Projet d'informatique

UMONS – Année académique 2013-2014

Le contenu de ce document est présenté lors de la première séance du projet, le mardi 4 février 2014. Il vous est conseillé de relire attentivement ce document et d'y revenir au besoin.

1 Objectifs du projet

Les cinq principaux objectifs de ce projet sont les suivants :

1. Participer au développement d'un logiciel de taille conséquente (représentant environ 120 heures de travail, à répartir entre 2 étudiants).
2. Appliquer les notions d'*algorithmique* vues au cours « Programmation et Algorithmique 1 ».
3. Appliquer les notions de programmation *orientée objet* vues au cours « Programmation et Algorithmique 2 ».
4. S'initier aux bonnes méthodes / habitudes utiles au développement d'un projet logiciel (design modulaire des classes, éviter la redondance du code, documentation correcte, création et utilisation de tests, création de packages, etc.).
5. Apprendre à gérer son temps, à travailler en groupe et à s'organiser.

Gardez ces objectifs à l'esprit : ils nous servent également de critères pour vous évaluer (voir ci-dessous).

Contenu

2 Déroulement général du projet

Le but final de ce projet est de réaliser une application graphique en Java permettant de jouer à plusieurs jeux de société (Othello, Puissance 4 et Tic-tac-toe), soit contre un joueur humain, soit contre une des différentes intelligences artificielles (IA) que vous implémenterez. L'application devra en outre permettre de comparer deux IA et de charger une partie précédemment enregistrée. Une fois que les fonctionnalités de base exigées seront correctement mises en place, vous aurez le loisir de personnaliser et d'étendre votre projet (ce qui est encouragé, notamment via l'évaluation comme expliqué ci-dessous).

Le déroulement du projet est particulier. Vous allez devoir le réaliser petit à petit, mais sans perdre de temps et en vous répartissant correctement les tâches. Vous devrez faire des choix consciencieux lors de la conception. Pour ce faire, vous utiliserez au mieux les notions de la programmation *orientée objet* vues au cours de *Programmation et Algorithmique 2*. Ce cours étant donné en parallèle au projet, nous vous proposerons un calendrier adapté et nous organiserons des séances particulières pour vous aider sur un thème précis (voir ci-dessous).

2.1 Séances à l'horaire et travail en dehors de celles-ci

Parmi les heures prévues à l'horaire, il y a trois types de séances :
- **Séances spéciales** : séances (obligatoires) durant laquelle un assistant présente un sujet directement utile pour la réalisation du projet (par ex., une séance sur la conception d'interfaces graphiques). La première séance de ce type aura lieu le mardi 18 février 2014.
- **Séances questions** : séances (facultatives) où un assistant est présent pour répondre aux questions. Vos questions doivent être préparées à l'avance car l'assistant passera une et une seule fois parmi chaque groupe pour y répondre !
- **Autres séances** : lors des autres séances notées à l'horaire ou lors des fins de séances « spéciales » et « questions » où l'assistant n'est plus présent, vous êtes prioritaires par rapport aux autres étudiants en ce qui concerne l'utilisation de la salle Escher.

Un calendrier reprenant le type des séances sera disponible sur moodle (à partir du 17 février).

Le nombre d'heures nécessaires pour réaliser le projet est important (60h par personne, ce qui fait environ 120 heures de travail) et les heures indiquées dans l'horaire ne sont donc pas suffisantes : ce sont des heures destinées à vous donner des consignes ou des conseils, et permettant de répondre à vos questions. Pour mener à bien votre projet, vous devrez les compléter par des heures de travail en dehors des séances ! Essayez également de vous répartir les tâches de manière équilibrée et de vous organiser au mieux au sein de votre groupe. La gestion de votre temps est une des difficultés de ce projet : ne laissez pas le retard s'accumuler.

