Quarto + : Rendu de la première partie du projet

Nicolas Devatine et Julien Guyot

Lundi 5 Mars 2016

1 Question 1

Note : Cette réponse est bien plus longue que ce qui est a priori demandé, néanmoins nous avons fait le choix de la laisser, afin d'être le plus clair possible, sur les "pourquoi" et "comment ?" de ce projet d'implémentation.

Il y a néanmoins un bref résumé de la réponse ici si vous préférez ne pas tout lire.

Nous avons pris le parti de la performance : Utiliser des types/classes, ainsi que des opérateurs les plus simples possible — Du point de vue machine — afin que les performances soient les meilleures possible.

1.1 Modélisation du plateau :

Il y a seize pièces, donc "l'identifiant" de chaque pièce peut être codé sur 4 bits. Il y a 16 cases, donc on peut modliser un plateau comme un long (de 16*4 = 64 bits), que nous appellerons "plateau".
Nous aurons aussi deux short (16b): Le premier, "S" donne les cases qui sont occupées par une pièce, et enfin l'autre donne les pièces qui ont déjà été posées "P".
Nous aurons aussi un byte (8b), qu'on appellera "B", qui donne le stade du tour dans lequel on est (i.e "quel joueur joue quel type de coup?"), et la pièce qu'il faut jouer, le cas échéant.

1.2 Modélisation des pièces :

Par conséquent, pour modéliser une pièce sur 4 bits, on se donne le codage suivant :

- le premier bit (de poids fort) comme donnant la couleur de la pièce (0 = noir, 1 = blanc)
- le second bit représente la hauteur (0 = bas, 1 = haut)
- le troisième le sommet (0 = troué, 1 = plein)
- le dernier la forme (0 = cylindrique, 1 = parallepipedrique)

1.3 Modélisation d'une coordonnée :

Les cases sont répertoriées par une lettre + un entier. On aura, du coup, la partie "lettre" qui adresse les deux bits de poids fort, la partie "chiffre" qui adresseront les bits de poids faible.

Ainsi, on aura des coordonnées de codée sur quatre bits, qui sera la suivante [1]

$$coord = (id\_lettre << 2 \quad (id\_chiffre))$$

On se donne A=0, B=1, C=2, D=3.

1.4 Modélisation de l'état du jeu

Pour modéliser l'état de jeu (qui est en train de jouer/quel type de coup doit-il jouer ?), on se donne un code.
Il peut y avoir quatre "états de jeu" (Noir/Blanc donne une pièce/Joue), qui seront représentés par les deux bits les plus à gauche de B.
On se donne $donner\_piece = 0, jouer = 1$. On aura :

$$B = id\_joueur \quad id\_action \quad 00 \quad id\_piece$$

Où id_piece est l'identifiant de la pièce à jouer si une pièce donnée doit être jouée (dans le cas contraire, on considèrera que id_piece n'est pas significatif), et 00 sont deux bits de remplissage.

Par exemple, si le joueur noir doit jouer une pièce qui vient d'être donnée par un autre joueur, id_joueur = 0 et id_piece = 1.

1.5 Modélisation d'un dépôt de pièce :

Le coup est codé sur un byte (8b).

$$coup = (coord << 4) \quad code\_piece$$

Les 4 bits le plus à gauche seront liés aux coordonnées du plateau, ceux à droite dénotent à la pièce.

Par exemple, quand on pose la pièce noire, haute, trouée et parallepipédrique sur la case B4, le coup sera calculé de la manière suivante:

- Le code de la pièce est $0101_2 = 5_{10}$
- Le code des coordonnées est $0111_2 = 7_{10}$
- Par conséquent, le coup sera codé par 01110101_2

[1] Durant ce texte, nous utiliserons les notations $A << B$ (resp. $>>$) pour le décalage binaire à gauche (respectivement à droite) de l'entier A de B bits. Les opérations logiques sont les opérateurs bit à bit.

1.6 Modélisation d'un don de pièce :

Même chose que le dépôt de pièce, mis à part que la partie "coordonnées" sera à 0.

1.7 Ajout d'une pièce sur un plateau :

Pour les pièces restantes, même s'il est possible de recalculer si elles ont été posées, on prend P, contenant des 0 quand aucune pièce n'a été posée, des 1 quand toutes les pièces ont été posées. Alors que la 80e minute tait passe et que le XV de France venait de gratter le ballon au terme d'un combat alletant, le demi d'ouverture a voulu trop allonger son coup de pied en touche, permettant aux anglais de remettre la France sous pression. Un coup de pied finalement anecdotique au coup de sifflet final mais qui aurait pu coter très cher. '

Par exemple, quand on voudra poser la pièce noire haute et trouée, on va regarder si le bit 5 de P est à 0 (ie si elle n'a pas déjà été jouée).
Puis on regarde si le bit 7 de S est à 0 (c'est à dire si la case n'est pas déjà prise).

