Simulation d’un jeu d’échecs

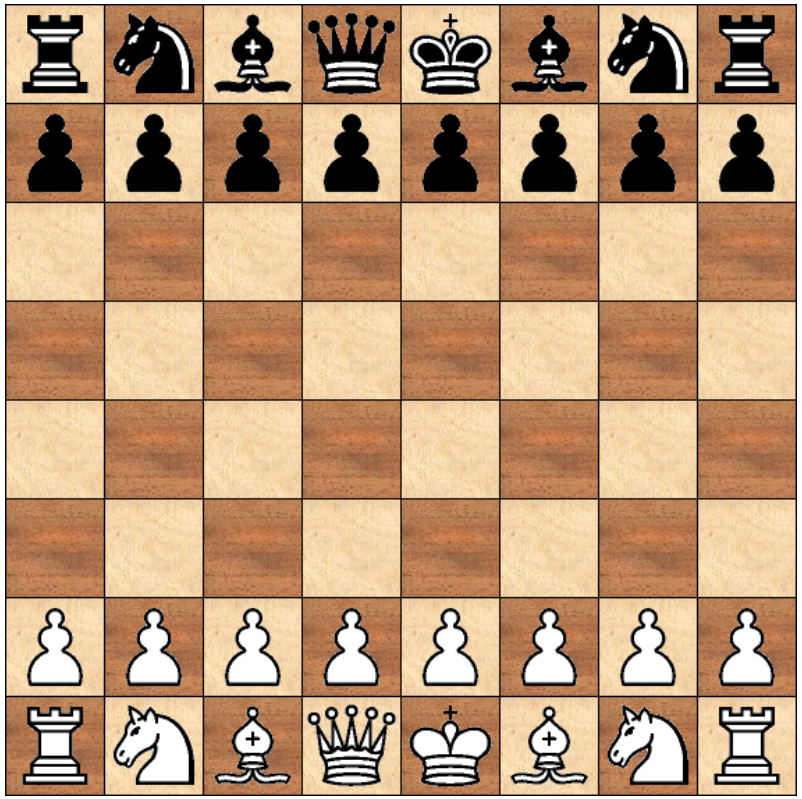
Axelle De Brito, Louise Lallemand et Grégoire Guillot

