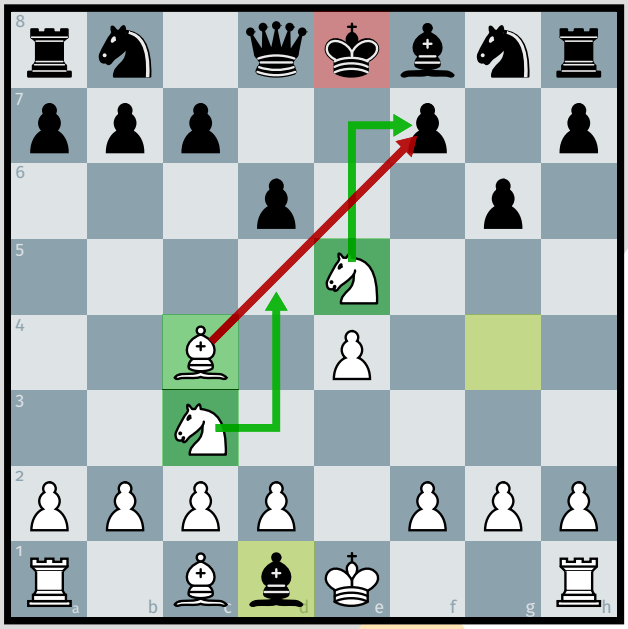
MPchess

dessiner des plateaux d'échecs et des positions avec METAPOST



Contributor
Maxime CHUPIN
notezik@gmail.com

Version 0.1, 2023, March, 8th
https://plmlab.math.cnrs.fr/mchupin/mpchess

Résumé

Abstract

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

Table des matières

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/tex/metapost/mpchess/`

MP*chess* est constitué de 5 fichiers METAPOST :

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

1.4 Utilisation avec LuaLaTeX et luamplib

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

MP*chess* utilise, pour certaines fonctionnalités, l'opérateur `infont` de METAPOST. Ainsi, pour que le contenu de ces fonctionnalités soient 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 [3].

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] qui couplé avec le package chessboard [1]. Ulrike Fisher a réalisé là un travail d'amélioration, de maintient, et nous a fournit d'excellent 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 [5].

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 certaine 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.

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`

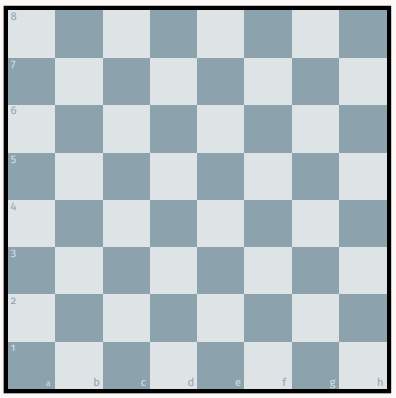
Cette commande construit une `picture` de METAPOST nommée `backboard`. Il faudra ensuite la tracer comme l'illustre l'exemple suivant.

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

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 peux 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*⟩`)`

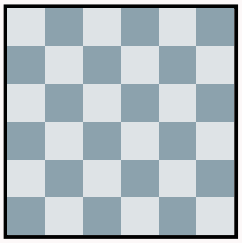
⟨***nbr***⟩ **:** est le nombre de cases souhaité. Le plateau sera alors carré de taille ⟨*nbr*⟩×⟨*nbr*⟩. Par défaut ce nombre est à 8.

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 par 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⟩)`

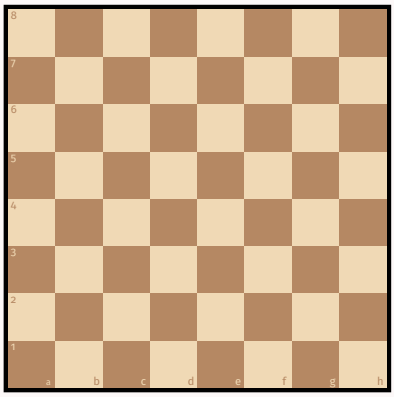
⟨*string*⟩ peut valoir :

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

Les exemples montrent les résultats obtenus.

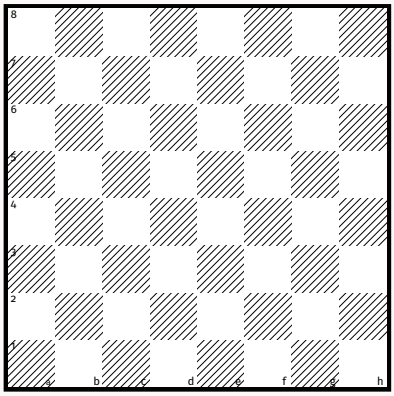
Exemple 3

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



Exemple 4

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



Les deux thèmes colorés fournis sont les couleurs des thèmes de Lichess.