Si non, on va faire l'opération suivante :

$$plateau = OR(plateau, id\_piece << (4 \times coordonnee))$$

1.8 Test des caractéristiques en commun entre plusieurs pièces

Ainsi, pour voir si deux pièces (p1, p2) ont quelque chose en commun, il faudrait que $NOT(XOR(p1, p2)) = 1$ pour un bit donné.
Pour voir si quatre pièces ont quelque chose en commun, on teste

$$c = AND(NOT(XOR(p1, p2)), NOT(XOR(p3, p4)))$$

On récupère ainsi les caractéristiques communes des quatre pièces, là ou les bits de c sont à 1.

Pour trois pièces, on fait le même calcul, avec $p4 = p1$

1.9 Résumé

Le plateau sera modélisé par quatre attributs :

- Le plateau de jeu en lui même sera modélisé par un long (64 bits).
- Un répertoire des cases libres : modélisé par un short (16 bits).
- Un répertoire des pièces jouées: aussi modélisé par un short.
- La pièce à jouer : qui est modélisée par un byte (8 bits).

Les coups seront modélisés par un byte, contenant soit un identificateur de pièce (dans le cas d'un don), soit un identificateur de pièce plus une coordonnée.

Cette représentation possède l'avantage d'être très compacte (en principe une douzaine d'octets par plateau de jeu, seulement trois bits qui sont "inutiles"), ainsi que de faire appel à des actions élémentaires au niveau machine (des entiers + opérateurs logiques/de décalage).

Par conséquent, elle sera a priori très rapide.
La critique que l'on peut formuler est que cette modélisation n'est a priori pas évidente à implémenter du point de vue humain, quoique loin d'être impossible.

Néanmoins, nous avons fait en sorte de déterminer, et expliquer les opérations possibles sur le plateau.

2 Question 2.

Dans un premier temps, on teste si il y a un carré /une diagonale/ligne/colonne qui comporte des pièces ayant des caractéristiques en commun.

Ensuite, on regarde s'il reste des pièces/des cases libres (les deux sont équivalents).

3 Question 3

La principale source de difficulté est le fait qu'il y a deux "types" de coups à jouer, ce qui pose un problème, étant donné qu'il va falloir différencier l'appel lors de la recherche du meilleur coup en fonction du type de coup à jouer. Cela vaut notamment pour l'heuristique.

4 Question 4

Il existe bien des coups imparables, mais qui sont accessibles seulement dans des configurations très particulières de jeu, qui se trouvent plutôt en fin de partie.

Le principal coup imparable qui existe est le coup qui forcant l'adversaire à jouer sur un emplacement précis, et dans lequel ce coup mènerait dans une configuration de jeu où la victoire nous est assurée au vu du choix de pièce qu'il a à nous donner.

Par exemple, un coup imparable pourrait être de forcer l'adversaire à jouer un coup pour bloquer un coup qui nous donnerait gagnant, mais au vu de la pièce qu'on lui a donné, jouer ce coup ouvrirait sur un nouveau coup gagnant qu'il ne pourrait pas non plus éviter avec le choix de pièce qu'il a à nous donner (cela veut dire que les pièces restantes ont toutes au moins deux caractéristiques en commun qui permettent la victoire dans les deux configurations).

Il existe donc bien des coups imparables mais ils sont compliqués à mettre en place car ils demandent des configurations de plateau très particulières et un ensemble de pièces restantes très précis, plutôt accessible en fin de partie.

5 Question 5

Les critères que nous envisageons de prendre en compte pour la conception de nos heuristiques sont les suivants :

- Les caractéristiques et des pièces restantes et leur nombre.
- Le positionnement des pièces sur le plateau par rapport aux configurations gagnantes.
- Le nombre de caractéristiques communes entre les pièces côte à côté les unes des autres sur le plateau (horizontale, verticale ou diagonale).

6 Question 6

Adopter une stratégie particulière en début et milieu de partie semble compliqué étant donné que le joueur adverse choisit les pièces que l'on doit jouer, et que le choix des pièces est grand, tenir une stratégie avec ce facteur trop aléatoire est donc difficile.

Cependant en fin de partie le nombre de pièces restantes est réduit et le plateau dispose de moins d'emplacements libres, il est donc plus facile de mettre en place une stratégie adapté à la configuration du jeu qui nous permettrait de remporter la partie.

Il est donc plus souhaitable d'adopter une stratégie particulière en fin de partie.

7 Question 7

Le plateau ayant une taille fixe de 16 cases, il y aura 16 pièces qui seront posées durant la partie. Sachant que le tour d'un joueur se compose de deux coups, le premier étant de jouer sa pièce et le second de choisir la pièce que l'adversaire va jouer, il y aura un total de 2*16 coups dans une partie, soit 32 coups.

Pour respecter la contrainte de temps, nous comptons mettre en place différentes techniques dajà vues en cours — Ainsi que celles que nous verrons dans le futur —

Au niveau de l'algorithme, nous utiliserons a minima ID alphabeta, ou encore MTD.