Introduction

I. Modélisation du jeu d'échecs

II. Algorithme MinMax

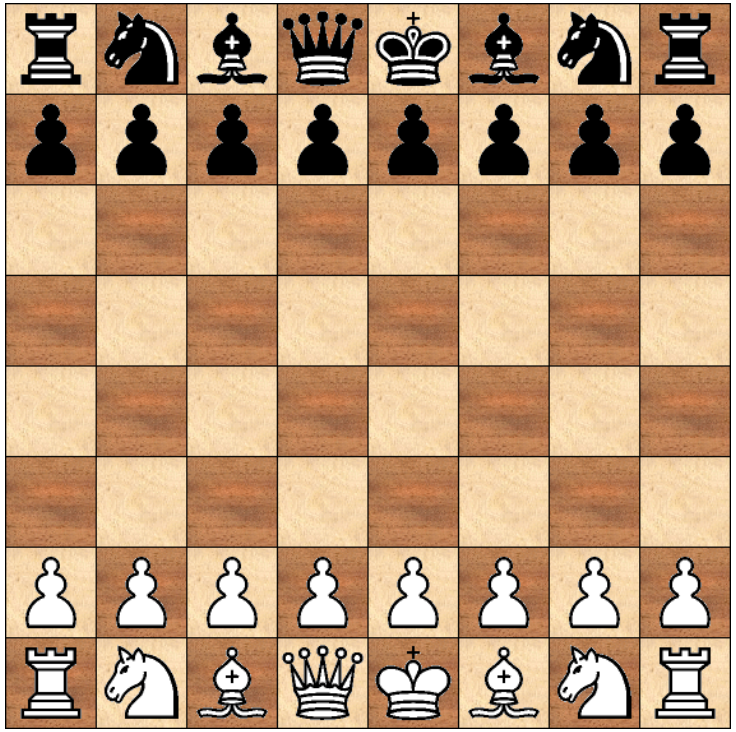
III. Résultats



Modélisation du jeu d'échecs

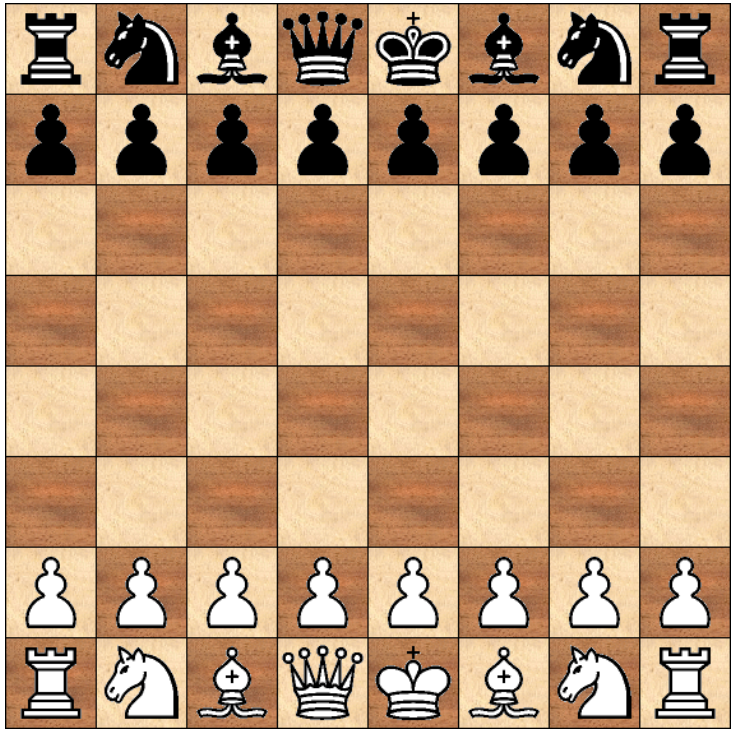
L’échiquier

- int taille
- Piece ** plateau
- Piece *roi_noir
- Piece *roi_blanc
- list<Coup> L_coup_depuis_dep



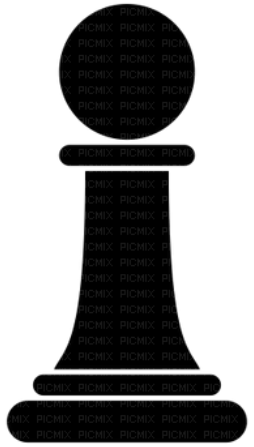
L’échiquier

- Echiquier(int n)
- ~Echiquier()
- affiche()
- mise_en_place_echec_piece(Echiquier & Echi)



Les types de pièces

- string type
- list<Deplac_rel> deplac_relatif
- int valeur

