLE 2 – Les différents thèmes de pièces fournits par MP*chess*.

4.2 Trait

MP*chess* indique qui a le trait entre les blancs et les noirs. Ceci ce fait par un petit triangle coloré (blanc ou noir) à l'extérieur du plateau (que vous pourrez observer dans les nombreux exemples suivants).

Pour spécifier qui a le trait on utilisera les commandes suivantes.

`set_white_to_move`

`set_black_to_move`

Par défaut, c'est aux blancs de jouer, et cette information est affichée.

Pour activer ou désactiver l'affichage du trait, on utilisera une des deux commandes suivantes[9].

`show_whos_to_move`

`hide_whos_to_move`

4.3 Dessiner une position

Les commandes décrites ci-dessous permet de construire une position de plusieurs façons (ajout de pièces une à une, lecture de fichier FEN, etc.). Une

9. Attention, lors du passage à la version 0.6, `set_whos_to_move` et `unset_whos_to_move` sont devenus `show_whos_to_move` et `hide_whos_to_move`.

fois une position construite, on peut la tracer grâce à la commande suivante qui génère une `picture`.

`chessboard`

L'utilisation de cette commande va être largement illustrée dans les exemples suivants.

4.4 Construire une position

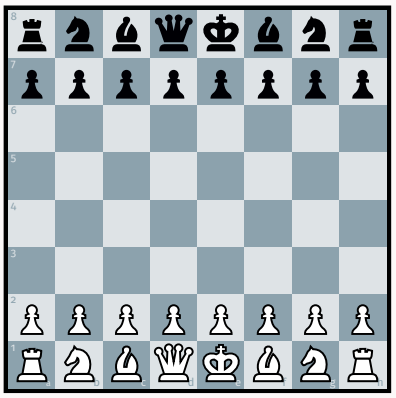
4.4.1 Initialisation

Pour obtenir la position initiale d'une partie, il suffit d'utiliser la commande suivante.

`init_chessboard`

Exemple 9

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



On pourra aussi initialiser un `chessboard` vide grâce à la commande suivante.

`set_empty_chessboard`

4.4.2 Ajout de pièces

On peut ajouter des pièces pour construire une position grâce aux deux commandes suivantes.

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

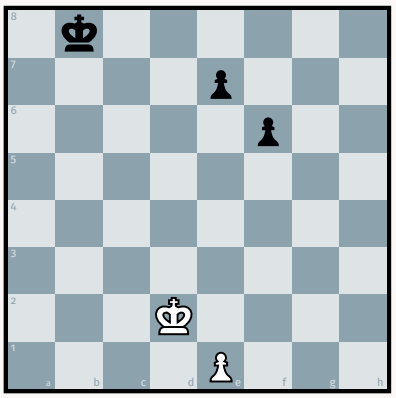
`add_black_pieces(⟨piece1⟩,⟨piece2⟩,etc.)`

Ces commandes prennent des listes de ⟨***piece***⟩ qui sont des chaînes de caractères qui décrivent la pièce et la position en utilisant la notation algébrique. Il n'y a pas de limitation au nombre de pièces dans la liste.

L'exemple suivant illustre l'utilisation de ces commandes.

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 Suppression de pièces

MP*chess* fournit plusieurs commandes permettant de supprimer des éléments d'une position.

La première commande permet de supprimer un élément d'une case. Cette commande permet de prendre une liste de cases, en utilisant la notation algébrique.

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

Les ⟨***square1***⟩, ⟨***square2***⟩, etc., sont des chaînes de caractères, par exemple `"a3"`.

La commande suivante permet de supprimer un ensemble de cases dans une région déterminé par deux coordonnées sur le plateau. Cette commande permet de prendre une liste de régions.

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

Les ⟨***area1***⟩, ⟨***area2***⟩, etc., sont des chaînes de caractères constituées de deux coordonnées séparées par un tiret, par exemple `"a3-g7"`.