3.4.2 Configuration d'un thème personnel

Un thème de couleur est en réalité simplement la définition de deux couleurs. Celles-ci peuvent se définir avec les commandes suivante.

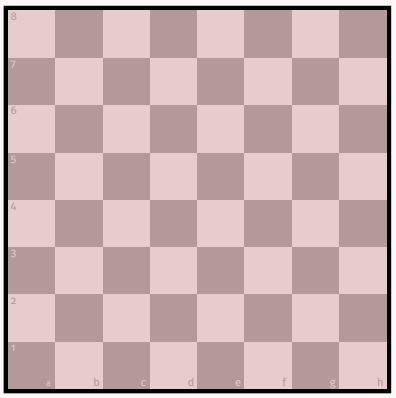
`set_white_color(⟨color⟩)`

`set_black_color(⟨color⟩)`

⟨*color*⟩ est une `color` METAPOST.

Exemple 5

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



3.5 Affichage des coordonnées

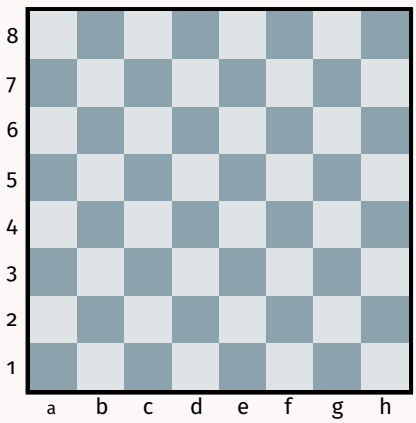
Vous avez pu constater dans les divers exemples que par défaut, les coordonnées sont, comme le fait le site Lichess, inscrites en petit à l'intérieur des cases.

MP*chess* permet de positionner ces coordonnées à l'extérieur du plateau avec la commande suivante.

`set_coords_outside`Le résultat est alors le suivante.

Exemple 6

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



Il existe aussi la commande permettant de positionner les coordonnées à l'intérieur du plateau.

`set_coords_inside`

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[2]. On peut modifier cette fonte avec la commande suivante.

`set_coords_font(⟨font⟩)`

Il faudra alors utiliser les conventions de nommage propres à l'opérateur `infont` de METAPOST et nous renvoyons à la documentation [5] pour plus de détails.

On pourra aussi supprimer les coordonnées avec la commande suivante.

`set_no_coords`

Et la commande inverse aussi existe.

`set_coords`

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⟩)`

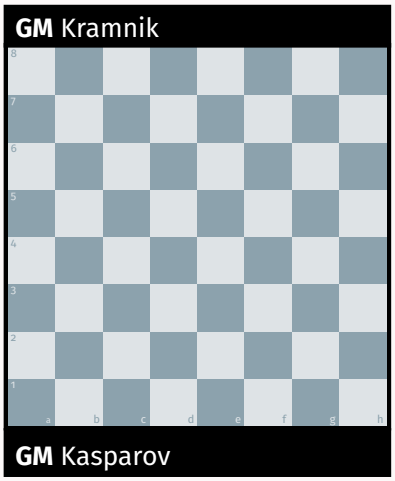
⟨***string***⟩ **:** est la chaîne de caractères interprétée par LaTeXà afficher.

2. Avec luamplib l'opérateur `infont` est redéfini et son argument est simplement ingnoré, ainsi, il n'est pas possible de modifier la fonte de composition des coordonnées.

Exemple 7

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

init_backboard;
draw backboard;
endfig;
```

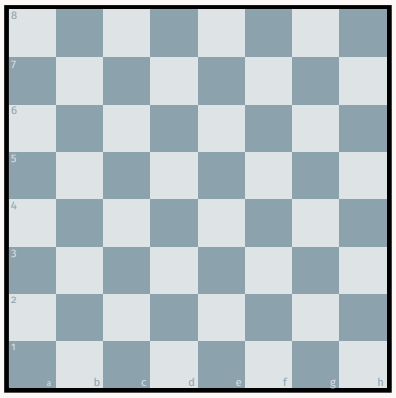


Il est possible de placer les noms sur le côté droit du plateau sans les bandeaux noirs présents par défaut. Cela se produit soit si les coordonnées sont imprimées à l'extérieur du plateau, soit si la commande suivante est utilisée.

`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;
```