2.2 Aide et réponses aux questions : séances, forum et permanences

Nous ne répondons pas aux questions par email. Par contre, outre les séances « questions » décrites ci-dessus, un **forum** est disponible sur moodle pour y poser vos questions et pour vous entraider entre étudiants. L'utilisation du forum doit se faire de manière constructive et courtoise :
- Soignez la manière de rédiger vos questions et vos réponses pour qu'elles soient claires

et compréhensibles.
- Vous êtes autorisés à répondre aux autres étudiants mais dans le but de les faire réfléchir ou en donnant des pointeurs vers des références utiles. Ne donnez donc pas de solution toute faite. Ne postez pas non plus du code qui sera utilisé dans votre projet.
- L'équipe enseignante ne donnera a priori pas de réponse, sauf si la question (et sa réponse) s'avère constituer un éclaircissement utile à l'ensemble des étudiants. Même dans ce cas, n'attendez pas de réponse immédiate : il peut être nécessaire de discuter des éclaircissements à donner avant de les publier.

L'équipe enseignante peut modifier (corriger) ou supprimer vos messages si cela s'avère nécessaire. Une utilisation abusive du forum entraînera la fermeture de celui-ci.

Lors des derniers jours avant la remise du projet, des **permanences** (bureaux au deuxième étage aile E du Pentagone) seront organisées pour répondre à vos questions. Nous ne répondrons plus à vos questions en dehors de celles-ci. Veuillez dès à présent noter qu'il n'y aura plus de permanences à partir du mercredi 14 mai compris. Cela signifie que la dernière permanence sera organisée le mardi 13 mai 2014. Vous êtes prévenus plus de quatre mois à l'avance : prenez donc vos dispositions !

2.3 Soumission et défense du projet final

En fin d'année, vous rendrez un rapport écrit, votre code source et défendrez oralement votre projet (la défense orale a lieu pendant la session d'examen de juin). L'évaluation se base sur tous ces points (voir ci-dessous).

En ce qui concerne la rédaction du rapport, sachez que la présentation et l'orthographe seront considérées lors de l'évaluation. Pour vous aider, le document « Eléments de rédaction scientifique en informatique » est mis à votre disposition sur e-learning et rassemble quelques conseils [1].

La défense orale se fera par groupe lors de la session de juin. Elle commencera par une démonstration du programme (environ 10 minutes) et se terminera pas une série de questions posées de manière individuelle ou collective. Même si une partie du programme a été réalisée par une personne du groupe, les deux membres doivent être capables de répondre aux questions. La date et le lieu vous seront communiqués ultérieurement (via les horaires d'examens de la première session). Notez que les deux étudiants d'un groupe seront interrogés de manière équilibrée lors de la défense et n'obtiendront pas forcément la même note.

3 Aspects administratifs et consignes

3.1 Composition des groupes

Le projet est réalisé par **groupes de 2 étudiants**. Prévenez-nous rapidement si vous avez un souci pour former un groupe de 2 étudiants. Vous nous communiquerez la composition de votre groupe le mardi 18 février 2014 (séance obligatoire). A partir de cette date, prévenez-nous le **plus vite possible** si vous avez un problème avec votre binôme (abandon, etc.). Si

1. Tout au long de vos études vous serez amenés à rédiger différents rapports, jusqu'à la rédaction de votre mémoire.

nous l'apprenons trop tard, nous ne pourrons plus vous aider à trouver une solution (si nous sommes prévenus après Pâques, nous ne pouvons plus en tenir compte pour l'évaluation de votre projet).

3.2 Rapport

Le rapport final doit contenir :
- la répartition des tâches au sein du groupe ;
- une description argumentée des choix personnels effectués ;
- les points forts de votre projet (fonctionnalités supplémentaires, optimisation, complexité,...) ;
- les points faibles de votre projet (vitesse d'exécution lente, faiblesses de certains algorithmes,...) ;
- les différentes erreurs connues du programme. Les erreurs rencontrées lors de nos tests et qui ne sont pas répertoriées dans le rapport seront considérées plus sévèrement ; et
- les apports positifs et/ou négatifs de ce projet.