La commande suivante permet de supprimer l'ensemble des cases d'une colonne déterminé par une lettre sur le plateau. Cette commande permet de prendre une liste de colonnes.

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

Les ⟨***file1***⟩, ⟨***file2***⟩, etc., sont des chaînes de caractères constituées d'une lettre, par exemple `"a"`.

La commande suivante permet de supprimer l'ensemble des cases d'une ligne déterminé par un nombre sur le plateau. Cette commande permet de prendre une liste de lignes.

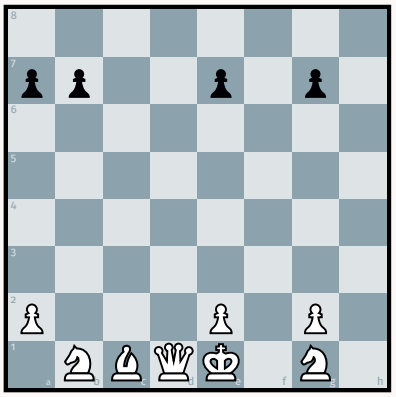
`clear_ranks(`⟨*rank1*⟩`,`⟨*rank2*⟩`,etc.)`

Les ⟨***rank1***⟩, ⟨***rank2***⟩, etc., sont des chaînes de caractères constituées d'un nombre, par exemple `"4"`.

L'utilisation de l'ensemble des ces commandes est illustrée dans l'exemple suivant.

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 Lecture de données au format FEN

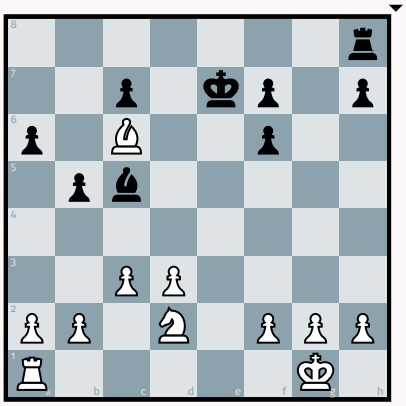
MP*chess* permet de lire une position au format FEN grâce à la commande suivante.

`build_chessboard_from_fen(⟨string⟩)`

⟨***string***⟩ **:** est une chaîne de caractères décrivant une position au format FEN. Notons que toutes les informations après l'information du *trait* (w ou b) sont ignorées.

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;
```



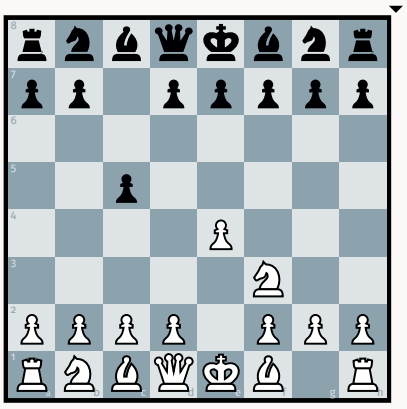
Il est aussi possible de lire un fichier externe contenant sur la première ligne une chaîne de caractères au format FEN avec la commande suivante.

`build_chessboard_from_fen_file(⟨string⟩)`

⟨***string***⟩ **:** est une chaîne de caractères (entre double-quotes) indiquant le nom du fichier à lire.

Exemple 13

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



4.6 Lecture de données au format PGN

MP*chess* permet aussi de lire une chaîne de caractères au format PGN. Attention, il s'agit d'une gestion partielle du format, en particulier MP*chess* ne gère pas les *tags* du format. En réalité, MP*chess* ne traite que la chaîne de caractères décrivant les coups joués. De même, le format PGN accepté par MP*chess* n'accepte ni les variantes ni les commentaires.

Lorsqu'une telle fonctionnalité est utilisé, MP*chess* stocke toutes les positions intermédiaires et permet ainsi de les représenter.

Pour construire les positions, on utilisera la commande suivante.

`build_chessboards_from_pgn(⟨string⟩)`

Une fois le positions construites, on pourra les représenter grâce à la commande suivante.

`chessboard_step(⟨int⟩)`

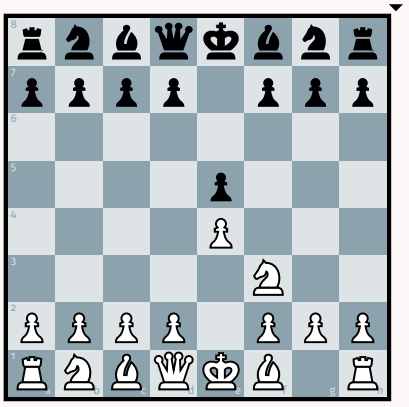
⟨***int***⟩ **:** est le numéro du l'étape. La configuration initiale est numérotée 0, et ensuite, chaque coup, blanc ou noir, est numéroté.

Cette commande, comme `chessboard` (voir page 14), retourne une `picture` .

L'exemple suivant illustre l'utilisation de ces commandes.

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;
```