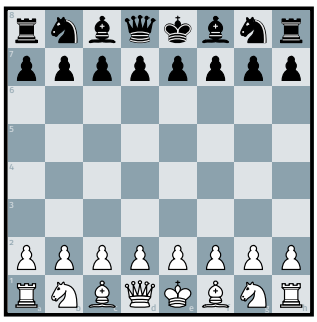


4 Pièces et positions

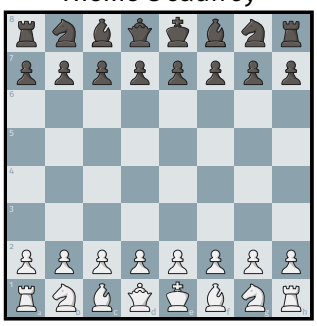
MP*chess*, comme décrit plus haut, construit le graphique d'une position d'échec couche par couche. Cette partie est dédiée à la configuration des pièces et des positions.

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`).

Thème cburnett



Thème staunty



Thème skak

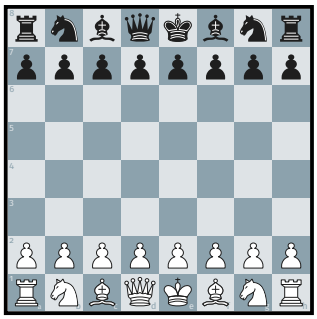


TABLE 1 – Les différents thèmes de pièces fournits par MP*chess*.

4.1 Réglage du thème des pièces

MP*chess* fournit pour l'instant trois thèmes de pièces, deux emprunté à Lichess, et l'autre emprunté au package skak [4][3].

Pour choisir le thème on utilisera la commande suivante.

`set_pieces_theme(`⟨*string*⟩`)`

⟨***string***⟩ **:** peut valoir :

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

Le tableau 1 montre le résultat des trois ensembles de pièces

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).

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

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.

`set_whos_to_move`

`unset_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, lecteur de fichie FEN, etc.). Une 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

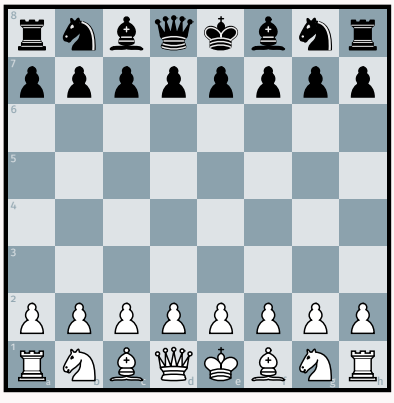
4.4.1 Initialisation

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.)`

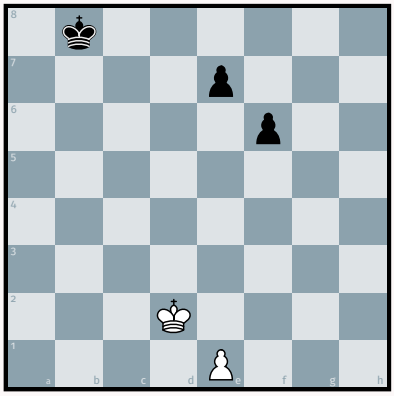
`add_black_pieces(⟨piece1⟩,⟨piece2⟩,etc.)`

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.

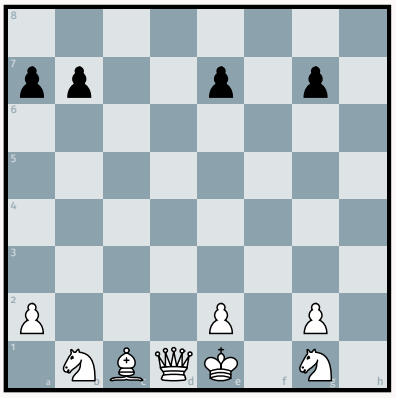
`clear_ranks(⟨rank1⟩,⟨rank2⟩,etc.)`

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

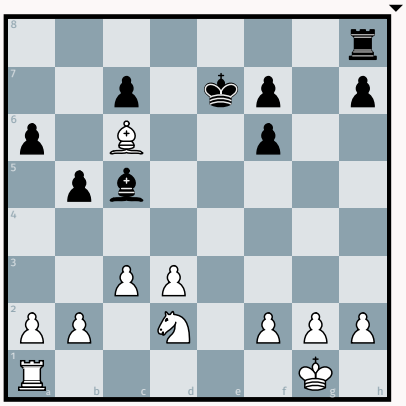
MP*chess* permet de lire une position au format FEN grâce à la commande suivante.