En plus des éléments repris ci-dessus, vous pouvez ajouter tout élément qui vous semble utile comme par exemple un *diagramme de classes UML* (cf. cours de Programmation et Algorithmique 2).

Le rapport devra être remis en PDF. **Attention à votre orthographe, elle sera considérée lors de l'évaluation.** Pour rappel, une note contenant des conseils de rédaction est disponible sur la page moodle du projet.

Pour vous aider à réaliser votre rapport, nous vous conseillons de maintenir un « journal de bord » tout au long du projet (concrètement, cela peut être un petit carnet ou un simple fichier de texte). Celui-ci vous permettra d'y noter progressivement vos idées, d'y indiquer la répartition des tâches (qui réalise quelle tâche au sein de votre groupe). Egalement, si quelque chose dans l'énoncé vous paraît ambigu ou trop peu défini, vous devrez faire un choix (en le justifiant) : notez ce choix et vos arguments dans votre journal puis dans votre rapport.

3.3 Code source

Voici les consignes à respecter concernant le code source.
- Les noms des interfaces, classes et méthodes que vous créerez seront en **anglais**.
- Par convention, les noms des interfaces commenceront par `I` (par ex. `IPlayer`) et ceux des classes abstraites par `A`.
- Le code doit obligatoirement être **documenté** en utilisant la javadoc de base (`@return`, `@param` et `@throws`).
- Les commentaires et la documentation peuvent être en français ou en anglais (mais restez cohérents une fois la langue choisie).
- Vous devrez inclure des tests unités pour **au moins une partie précise de votre projet** (voir section 7.4).
- Pour l'évaluation et la défense orale, votre code sera exécuté sur une machine de l'université et seule la version soumise sera prise en compte. Une description technique de la machine utilisée vous sera fournie (OS, version de la machine virtuelle java, etc.). Celle-ci sera similaire à ce que vous utilisez en salle machine (Escher).

– Vous devez utiliser *Ant* (Apache) pour simplifier et automatiser le processus de compilation et d'exécution de votre code source : en ouvrant votre archive les 3 commandes suivantes doivent impérativement avoir le comportement attendu :
  1. `ant build` : compile votre programme ;
  2. `ant run` : exécute votre programme ;
  3. `ant test` : lance les tests unités.

3.4 Défense du projet

Lors de la défense orale du projet (pendant la session de juin), votre application sera exécutée sur une machine de la salle Escher et non pas sur un ordinateur personnel.

3.5 Plagiat et triche

Le *plagiat* consiste à s'approprier comme personnel du texte, une image ou du code réalisé par une autre personne (en ce compris un texte traduit), sans le préciser de manière explicite. Vous pouvez baser vos arguments sur des éléments que vous avez lu, mais vous devez citer vos sources. S'appuyer sur des résultats connus pour en développer de nouveaux est d'ailleurs à la base de la démarche scientifique. Il faut simplement être honnête et très clair sur ces points, mais le plagiat est donc simple à éviter.

D'autre part, le plagiat est **strictement interdit** à l'université et peut entraîner des sanctions graves allant beaucoup plus loin que le simple échec au projet (cf. règlement des études et page dédiée à ce sujet sur le site de l'université [2]). Vous devez soumettre vos fichiers via moodle (et donc pas par email) où ils subiront une détection automatique du plagiat.

Le projet étant individuel, le fait que deux groupes différents ont manifestement échangé du code sera donc considéré comme de la triche et sera sanctionné en conséquence (pour les deux groupes).

3.6 Notes de présence et absences

En ce qui concerne le projet en première session, vous obtiendrez
– une **note entre 0 et 20** si le projet est soumis, recevable *et* défendu oralement pendant la session ;
– une **note de 0 sur 20** si le projet n'est pas recevable pour au moins un des critères énoncés en Section 4 ;
– une **note de présence** si le projet n'est pas déposé mais que vous signalez votre intention de ne pas le soumettre en envoyant un email à `hadrien.melot@umons.ac.be` avant le 16 mai 2014 à 20h (vous recevrez un accusé de réception).