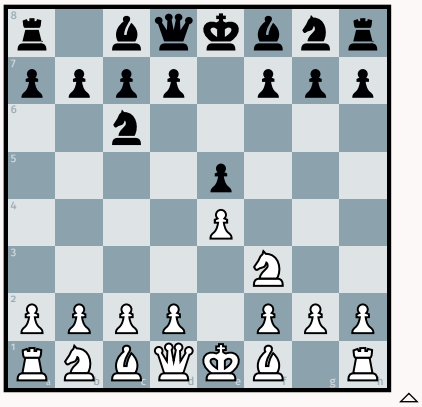
Il est aussi possible de lire un fichier externe contenant sur la première ligne une chaîne de caractères au format PGN avec la commande suivante.

`build_chessboard_from_pgn_file(`⟨*string*⟩`)`

⟨***string***⟩ **:** est une chaîne de caractères (entre double-quotes) indiquant le nom du fichier à lire.

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 Montrer le dernier coup

On peut afficher automatiquement le dernier coup grâce à la commande suivante.

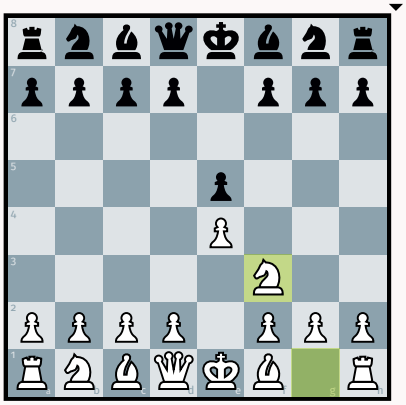
`show_last_move(`⟨*int*⟩`)`

⟨***int***⟩ **:** est le numéro du l'étape. La configuration initiale est numérotée 0, et ensuite, chaque coup, blanc ou noir, est numéroté.

