

IA et contraintes - Devoir : Escampe

Modalités pratiques : L'objectif de ce devoir est de modéliser et réaliser un joueur artificiel capable de jouer à un jeu à deux joueurs. Le programme que vous devez réaliser prendra part, à l'issue de ce cours, à un tournoi qui opposera les différents joueurs réalisés. Pour cette raison il vous est demandé de bien respecter les spécifications et les consignes.

L'intégralité du projet est à faire **en binôme**.

Si vous trouvez des références ou des programmes permettant de jouer à ce jeu, vous pouvez naturellement vous en inspirer (sans pour autant recopier le code), mais vous devez très clairement indiquer dans le rapport de votre devoir les sources et la bibliographie que vous utilisez.

Présentation

Escampe est un jeu qui se joue sur un plateau de 36 cases (6X6) qui sont marquées par des cercles portant un liseré simple, double ou triple (cf. figure 1). Chaque joueur dispose de 6 pièces (noires ou blanches) : une licorne et 5 paladins (cf. figure 2).

Les lignes du plateau sont numérotées de 1 à 6, et les colonnes sont nommées A à F. Chaque case est ainsi identifiée par un couple chiffre-lettre. Chaque joueur est identifié par une couleur (blanc, noir).

Le but du jeu est de prendre la licorne de l'adversaire.

Les règles du jeu

Une caractéristique importante d'Escampe est que le choix de la pièce à jouer est directement contraint par le coup précédent de l'adversaire.

1. Le joueur Noir choisit un bord du plateau (haut ou bas), et place comme il le souhaite ses pièces sur les deux premières lignes de ce bord.
2. Le joueur Blanc pose ensuite ses propres pièces sur les deux premières lignes du bord opposé. C'est lui qui jouera en premier, en choisissant librement la pièce qu'il désire déplacer.
3. Pendant le reste de la partie, chaque joueur peut à son tour bouger sa licorne ou l'un de ses paladins, en devant toutefois respecter la consigne suivante : la pièce jouée doit partir d'une case ayant le même liseré que celle sur laquelle l'autre joueur a posé sa propre pièce au tour précédent.
4. En outre le liseré du cercle de départ détermine le nombre de cases que la pièce peut parcourir : une case si le liseré est simple, deux cases s'il est double, et trois cases s'il est triple. Lors d'un déplacement il est interdit de faire passer la pièce par une case occupée, ou de la faire passer deux fois par une même case. Les mouvements en diagonal ne sont pas autorisés.
5. Si un joueur ne peut bouger aucune pièce, il saute son tour. L'autre joueur peut alors jouer la pièce de son choix.

Fin de partie

La partie se termine quand l'un des joueurs prend la licorne de son adversaire avec l'un de ses paladins (il s'agit du seul type de prise possible, les paladins en particulier étant imprenables).

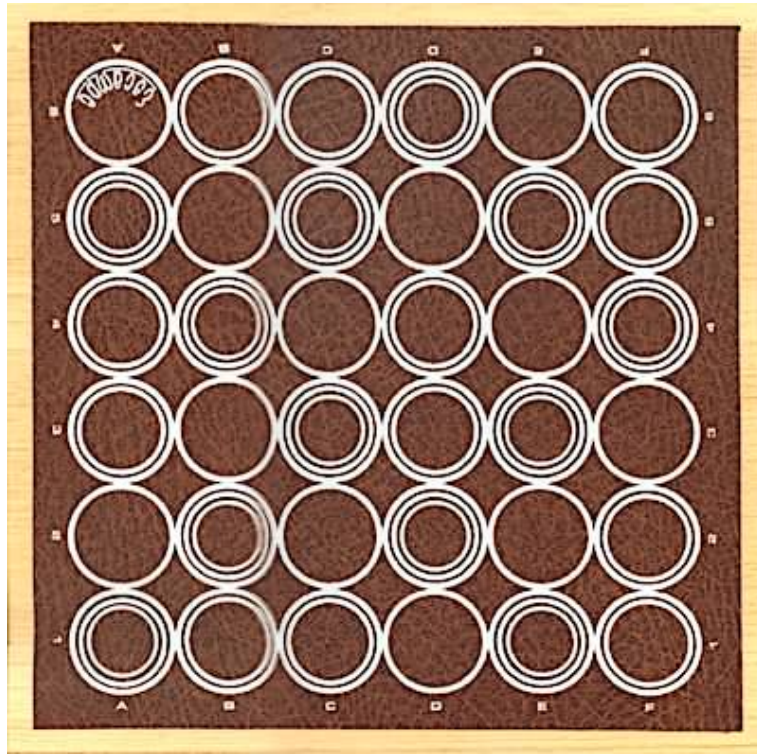


FIGURE 1 – Plateau du jeu Escampe



FIGURE 2 – Les différentes pièces du jeu Escampe

I) Première partie du devoir : analyse des caractéristiques du jeu (à rendre pour le 12/04 via eCampus)

La première partie de ce devoir est à rendre sous la forme d'un document au format pdf que vous soumettez sur eCampus.