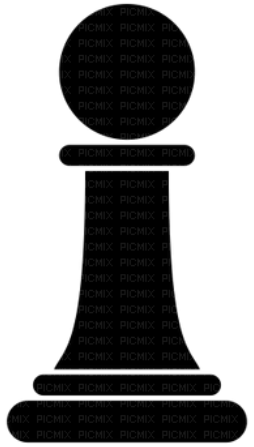


- TypePiece(string type)
- ~TypePiece()

Les pièces

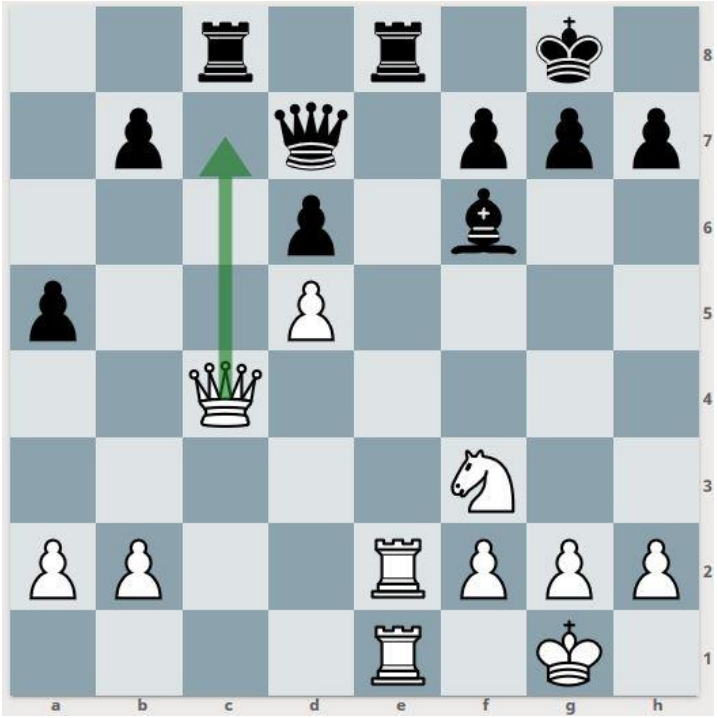
- bool isWhite
- int a_bouge
- pair<int,int> position_coor

- Piece(bool isWhit, int a_bouge, pair<int,int> coor, string typee)
- Piece(const Piece &)
- ~Piece()



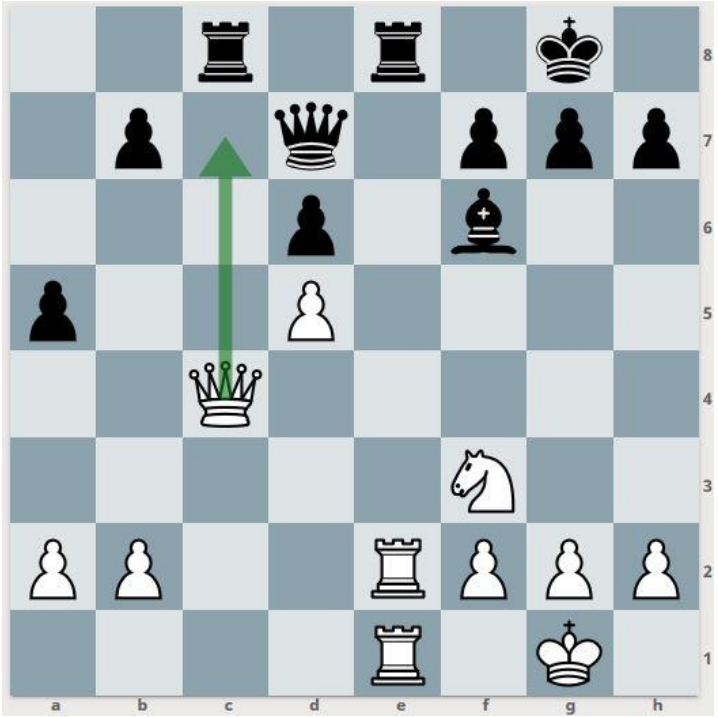
Les coups

- bool isWhite
- Piece pieceJouee
- pair<int,int> oldPosition
- pair<int,int> newPosition
- Piece *Taken
- Coup_special CoupSpecial
- string Type_Promu
- bool is_echec
- bool is_mat
- int num_tour_de_jeu