Cette commande colorie en transparence les deux cases de départ et d'arrivée du dernier coup. Ainsi, elle doit être utilisée entre le dessin du plateau (`draw backboard`) et le dessin des pièces (`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;
```



On pourra configurer la couleur utilisée pour colorier en transparence les cases du dernier coup grâce à la commande suivante.

`set_last_move_color(⟨color⟩)`

⟨***color***⟩ **:** est une `color` METAPOST.

4.6.2 Obtenir le nombre de coups

On pourra récupérer le nombre de *demi*-coups grâce à la commande suivante.

`get_halfmove_number`

Cette commande retourne un `numeric`.

On pourra aussi récupérer le nombre de coups « total » au sens où il sont numéroté dans le format PGN, grâce à la commande suivante :

`get_totalmove_number`

Cette commande retourne un `numeric`.

5 Annotation

De nombreuses commandes permettent d'annoter l'échiquier (flèche, couleur, cercle, croix, etc.).

5.1 Flèches

La commande pour tracer des flèches sur l'échiquier est la suivante.

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

⟨***color***⟩ **:** est une `color` METAPOST.

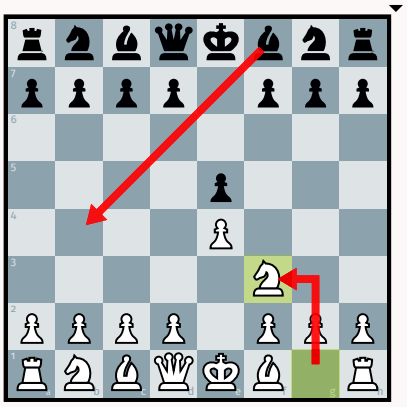
⟨***string1***⟩ : est une chaîne de caractères (entre double-quotes) constituée de deux coordonnées (lettre et chiffre) séparés par deux caractères qui peuvent être

- `--` pour relier les deux cases en ligne droite ;
- `-|` pour relier les deux cases en ligne brisée, d'abord horizontalement puis verticalement ;
- `|-` pour relier les deux cases en ligne brisée, d'abord verticalement puis horizontalement.

L'exemple suivant illustre l'utilisation de cette commande.

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;
```



On pourra modifier l'épaisseur des flèches grâce à la commande suivante.

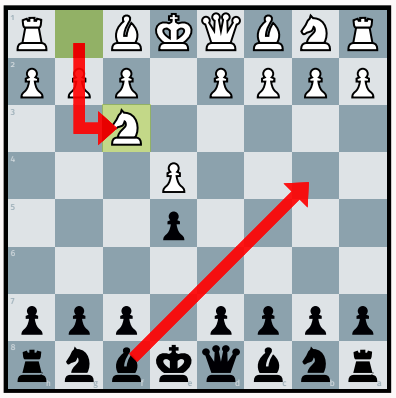
`set_arrow_width(`⟨*coeff*⟩`)`

⟨***coeff***⟩ : est un coefficient (`numeric`) qui permet de régler la largeur des flèches proportionnellement à la largeur d'une case de l'échiquier. Par défaut, ce coefficient vaut `0.08`.

L'exemple suivant illustre l'utilisation de cette commande.

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 Coloration de cases

MP*chess* permet aussi de colorer des cases grâce à la commande suivante.

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

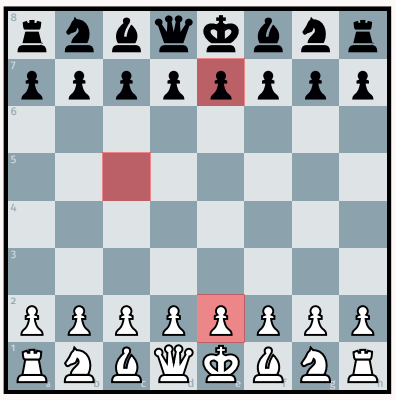
⟨***color***⟩ : est une `color` METAPOST.

⟨***coord1***⟩ : est une chaîne de caractères (entre double-quotes) constituée de deux coordonnées (lettre et chiffre).

L'exemple suivant permet d'illustrer l'utilisation de cette commande.

Exemple 19

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



Cette commande colorie les cases avec une certaine transparence pour s'adapter aux cases blanches et noires.

5.3 Cercles

MP*chess* permet d'entourer des cases avec des cercles grâce à la commande suivante.

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

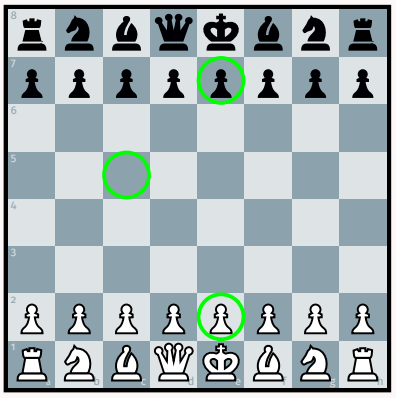
⟨***color***⟩ **:** est une `color` METAPOST.