Dans *tous les autres cas* (soumis mais pas défendu, pas soumis sans nous prévenir, etc.), vous serez noté **absent** pour le projet d'informatique.

2. `http://portail.umons.ac.be/FR/universite/admin/aff_academiques/Pedagogie_Qualite/Pages/Plagiat.aspx`

4 Checklist des éléments indispensables pour que votre projet soit corrigé !

Vérifiez scrupuleusement les quatre éléments repris ci-dessous. **Le manquement à l'un de ces éléments entraînera le fait que votre projet ne sera pas recevable et donc une note de 0 / 20.** Notez que ce sont des choses toutes simples à vérifier et qu'aucune exception ne sera faite, quelles que soient les circonstances. Ce serait vraiment dommage de ne pas pouvoir présenter votre projet à cause d'un des quatre points ci-dessous !

1. *Deadline.* La date limite de remise est le **vendredi 16 mai 2014 à 20h**. **La plateforme n'acceptera pas de retard.** Aucun ajout ni aucune modification du projet ne sera acceptée au-delà de cette date.
2. *Ant.* Vous devez créer un fichier `build.xml` pour utiliser *Ant*[3] (Apache) pour simplifier et automatiser le processus de compilation et d'exécution de votre code source : en ouvrant votre archive dans une console, les 3 commandes suivantes doivent impérativement avoir le comportement attendu :
   (a) `ant build` : compile votre programme ;
   (b) `ant run` : exécute votre programme ;
   (c) `ant test` : lance les tests unités.
3. *Compilation.* Votre code doit pouvoir être compilé et exécuté sur une machine de l'université et seule la version soumise sera prise en compte. La description technique de la machine correspond à une machine de la salle Escher, à savoir un environnement **Linux (Ubuntu 12.04)** et **Java 1.6**. La compilation **ne peut pas produire d'erreur** empêchant la création des fichiers `.class` (warnings autorisés). En d'autres mots, nous devons être capable d'exécuter le programme sans modifier le code.
4. *Archive.* Le travail doit être livré sous la forme d'une archive (`.zip` ou un format libre) portant, en majuscules uniquement, votre nom. L'archive **doit** contenir un répertoire portant ce même nom. Ce répertoire comporte :
   a) le rapport au format **PDF** ;
   b) le code de votre application (dans un sous-répertoire `code`) ;
   c) un fichier `build.xml` (*Ant*) pour permettre d'utiliser les trois commandes `ant` décrites ci-dessus.

   Ne sont requis que le code source (fichiers `.java`), les tests unités imposés, ainsi que les fichiers nécessaires à la compilation et l'exécution (par ex. images). Votre ne devez **pas** inclure les fichiers compilés `.class`. Si vous souhaitez fournir d'autres éléments, placez-les dans un sous-répertoire nommé `misc` et indiquez dans un fichier `README.txt` les détails de ces différents éléments.

5 Conseils

Comment éviter le stress et m'assurer à temps que mon projet sera recevable ?

3. Voir slides disponibles à ce sujet sur la page Moodle du cours de Programmation et Algorithmique 2.

1. *Mettez la priorité là où il le faut.* Assurez-vous d'avoir en priorité une version simple mais qui répond à nos exigences et qui implémente les fonctionnalités demandées. Ensuite, s'il vous reste du temps, vous pourrez peaufiner le côté esthétique ou améliorer d'autres points de détails.
2. *Rédaction.* Rédiger prend du temps ! Le rapport ne doit pas être plus long que nécessaire mais doit contenir ce qui est attendu (voir section 3.2), et le présenter de manière adéquate. Commencez votre rapport au plus vite (pourquoi pas dès maintenant ?). Soignez la forme autant que le contenu et surtout : citez vos sources et évitez le plagiat (voir ci-dessus).
3. *Test réel.* Faites un test dans la salle Escher en utilisant exactement ce que nous recevrons :
   – ouvrez sur une machine du Escher l'archive telle que vous comptez la soumettre ;
   – testez la compilation et l'exécution.
4. *Soyez prévoyants.* Prévoyez de soumettre une version préliminaire au moins 24 heures à l'avance pour éviter toute mauvaise surprise (problèmes de connexion, etc.). C'est toujours mieux d'être évalué sur cette version sans doute imparfaite [4], plutôt que de ne pas être jugé du tout !