Important : Le fichier soumis, devra avoir un nom de la forme Nom1_Nom2.pdf, où Nom1 et Nom2 sont les noms des membres du binôme.

L'objectif de cette première partie est de vous faire réfléchir sur les caractéristiques spécifiques de ce jeu. Vous pourrez remettre en cause vos choix pour la version finale.

On vous demande notamment de répondre aux questions suivantes. :

1. Comment modéliser un état du jeu (plateau et pièces restantes)? Préciser les avantages/inconvénients de votre représentation.
2. Comment déterminer si une configuration correspond à une fin de partie?
3. Essayez d'identifier les paramètres source de difficulté dans ce jeu. Quel est le facteur de branchement maximal de ce jeu pour chaque action?
4. Existe-t-il dans ce jeu des coups imparables, permettant la victoire à coup sûr d'un des joueurs?
5. Quels sont les critères que vous envisagez de prendre en compte pour concevoir des heuristiques

d'estimation de configuration de jeu (donner au moins 3 critères) ?

6. Est-il souhaitable pour ce jeu d'adopter une stratégie particulière en début, milieu ou fin de partie ?
7. Donnez un majorant du nombre de coups dans une partie. Détaillez les techniques que vous comptez mettre en œuvre pour respecter une contrainte de temps imposée sur la durée totale d'une partie.

II) Deuxième partie : codage des opérations de base pour Escampe (à rendre pour le 10/05/2026 (voir eCampus))

La seconde partie de ce devoir est à rendre sous la forme d'une archive compressée au format `.tgz` que vous disposerez sur eCampus

Important : Le fichier déposé, devra avoir un nom de la forme `Nom1_Nom2.tgz`, où `Nom1` et `Nom2` sont les noms des membres du binôme. L'archive devra contenir un unique répertoire, dont le nom sera `Nom1_Nom2` et contenant un sous-répertoire nommé `src` contenant les sources de votre code (NB : pour créer l'archive utiliser la commande : `tar cvzf Nom1_Nom2.tgz Nom1_Nom2`).

Il vous est demandé de vous concentrer dans cette partie sur les aspects spécifiques liés à la gestion des plateaux et au calcul des coups pour ce jeu.

1. Coder une classe JAVA `EscampeBoard` permettant de représenter une configuration de jeu. Vous ferez en sorte que cette classe implémente les méthodes de l'interface `Partie1.java` décrite dans la figure 3 et permettant de :
 - lire/sauvegarder un plateau dans un fichier
 - tester une fin de partie
 - tester la validité d'un coup
 - jouer un coup

Remarque : afin de réduire au minimum les contraintes d'interactions avec l'arbitre lors du tournoi, vous devez utiliser les conventions suivantes. Le coup sera présenté sous la forme d'une chaîne de caractères de type "B1-D1" (qui déplace la pièce en case B1 vers la case D1), ou de type "C6/A6/B5/D5/E6/F5" (qui pose le Licorne sur la case C6, et les paladins sur les cases A6, B5, D5, E6, F5) lors de la première disposition des pièces en début de partie. Quand un joueur ne peut pas jouer et doit passer son tour, il envoie "E".

2. Ajouter dans le fichier `EscampeBoard.java` une méthode `main` illustrant l'utilisation de ces méthodes sur des exemples convaincants.

Format de sauvegarde des plateaux

Les fichiers de saisie/sauvegarde de plateaux doivent être des fichiers texte comportant au moins 6 lignes consécutives, correspondant respectivement aux 6 lignes du plateau. Les lignes sont indicées de bas en haut, et les colonnes de gauche à droite (cf. figure 5).

Les lignes du fichier décrivent les lignes du plateau à l'aide des caractères suivants : "N" : licorne noire ; "n" : paladin noir ; "B" : licorne blanche ; "b" : paladin blanc ; "-" : case vide. Ces lignes commencent et finissent par un numéro. Toute autre ligne doit nécessairement débiter par le caractère "%" et sera considérée comme un commentaire, donc ignorée lors de la lecture.

La figure 6 décrit un exemple pour un tel fichier.

III) Version finale (à rendre pour le 30 mai 2026)

Les modalités pour la version finale seront précisées prochainement sur la page du cours eCampus.

Vous serez amenés à développer une I.A. pour votre joueur. Celle-ci devra être en mesure de conduire de façon autonome une partie en temps limité, en dialoguant avec un arbitre qui servira d'interface entre les deux joueurs. L'arbitre vérifiera la validité des coups proposés et s'assurera qu'aucun des joueurs ne dépasse la durée maximale autorisée pour une partie.

La durée maximale sera entre 300s et 900s (elle sera précisée plus tard) par partie et par joueur.

Afin de pouvoir automatiser le processus, votre joueur devra implémenter une interface particulière qui vous sera communiquée ultérieurement.