⟨***coord1***⟩ **:** est une chaîne de caractères (entre double-quotes) constituée de deux coordonnées (lettre et chiffre).

L'exemple suivant permet d'illustrer l'utilisation de cette commande.

Exemple 20

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



5.4 Croix

MP*chess* permet de tracer des croix sur des cases grâce à la commande suivante.

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

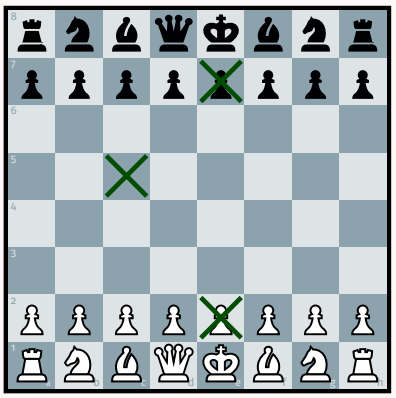
⟨***color***⟩ **:** est une `color` METAPOST.

⟨***coord1***⟩ **:** est une chaîne de caractères (entre double-quotes) constituée de deux coordonnées (lettre et chiffre).

L'exemple suivant permet d'illustrer l'utilisation de cette commande.

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 Commentaires de coup

MPchess permet d'afficher les classiques commentaires de coups du type «!? » grâce à la commande suivante.

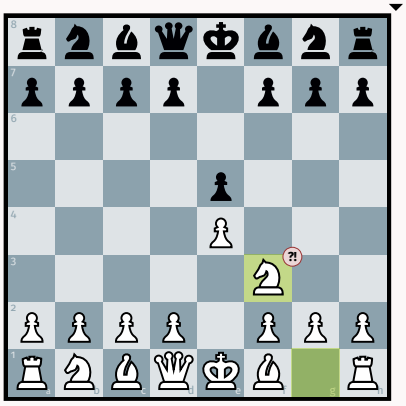
`draw_comment(`⟨*str*⟩`,`⟨*pos*⟩`)`

⟨***str***⟩ **:** est une chaîne de caractères (entre double-quotes) à afficher.

⟨***pos***⟩ **:** est une chaîne de caractères (entre double-quotes) constituée d'une coordonnée (lettre et chiffre).

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;
```



La couleur des annotation de commentaires peut être changé grâce à la commande suivante.

`set_comment_color(`⟨*color*⟩`)`

5.6 Lignes principales

MP*chess* fournit une commande permettant d'afficher les flèches des coups des lignes principales d'analyses. Il y a les commandes pour les deux couleurs.

`draw_white_main_lines(`⟨*move1*⟩`,`⟨*move2*⟩`,etc.)`

`draw_black_main_lines(`⟨*move1*⟩`,`⟨*move2*⟩`,etc.)`

⟨***move1***⟩, ⟨***move2***⟩, **etc. :** sont les coups à illustrer par une flèche, en suivant la notation de type PGN.

Lorsqu'on utilise la lecture de format PGN pour la construction des positions à afficher, on pourra alors utiliser les commandes suivantes permettant de spécifier quelle étape de la partie on commente.

`draw_white_main_lines_step(`⟨*step*⟩`)(`⟨*move1*⟩`,`⟨*move2*⟩`,etc.)`

`draw_black_main_lines_step(`⟨*step*⟩`)(`⟨*move1*⟩`,`⟨*move2*⟩`,etc.)`

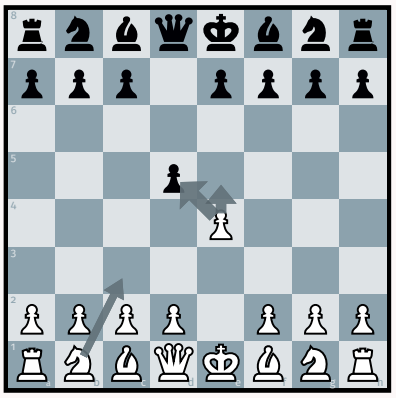
⟨***step***⟩ **:** est l'étape de la partie qu'on souhaite annoter ;

⟨***move1***⟩, ⟨***move2***⟩, **etc. :** sont les coups à illustrer par une flèche, en suivant la notation de type PGN.

L'exemple suivant permet d'illustrer l'utilisation de cette commande.

Exemple 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;
```