`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;
```



4.6 Lecture de données au format PGN

MP*chess* permet aussi de lire une chaîne de caractères au format PGN. Lorsque 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⟩)`

Le format PGN accepté par MP*chess* est un format simplifié qui n'accepte pas les variantes ni les commentaires.

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

`chessboard_step(⟨int⟩)`

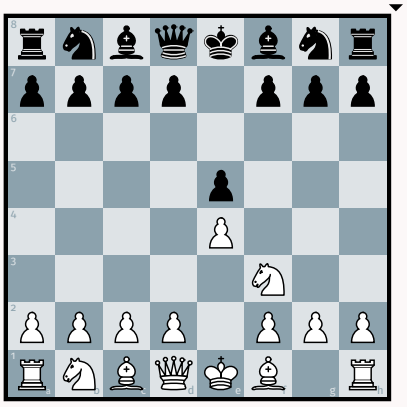
⟨***int***⟩ **:** est le numéro du l'étape. La configuration initiale est numérotée 0, et ensuite, chaque coup, blanc ou noir, est numéroté.

Cette commande, comme `chessboard` (voir page 12), retourne une `picture` .

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

Exemple 13

```
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;
```



4.6.1 Montrer le dernier coup

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

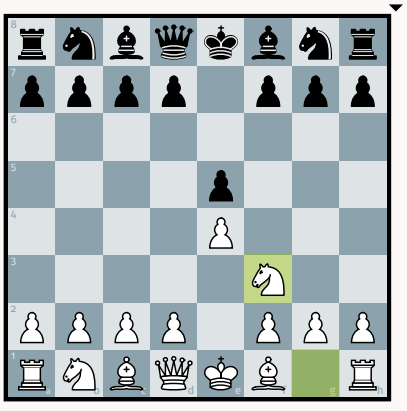
`show_last_move(⟨int⟩)`

⟨***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 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;
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_color_last_move(⟨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'annonter l'échiquier (flêche, coulour, cercle, croix, etc.).

5.1 Flêches

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

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

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

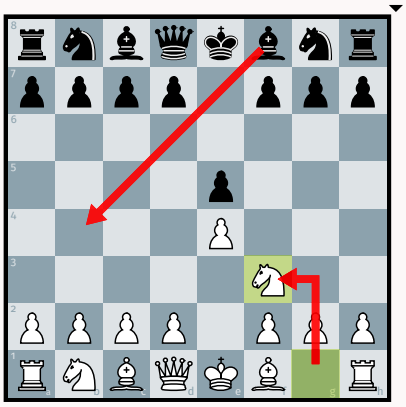
⟨***string1***⟩ **:** est une chaîne de caractères (entre double-quotes) consituée de deux coordonnées (lettre et chiffre) séparés par deux caractères qui peuvent être

- `--` 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 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;
show_last_move(3);
draw chessboard_step(3); % Nf
      3
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.)`

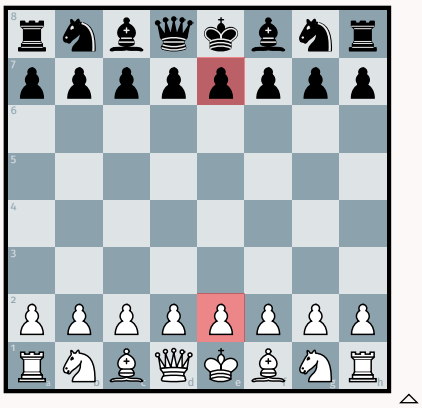
⟨***color***⟩ : est une `color` METAPOST.

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

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

Exemple 16

```
input mpchess
beginfig(0);
init_backboard;
draw backboard;
color_square(red)("e2","e7");
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.)`

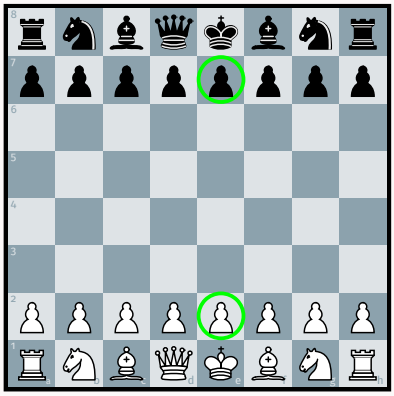
⟨***color***⟩ **:** est une `color` METAPOST.

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

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

Exemple 17

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



5.4 Croix

5.5 Commentaires de coup

6 Divers

Reset Clear chessboard

7 To do

— Captured

—

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] 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.

[4] 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.

[5] 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