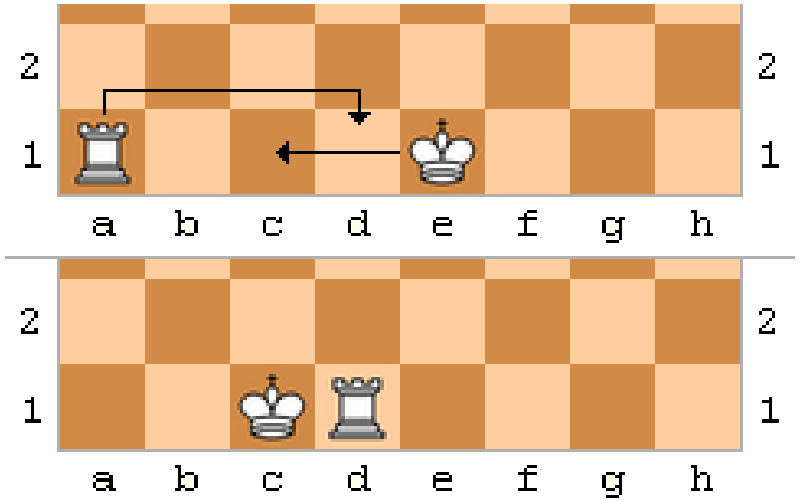
Les coups

- Coup(bool isW, const Piece &, pair<int,int> newP, pair<int,int> oldP, int num_tour_de_jeu, Piece *taken, CoupSpecial coup_special, string type_promu, bool is_echec, bool is_mat)
- Coup(const Coup &)
- ~Coup()



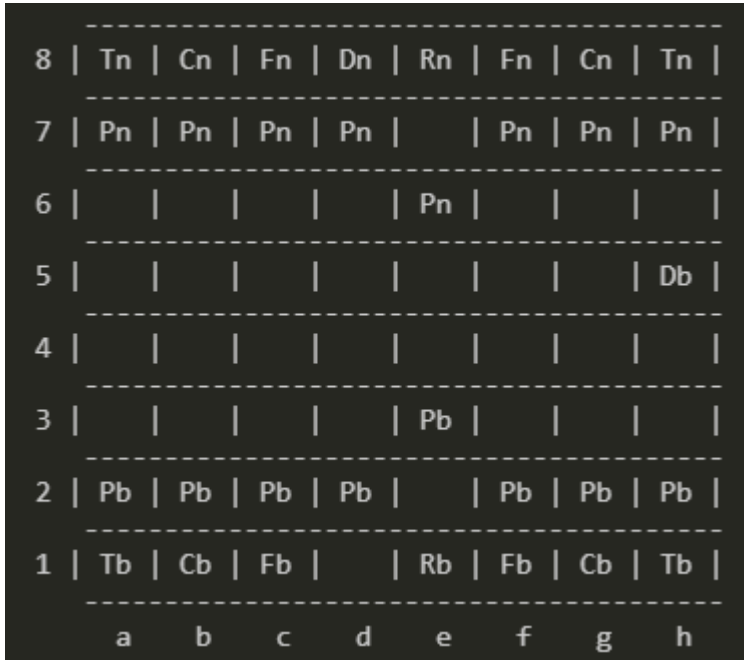
Les coups spéciaux

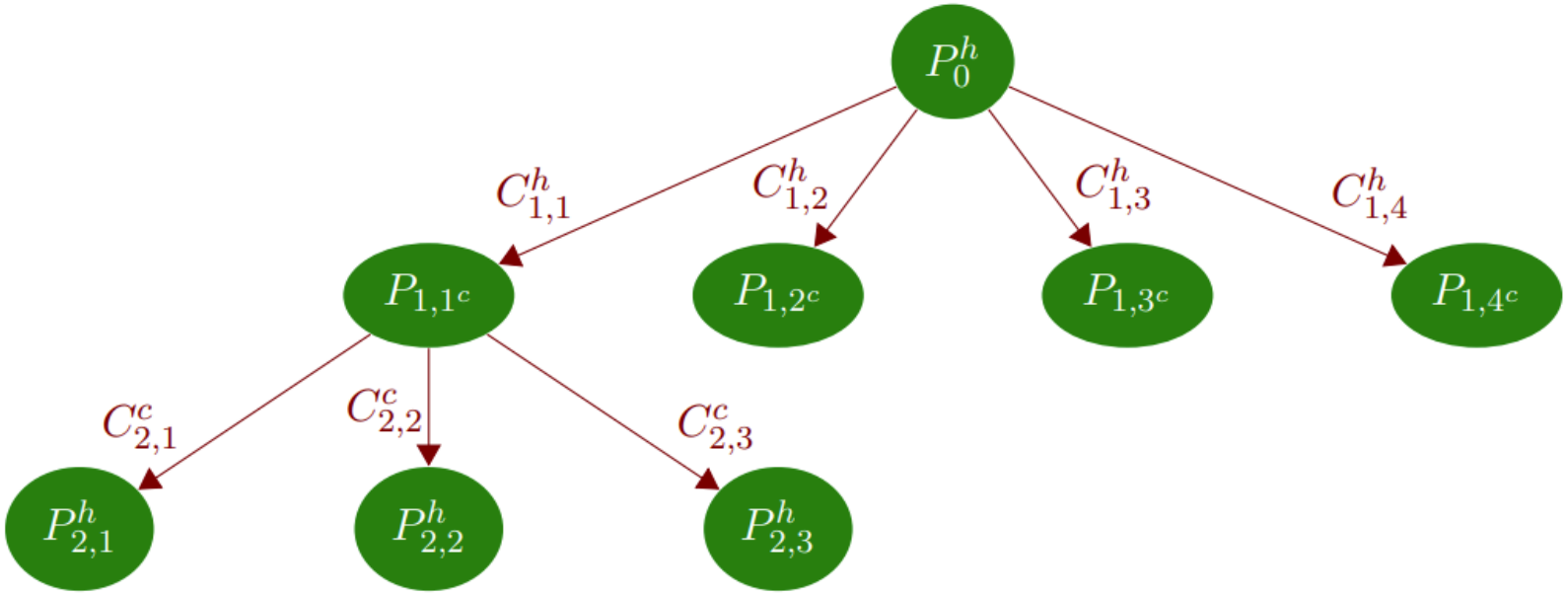
- petit roque
- grand roque
- promotion
- prise en passant