Pour changer la couleur (par défaut `0.3[_blackColorSquare,black]`), on utilisera la commande suivante.

`set_main_lines_color(`⟨*color*⟩`)`

5.7 Mouvements possibles

MP*chess* permet, à la manière du site https://lichess.org, de montrer les mouvements possibles pour une pièce. Pour cela, on utilisera la commande suivante.

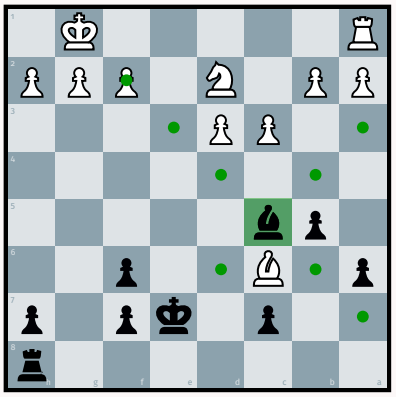
`show_possible_moves(⟨square⟩)`

⟨***square***⟩ **:** est une chaîne de caractères indiquant au format classique les coordonnées de la case où la pièce se situe.

Cette commande doit être utilisée après avoir tracé le `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);
  set_black_view;
  init_backboard;
  draw backboard;
  draw chessboard;
  show_possible_moves("c5");
  endfig;
```



Dans le cas où plusieurs positions ont été construites avec le format PGN, on utilisera la commande suivante pour acceder à la position choisie.

`show_possible_moves_step(⟨step⟩)(⟨square⟩)`

⟨***step***⟩ **:** est l'étape de la partie qu'on souhaite annoter ;

⟨***square***⟩ **:** est une chaîne de caractères indiquant au format classique les coordonnées de la case où la pièce se situe.

Par défaut, la couleur est réglée à `0.4[green,black]`, mais celle-ci peut-être modifiée avec la commande suivante.

`set_possible_moves_color(⟨color⟩)`

6 Divers

6.1 Réinitialisation du `chessboard`

Pour réinitialiser la structure interne stockant les positions des pièces, on pourra utiliser la commande suivante.

`clear_chessboard`

6.2 Réinitialisation globale

Pour réinitialiser les valeurs des différents paramètres de MP*chess*, on pourra utiliser la commande suivante.

```
reset_mpchess
```

6.3 Découpe de l'échiquier

On pourra faire une découpe dans l'image de l'échiquier grâce à la commande suivante.

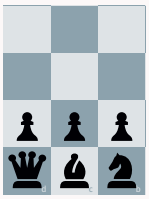
```
clip_chessboard(⟨string⟩)
```

⟨***string***⟩ **:** est une chaîne de caractères (entre double-quotes) composée de deux coordonnées (lettre et chiffre) séparées par un tiret, par exemple `"a1-c6"`.

Voici un exemple d'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 Utilisation avec LaTeX

7.1 Utilisation avec pdfLaTeX ou XeLaTeX

Il existe plusieurs façons d'inclure les images produites par MP*chess* dans un document LaTeX. La première est de générer des fichiers PDF avec METAPOST puis de les inclure avec `\includegraphics`. Cette solution fonctionne avec tous les moteurs.

On pourra aussi utiliser les packages gmp ou mpgraphics avec pdfLaTeX ou XeLaTeX[10].

10. Nous tenons à remercier Quark67 pour les questions et les conseils.

7.1.1 Avec mpgraphics

Avec mpgraphics [6], on chargera MP*chess* avec l'environnement `mpdefs` et on pourra produire des images grâce à du code METAPOST mais sans avoir recours à `beginfig` et `endfig`, le code pour générer une figure METAPOST se trouvant dans l'environnement `mpdisplay`. Il faudra de plus utiliser l'option `-shell-escape` à la compilation du document LaTeX.

Voici un exemple complet d'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{document}
```