Tous les échanges avec l'arbitre se dérouleront suivant un protocole qui vous sera communiqué ultérieurement. Les détails et les classes à utiliser pour tester votre joueur seront disponibles sur eCampus du cours (section Devoir). Vous êtes donc invités à la consulter régulièrement.

Un tournoi sera organisé entre tous les joueurs artificiels capables de conduire une partie. Une partie de la note finale (3 points) tiendra compte des résultats de votre joueur au tournoi. La moitié des points restants sera attribuée sur la base du rapport écrit que vous aurez rendu et dans lequel vous détaillerez tous les choix effectués pour la mise en oeuvre de votre joueur. Le reste de la notation prendra en compte les techniques mises en oeuvre dans votre devoir.

Indications pour le rapport

Dans le rapport, vous expliquerez les raisons de vos choix d'implantation et de vos choix d'heuristiques. Vous préciserez la solution adoptée pour choisir le placement des pièces en début de partie, celle pour déterminer du meilleur coup à jouer, ainsi que toutes les astuces que vous avez implantées (structures de données, calcul heuristique efficace, éventuelle bibliothèque d'ouvertures, ...) Vous détaillerez les heuristiques que vous avez testées et les résultats qui vous ont fait choisir celle(s) que vous avez adoptée(s). Vous n'oublierez pas d'expliquer les solutions mises en oeuvre pour la gestion du temps réel, l'analyse des performances de votre joueur, les tests effectués et, en conclusion, les difficultés rencontrées dans ce devoir.

Faites un document concis et simple à lire, mais informatif et **précis**. Il n'y a pas de limite quant au nombre de pages.

Attention : Dans l'archive, veuillez inclure un fichier nommé "mainClass", contenant les 3 lignes suivantes :

jar : nomDeVotreJar.jar

clientClass : NomPackage.ClientJeu

mainClass : NomPackage.NomdeVotreJoueurFinale

Pour toute précision/ambiguïté concernant l'énoncé du devoir, envoyer un mail à yue.ma@universite-paris-saclay.fr.

```

public interface Partie1 {

    /** initialise un plateau à partir d'un fichier texte
     * @param fileName le nom du fichier à lire
     */
    public void setFromFile(String fileName);

    /** sauve la configuration de l'état courant (plateau et pièces restantes) dans un fichier
     * @param fileName le nom du fichier à sauvegarder
     * Le format doit être compatible avec celui utilisé pour la lecture.
     */
    public void saveToFile(String fileName);

    /** indique si le coup <move> est valide pour le joueur <player> sur le plateau courant
     * @param move le coup à jouer,
     *      sous la forme "B1-D1" en général,
     *      sous la forme "C6/A6/B5/D5/E6/F5" pour le coup qui place les pièces
     * @param player le joueur qui joue, représenté par "noir" ou "blanc"
     */
    public boolean isValidMove(String move, String player);

    /** calcule les coups possibles pour le joueur <player> sur le plateau courant
     * @param player le joueur qui joue, représenté par "noir" ou "blanc"
     */
    public String[] possiblesMoves(String player);

    /** modifie le plateau en jouant le coup move avec la pièce choose
     * @param move le coup à jouer, sous la forme "C1-D1" ou "C6/A6/B5/D5/E6/F5"
     * @param player le joueur qui joue, représenté par "noir" ou "blanc"
     */
    public void play(String move, String player);

    /** vrai lorsque le plateau coorespond à une fin de partie
     */
    public boolean gameOver();

}

```

FIGURE 3 – L'interface Partie1

1	2	2	3	1	2
3	1	3	1	3	2
2	3	1	2	1	3
2	1	3	2	3	1
1	3	1	3	1	2
3	2	2	1	3	2

FIGURE 4 – Carte des types de liseré sur le plateau, qui vous sera utile pour l'implémentation de l'interface Partie1.

%	ABCDEF	
01	bb----	01
02	-Bb-bb	02
03	-----	03
04	-----	04
05	-n-n-n	05
06	n-N-n-	06
%	ABCDEF	

FIGURE 5 – Exemple de situation initiale (Noirs : licorne en C6 ; paladins en A6, B5, D5, E6, F5. Blancs : licorne en C1 ; paladins en A3, C2, C5, F1, F4.)

%	ABCDEF	
01	nnN---	01
02	--b---	02
03	-----b	03
04	b-nn--	04
05	--b---	05
06	n-B--b	06
%	ABCDEF	

FIGURE 6 – Exemple de situation intermédiaire (Noirs : licorne en C1 ; paladins en A1, B1, C4, D4, A6. Blancs : licorne en C6 ; paladins en A4, C5, C2, F6, F3.) Nous partons du principe que vous avez les blancs, et que le coup précédent de Noir l'a amené en D4. Vous avez en C2 un paladin capable de prendre la licorne adverse (en passant par D2 et D1) ; or votre adversaire, en atteignant D4, vient de vous obliger à jouer une pièce partant d'une case à liseré double... Si vous choisissez F6-E5, Noir devra jouer A1-A2, atteignant alors une case triple, et permettant ainsi C2 X C1.