Déroulement d’une partie

- Choix du jeu
- Choix de l’adversaire
- Interface graphique du terminal
- Déroulé du jeu

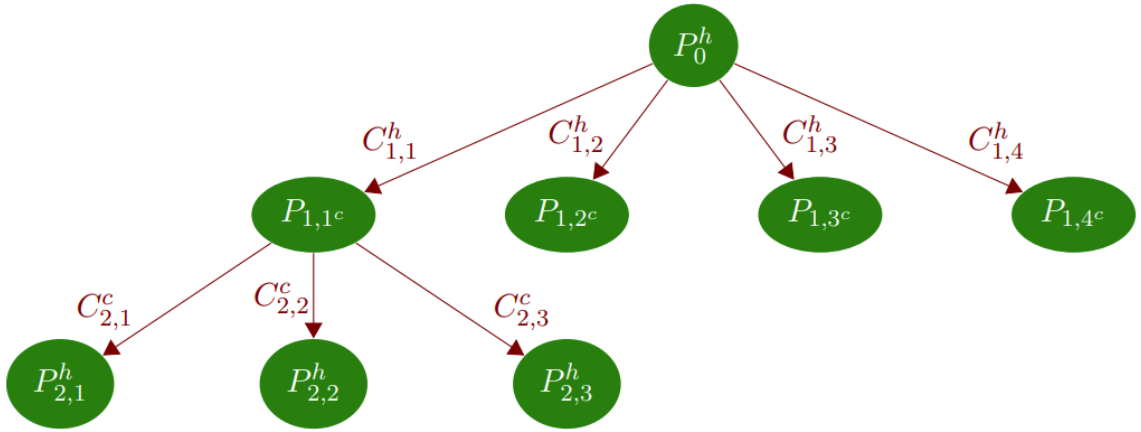




II. Algorithme MinMax

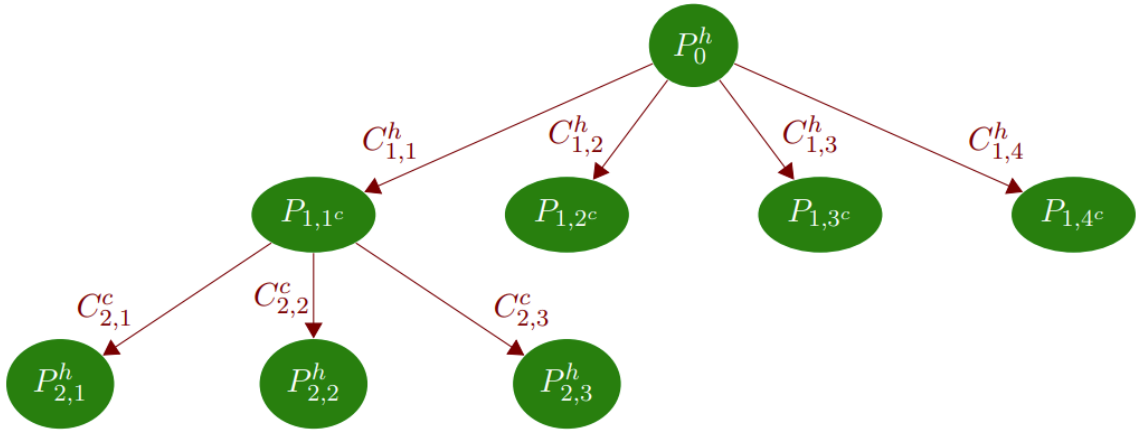
Principe de l’algorithme

- Echiquier
- Pointeur vers position fille
- Pointeur vers position soeur



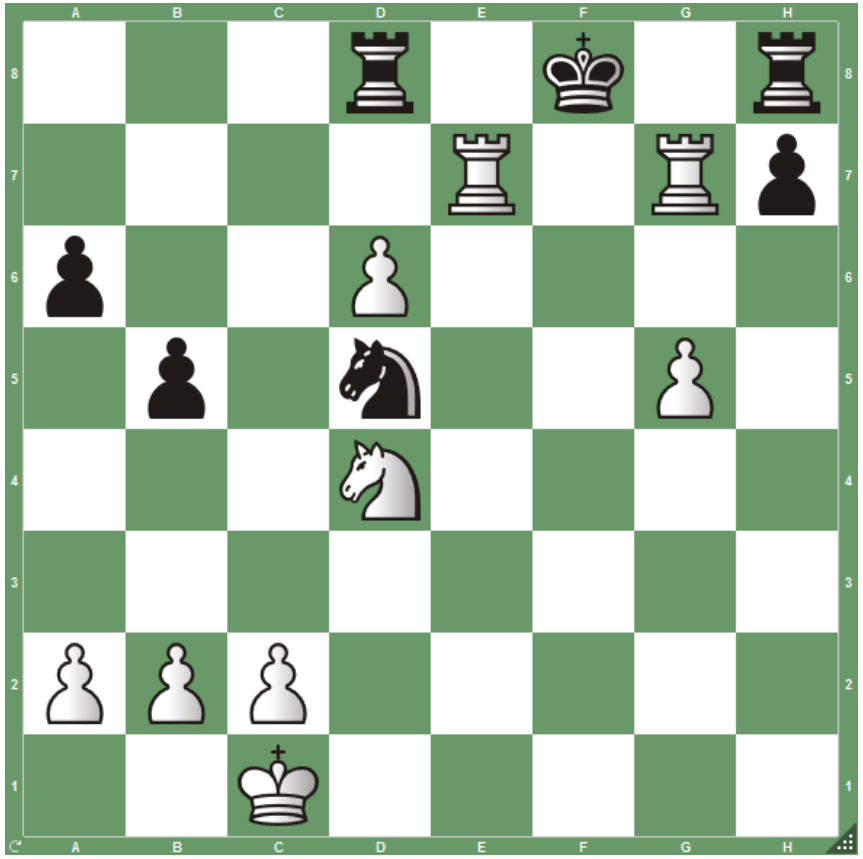
Principe de l’algorithme

- Si la position est au plus bas niveau de profondeur, on retourne sa valeur propre
- Sinon, on retourne le maximum ou le minimum de ses valeurs filles en fonction du joueur concerné
- Choix du coup avec coup_min_max



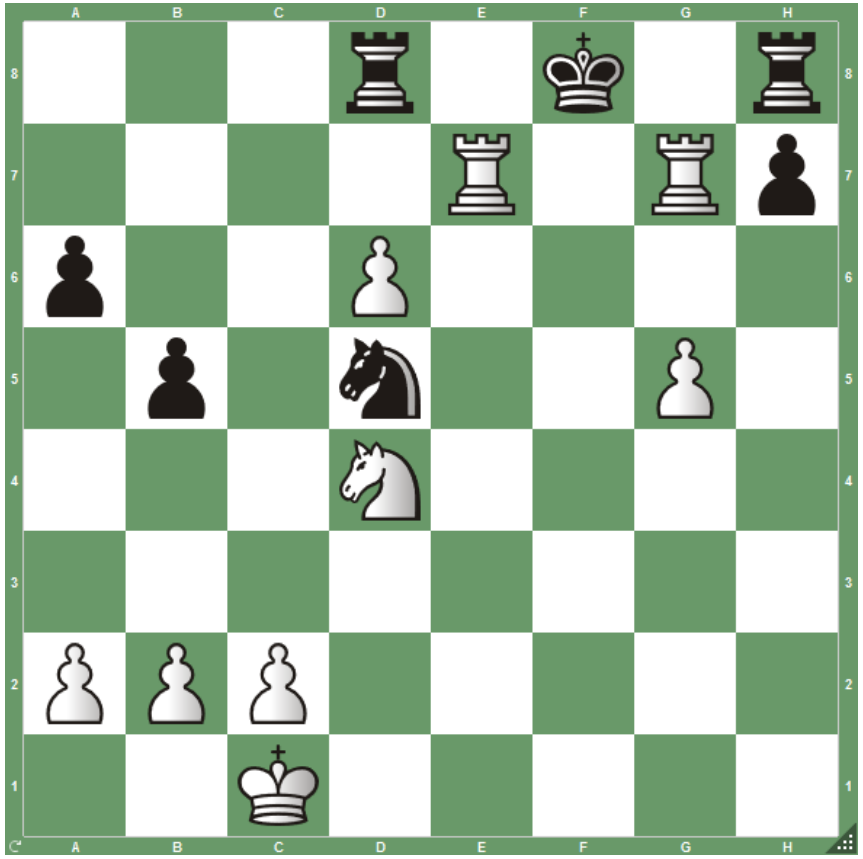
La classe Position

- Echiquier *plateauRef
- list<Coup> CoupsPrecedents
- Position *Sœur
- Position *Fille
- bool joueur_current
- bool joueur
- int num_tour_de_jeu