7.1.2 Avec gmp

L'utilisation de gmp [3] est assez similaire à celle de mpgraphics. Quelques commandes sont toutefois différentes, mais comme avec mpgraphics, on n'aura pas recours à `beginfig` et `endfig`. Le chargement de MP*chess* peut se faire au chargement du package, et le code METAPOST se trouve dans l'environnement `mpost`. Là encore il faudra compiler le document LaTeX avec l'option `-shell-escape`.

Voici un exemple complet d'illustration.

```
\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 Utilisation avec LuaLaTeX et luamplib

Il est tout à fait possible d'utiliser MP*chess* directement dans un fichier LaTeX avec LuaLaTeX[11] et le package luamplib. C'est d'ailleurs ce qui est fait pour écrire cette documentation.

Il suffira alors de mettre le code METAPOST dans l'environnement `mplibcode` comme illustré plus bas.

MP*chess* utilise, pour certaines fonctionnalités, l'opérateur `infont` de METAPOST. Ainsi, pour que le contenu de ces fonctionnalités soit composé dans la fonte courante du document, on devra ajouter dans son document LaTeX, la commande :

```
\mplibtextextlabel{enable}
```

Pour plus de détails sur ces mécanismes, nous renvoyons à la documentation du package luamplib [4].

On pourra charger globaletement MP*chess* avec la commande suivante.

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

Voici un exemple complet d'illustration (à compiler avec 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 Font TrueType

Le package MP*chess* fournit une fonte très simple composé d'uniquement les 12 glyphes correspondant aux pièces noires et blanches dans la table Uni-

11. Rappelons que METAPOST fait partie intégrante de LuaTeX.

code. Pour y avoir accès il suffit d'utiliser LuaLaTeX ou XeLaTeX et le package fontspec. Pour faciliter son utilisation, nous proposons quelques définitions.

```
\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}

Voici l'enchainement de coups : 1. e4 e5 2. \wB c4 d6 3.
     \wN f3 \bB g4 4. \wN
c3 g6 5. \wN xe5 \bB xd1.

\end{document}
```

Qui produira :

« Voici l'enchainement de coups : 1. e4 e5 2. ♗c4 d6 3. ♘f3 ♝g4 4. ♘c3 g6 5. ♘xe5 ♝xd1. »

8 To do

De nombreuses choses sont à ajouter à MP*chess*. Parmi celles-ci, on peut penser à :

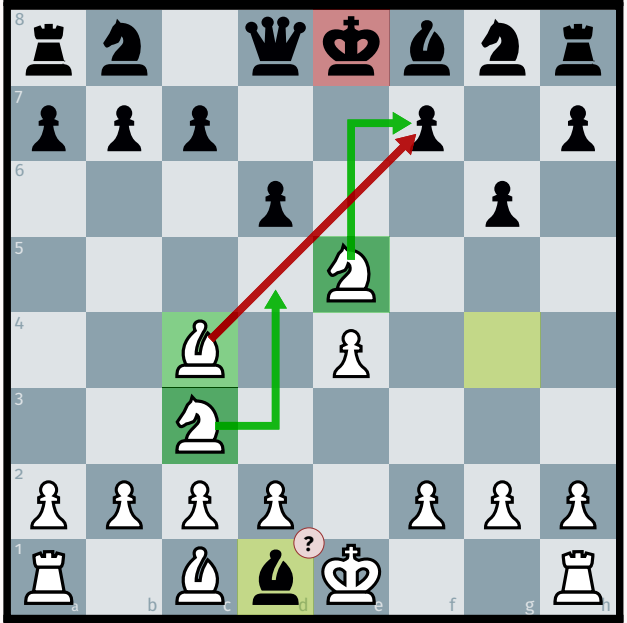
- afficher les pièces capturées, ou le différentiel des pièces capturées ;
- afficher le temps restant pour chaque joueur ;
- parser le format PGN complet et donc avec les tags optionnels ;
- ajouter les coordonnées en extérieur lorsque le plateau est découpé ;
- ajouter des thèmes de pièces.

9 Un exemple complet

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_backboard_width(8cm);
set_white_player("Kermur de Legal");
set_black_player("Saint-Brie");
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;
```

