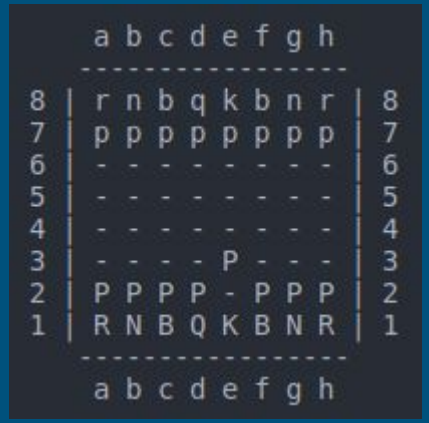
Audit Pdp Chess



Rossignon Morgan
Daniel Karl
Beites Marvin
Zucchelli Thomas
Salomode Florian

INTRODUCTION

Objectif : Programmer un joueur d’échecs complet.

- Application des algorithmes d’exploration d’arbre classiques.
- Développement d’heuristiques propres au jeu.
- Développement d’un moteur de jeu stable.
- Optimisation des performances du moteur.
- Interface utilisateur en mode texte.

POSITION DE PDP CHESS PARMI L'EXISTANT

Logiciels de jeux d'échecs disponibles dans les boutiques virtuelles permettant de :

- Sélectionner des paramètres (joueur qui commence, nature des participants humains ou IAs, la difficulté de l'IA)
- Visualiser les mouvements d'une pièce.
- Sauvegarder ou charger une partie.

POSITION DE PDP CHESS PARMI L'EXISTANT

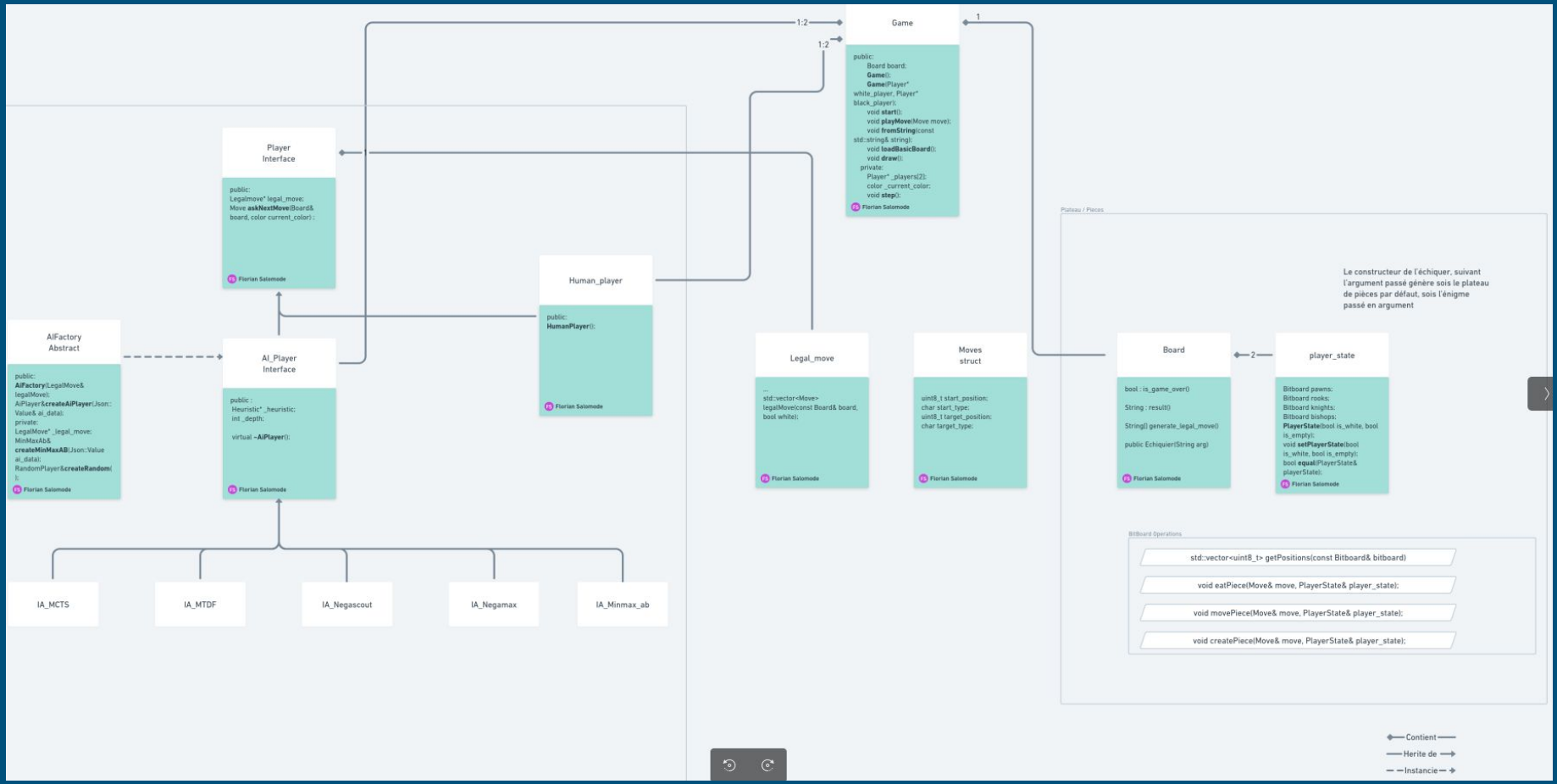
Logiciels portant davantage sur le domaine scientifique :