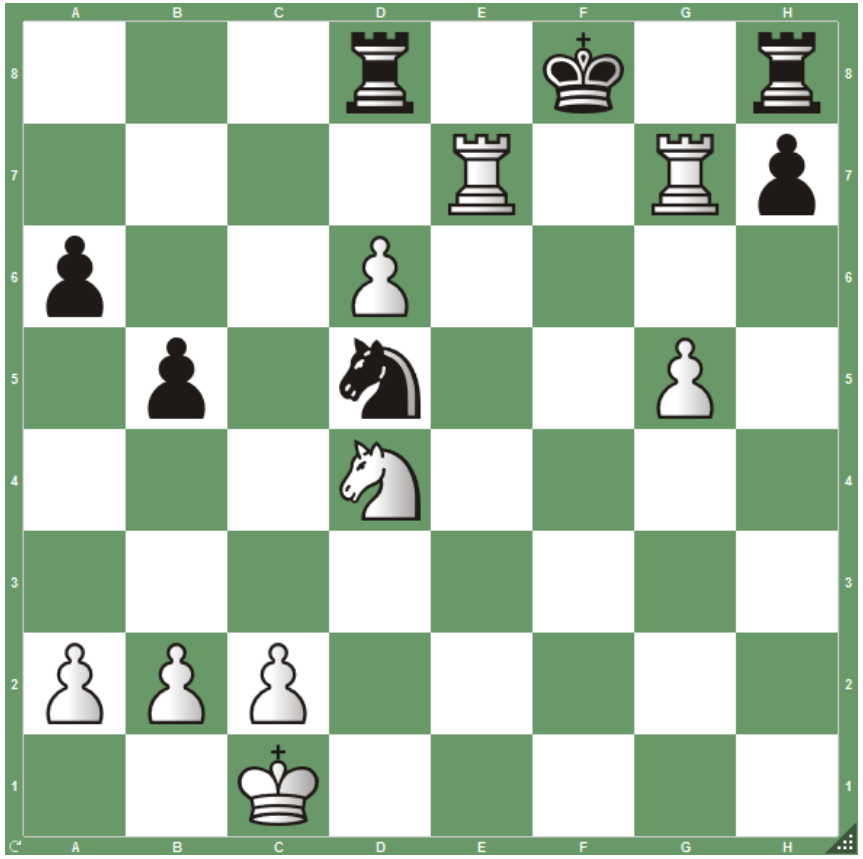
La classe Position

- Position(Echiquier *, list<Coup> coups, Position * Sœur, Position* Fille, bool joueurCoup, int num_tour)
- void set_valeur(int gammap, int gammad)
- void générateur(int profondeur)
- ~Position()



La classe Position

- actualisePlateau(Echiquier &plateau, const list<Coup> &coupsPrecedents)
- resetPlateau(Echiquier &plateau, const list<Coup> & coupsPrecedents)



Génération des états possibles

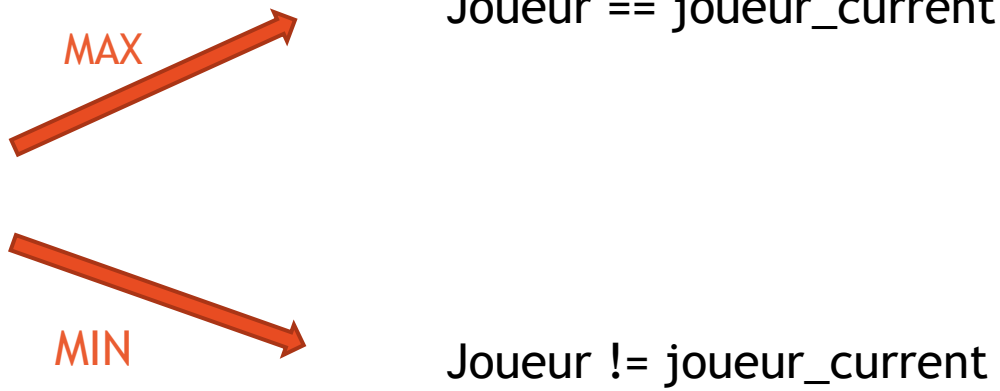
Conditions d'arrêt :

- Profondeur limite atteinte
- Pas de coup possible partant de la position
- La position est gagnante ou perdante

 set_valeur

Génération des états possibles

- new Position
- generateur(profondeur - 1)
- valeurMinMax



Evaluation d’un état

- Evaluation par la fonction générateur
- Si position terminale : valeur propre de la position

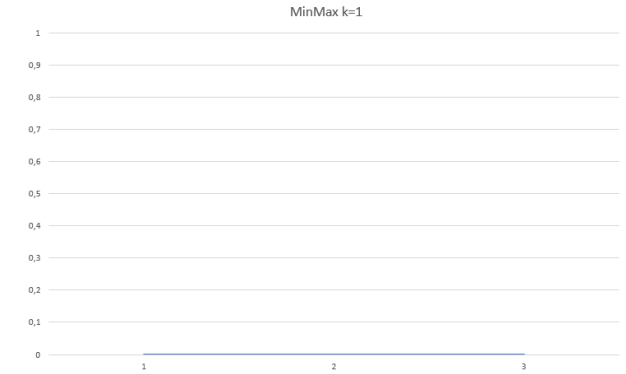
$$\gamma_p(\text{valeur}(\text{ordi}) - \text{valeur}(\text{humain})) + \gamma_c(\text{cont}(\text{ordi}) - \text{cont}(\text{humain}))$$