Saint-Brie



Kermur de Legal

10 Historique

v0.8, 26 octobre 2024 : Ajout du thème d'échiquier `"Dotted"` et ajout du thème d'échiquier `"User"` qui permet à l'utilisateur ou l'utilisatrice de définir les casses blanches et noires.

v0.7, juillet 2023 : Ajustement du cavalier noir, ajout de thèmes d'échiquier (`GreenLichess`, `PinkPyramidalLichess`, `Wood` avec des types de coloriage (`set_board_type`).

v0.6, avril 2023 : Corrections de bugs concernant la gestion des roques, et des ambiguïtés de pièces pour les déplacements sous format PGN. Changement de `set_white_color` en `set_white_squares_color` et `set_black_color` en `set_black_squares_color`. Changement de `set_no_coords` en `hide_coordinates` et `set_coords` en `show_coordinates`. Changement de `set_whos_to_move` en `show_whos_to_move` et `unset_whos_to_move` en `hide_whos_to_move`. Changement de `set_coords_inside` et `set_coords_outside` en `set_coordinates_position`. Changement de `set_coords_font` en `set_coordinates_font`.

v0.5, 20 avril 2023 : Correction d'un bug, **changement de l'ensemble de pièce par défaut** pour l'ensemble `mpchess` (qui a été ajouté à Lichess), ajout de la fonte TrueType, et mise à jour de la documentation.

v0.4, 6 avril 2023 : Corrections dans la documentation, notamment la version anglaise ; ajout des commandes pour visualiser les mouvements possible pour une pièce (section 5.7).

v0.3, 29 mars 2023 : Petit bug.

v0.2, 28 mars 2023 : Ajout des commandes de lecture de fichiers PGN et FEN ; ajout des commandes d'affichage des lignes principales d'analyse ; suppression du thème `staunty` (pour cause de licence) et création du thème de pièces `mpchess`.

v0.1, 23 mars 2023 : Première publication sur le CTAN.

11 Remerciements

Nous souhaitons remercier Quark67 pour ses retours et ses corrections, Douglas Johnson pour avoir corrigé la version anglaise de la documentation et Hans Nieuwenhuis pour ses conseils. Ces retours et encouragements font extrêmement plaisir !

Références

[1] Ulrike FISCHER. *The chessboard package. Print chess boards*. Version 1.9. 9 mars 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. 9 mars 2023. URL : https://ctan.org/pkg/xskak.

[3] Enrico Gregorio. *The gmp package. Enable integration between MetaPost pictures and LaTeX*. Version 1.0. 24 juin 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. 12 jan. 2022. URL : https://ctan.org/pkg/luamplib.

[5] Torben Hoffmann. *The skak package. Fonts and macros for typesetting chess games*. Version 1.5.3. 10 déc. 2021. URL : https://ctan.org/pkg/skak.

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

[7] The MetaPost Team et John Hobby. *The metapost package. A development of Metafont for creating graphics*. 26 août 2021. URL : https://ctan.org/pkg/metapost.

Index des commandes