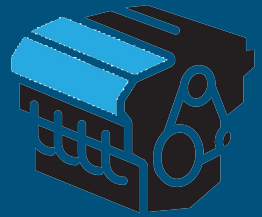
- Kaissa (1974)
- Deep Thought (1989)
- Deep Blue (1996)

LISTE DES BESOINS PRINCIPAUX

- Créer/ représenter le plateau de jeu
- Lister les mouvements possibles
- Déplacer une pièce
- Garder une trace de l’historique
- Choisir les algorithmes à utiliser

DIAGRAMME DU PROJET ACTUEL





MOTEUR DE JEU

REPRÉSENTER LE PLATEAU DE JEU : BITBOARDS

56	57	58	59	60	61	62	63
48	49	50	51	52	53	54	55
40	41	42	43	44	45	46	47
32	33	34	35	36	37	38	39
24	25	26	27	28	29	30	31
16	17	18	19	20	21	22	23
8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7

Indexation des cases du plateau afin d'utiliser des bitboards :

a1->0 a2 -> 8 …

Grille de 64 cases, soit le nombre de bits d'un int64_t.

REPRÉSENTER LE PLATEAU DE JEU: BITBOARDS, ZOOM SUR LES PIONS

Position des pions blanc au début d'une partie:

uint64_t pawns =
(1<<8)+(1<<9)+(1<<10)+(1<<11)+(1<<12)+(1<<13)+(1<<14)+(1<<15);

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0
Valeur totale du bitboard :
65280
```

Un bitboard pour chaque pièce.

Permet des opérations simple et légères.

LISTER LES MOUVEMENTS POSSIBLES

pawnsMove =

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
Valeur totale du bitboard :
134742016
```

pawsAttacks =

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 1 0 1 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
Valeur totale du bitboard :
1310720
```

pawnsMove | pawnsAttacks =

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 1 1 1 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
Valeur totale du bitboard :
136052736
```

Création des LookupTable pour les pièces non “glissantes” (pion,…).

Application d’opérations logiques entre bitboards.

LISTER LES MOUVEMENTS POSSIBLES : ZOOM SUR LE PION

```
void LegalMove::pawnsLegalMoves(const Board &board, bool color, std::vector<Move> &moves) {
    uint64_t movable;
    uint64_t eatable;
    uint64_t legal_positions;
    Bitboard bitboard = board._pieces[color]->pawns;

    for (int current_piece_position: getPositionsV2(bitboard.value)) {
        uint64_t move_target_positions = _pawns_moves_table[color][current_piece_position];
        uint64_t blocked_positions =
                move_target_positions & (board._pieces[!color]->all.value | board._pieces[color]->all.value);

        movable = move_target_positions - blocked_positions;
        eatable = _pawns_attacks_table[color][current_piece_position] & board._pieces[!color]->all.value;

        legal_positions = eatable | movable;

        if (legal_positions == 0) {
            continue;
        }

        generateMoves(bitboard, current_piece_position, moves, legal_positions);
    }
}
```

ARRÊTER UNE PARTIE

- Match nul :
  - Répétition
  - Composition particulière (Roi contre roi etc..)