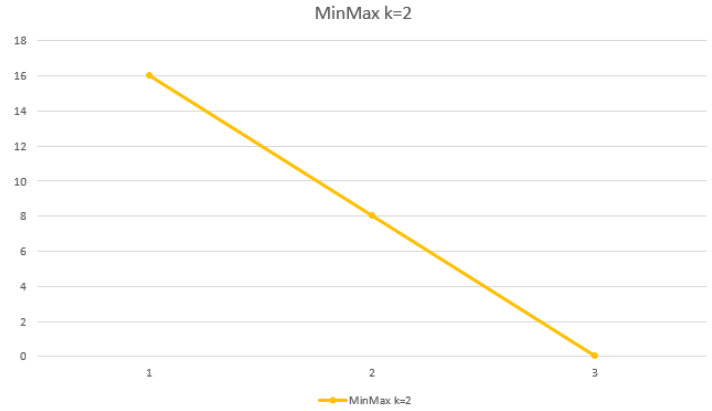
Résultats

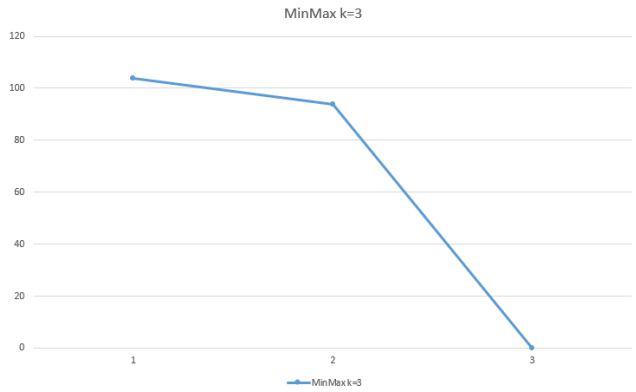
Résultats sur le morpion

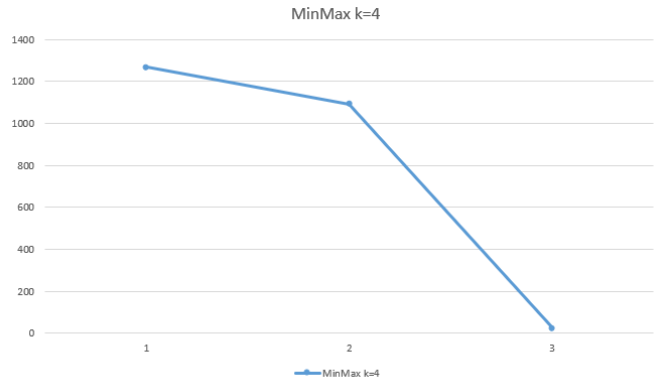
	Aléatore	MinMax *k=1*	MinMax *k=2*	MinMax *k=3*	MinMax *k=4*
Aléatore	11	42	72	72	68
	0	0	12	17	13
	89	58	16	13	19
MinMax k=1	0	0	0	0	100
	0	0	0	0	0
	100	100	100	100	0
MinMax k=2	1	0	0	0	100
	8	0	0	0	0
	91	100	100	100	0
MinMax k=3	1	0	0	0	0
	5	0	0	0	100
	94	100	100	100	0
MinMax k=4	4	0	0	0	0
	0	0	0	0	100
	96	100	100	100	0

Résultats en temps sur le morpion

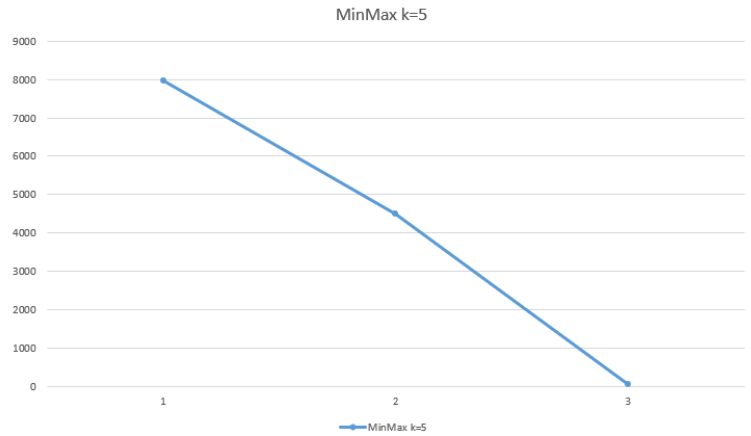


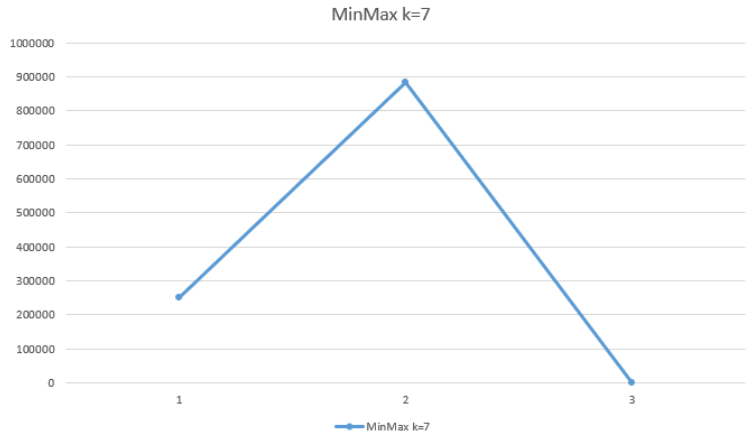


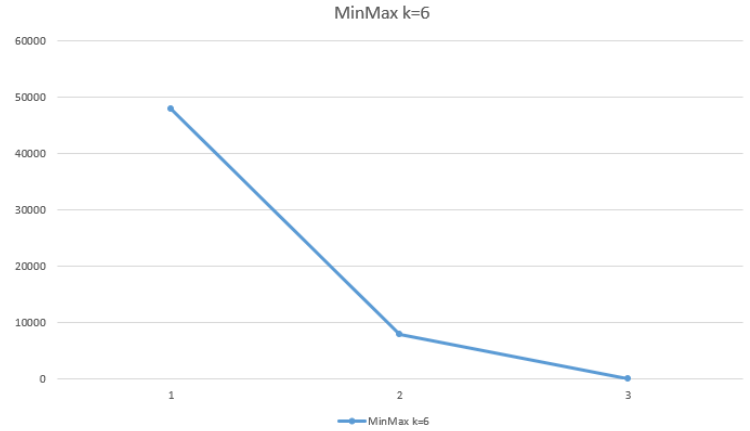


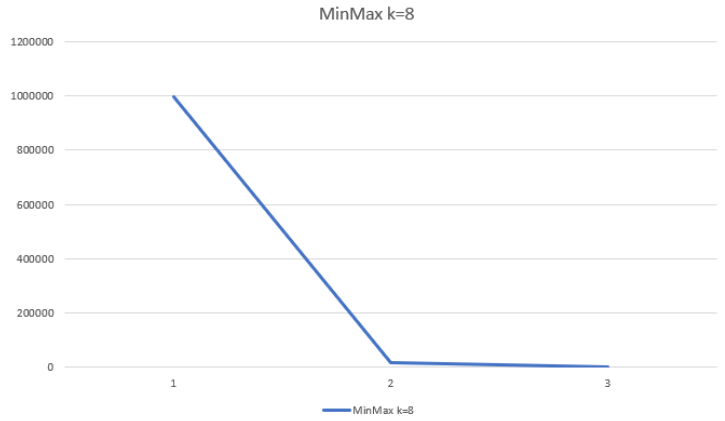


Résultats en temps sur le morpion

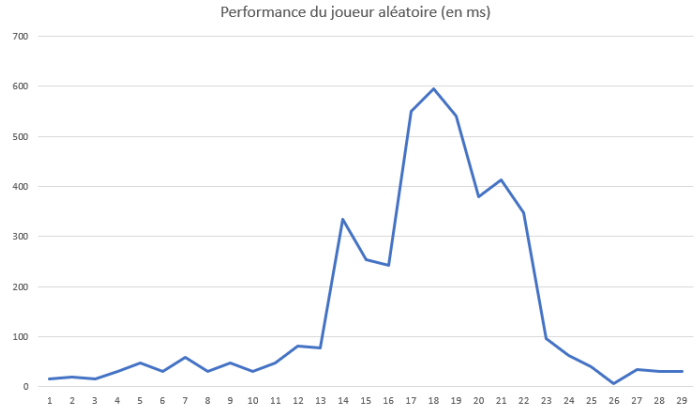


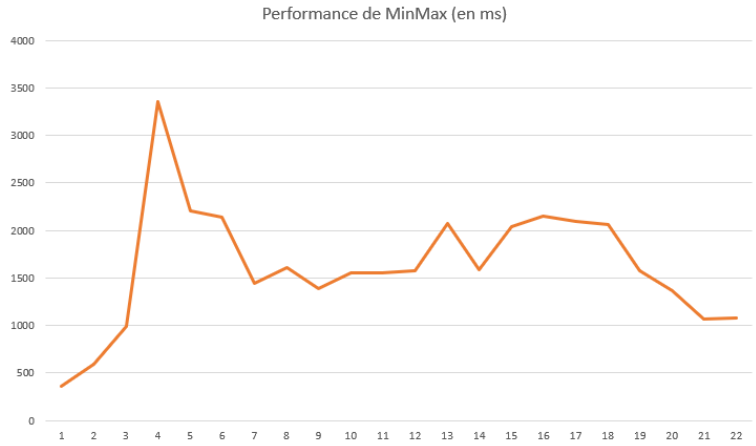


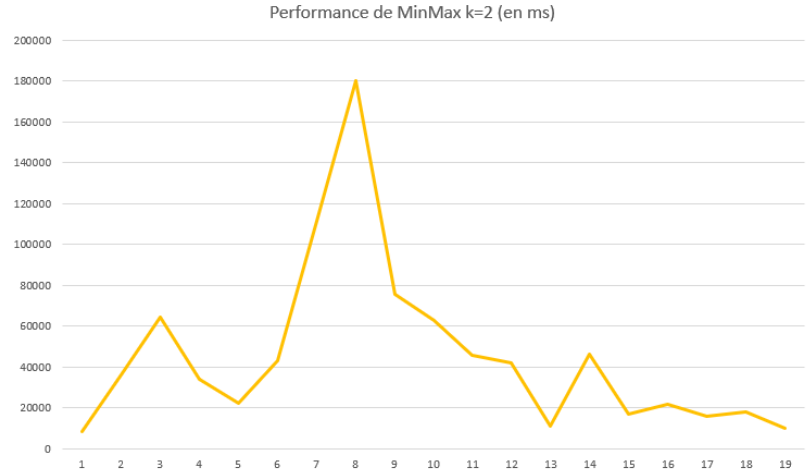




Performance en temps et en victoire sur les échecs







Prise de recul : les limites de notre simulation