- Défaite :
  - Mort du roi
  - Abandon (Pour un joueur humain uniquement)

OPTIMISATION DU MOTEUR DE JEU

Tire partie de l’utilisation des bitboards :

- Pour la génération des mouvements légaux
  - Version générant les coups à la volée (V1)
  - Version utilisant des lookup tables pour les pièces non glissantes (V2)

- Pour la recherche des positions des pièces
  - Recherche exhaustive (V1)
  - Recherche par soustraction des bits les plus à droite(V2)

COMPARAISON DES RÉSULTATS

Résultats sur deux joueurs AlphaBeta de profondeur 3 et sur 200 parties

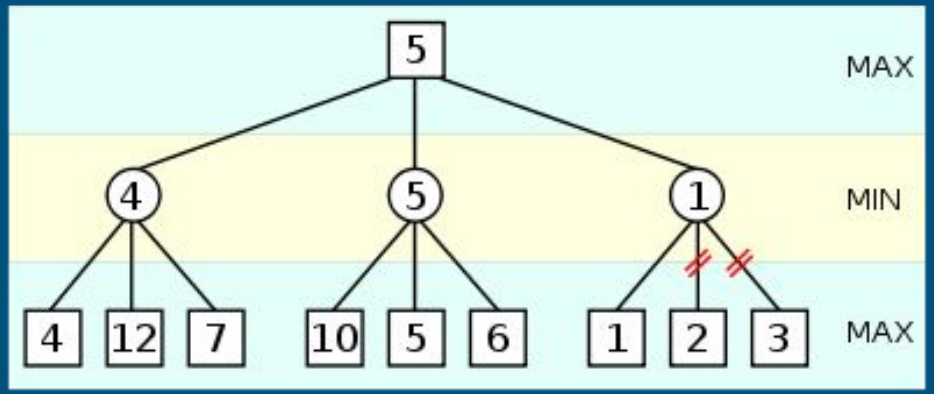
	Mouvements Légaux (V1)	Mouvements Légaux (V2)
Recherche d'index (V1)	200 game played Average game time 1714 ms	200 game played Average game time 2237 ms
Recherche d'index (V2)	200 game played Average game time 1663 ms	200 game played Average game time 2027 ms

PARTIE ALGORITHMIQUE

PRESENTATION DES DIFFERENTS ALGORITHMES

- Minmax Alpha-Bêta.
- Negamax.
- Negascout.
- MTD(f).
- Monte-Carlo Tree Search (MCTS).

ZOOM SUR L'ALGORITHME ALPHA BETA



HEURISTIQUES UTILISABLES

Heuristique modulable inspirée de l'heuristique de Shannon.

- Attribution d'une valeur pour chaque pièce du plateau.

- Prise en compte des éléments suivants :

  - pions doublés.
  - pions arriérés.
  - pions avancés.
  - pions isolés.
  - mouvements possibles des pièces.

ZOOM SUR L'HEURISTIQUE PERSONNALISABLE

```
{
  "White" : {
    "Type" : "AlphaBeta",
    "Depth" : 3,
    "Heuristic" : {
      "LegalMove" : 2,
      "ForwardPawn" : 1
    }
  },

  "Black" : {
    "Type" : "AlphaBeta",
    "Depth" : 3,
    "Heuristic" : {
      "LegalMove" : 2,
      "ForwardPawn" : 1
    }
  }
}
```

EXEMPLE D'UN MATCH ENTRE ALGORITHMES

Fin de match entre MinMax ab Blanc de profondeur 3 contre un MinMax ab Noir de profondeur 2 :

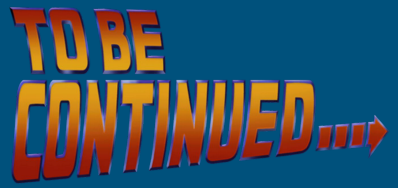
```
Meilleur coup : 3965
Player : White

    a b c d e f g h
   -----------------
8 | - - b - B - - - | 8
7 | p - - p - - - p | 7
6 | - - p - - - - - | 6
5 | - r - - P - - - | 5
4 | - p - - - - - - | 4
3 | - - - K P N - - | 3
2 | P - - - N q P P | 2
1 | - - R - - - - - | 1
   -----------------
    a b c d e f g h

Game won by white in 49 moves
```

LISTE DES ÉLÉMENTS À FINALISER

- Implémentation de Negascout, MTD, Monte-Carlo tree search.
- Implémentation du joueur humain.
- Sauvegarde d'une partie dans un fichier csv afin d'en générer des statistiques.
- Ajout des coups spéciaux.
- Généraliser l'utilisation des fichiers json



MERCI!