6 Evaluation de votre projet

Sur quels critères vais-je être évalué ?

Nous nous basons sur le rapport, le code et la défense orale. Pour chacun de ces éléments, une série de critères et de questions sont utilisés pour évaluer la qualité d'un projet.

1. *Rapport.*
   (a) Forme : orthographe, style adapté, structure, clarté.
   (b) Contenu : le contenu est-il complet et informatif ? Pour rappel, votre rapport doit contenir :
      – Une description argumentée des choix personnels effectués ; en particulier vous expliquerez les idées exploitées pour décrire vos IA ;
      – Les points forts de votre projet (fonctionnalités supplémentaires, optimisation, complexité,...) ;
      – Les points faibles de votre projet (vitesse d'exécution lente, faiblesses de certains algorithmes,...) ;
      – Les différentes erreurs connues du programme. Les erreurs rencontrées lors de nos tests et qui ne sont pas répertoriées dans le rapport seront considérées plus sévèrement ; et
      – Les apports positifs et/ou négatifs du projet.
2. *Code.*
   (a) Les fonctionnalités de base sont-elles respectées ?
   (b) Des éléments supplémentaires aux fonctionnalités de base ont-ils été ajoutés ?
   (c) Les tests unités (pour au moins une partie du projet) sont-ils complets et pertinents ?

4. Si vous êtes trop perfectionniste, quelle version sera vraiment parfaite ?

(d) Les algorithmes ont-ils été écrits avec un souci d'efficacité ?

(e) Les notions de la programmation Orientée Objet (héritage, interface, exceptions, etc.) sont-elles utilisées de manière correcte, élégante et pertinente ?

(f) La documentation (javadoc de base : `@return`, `@param` et `@throws`) est-elle présente ?

(g) La documentation (javadoc) est-elle pertinente et claire ?

(h) Le code est-il de bonne qualité et bien organisé ?
- lisibilité ;
- non redondance ;
- design modulaire ;
- organisation en packages ;
- respect des conventions, etc.

Pour rappel : les noms des interfaces, classes et méthodes sont en **anglais**. Les noms des interfaces commencent par `I` (par ex. `IPlayer`) et ceux des classes abstraites par `A`.

3. *Défense orale.*
   (a) Réponses aux questions.
   (b) Intérêt de la démonstration.
4. *Respect des consignes.* Présence aux séances obligatoires, etc.

Rappel : les étudiants au sein d'un groupe n'obtiendront pas systématiquement la même note. Cela dépendra des réponses aux questions ou d'autres critères objectifs.

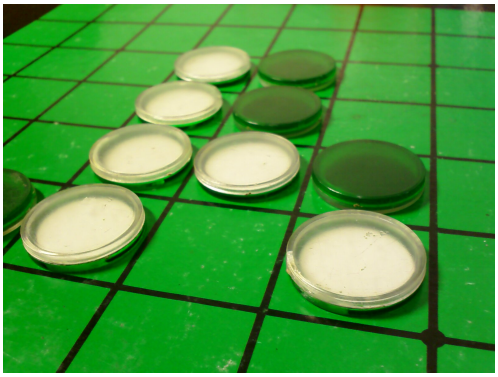
7 Description de votre application

Il est attendu que chaque groupe d'étudiants conçoive une application logicielle en essayant au maximum d'appliquer les concepts et principes vus au cours de l'année, principalement les concepts orienté objet (héritage, interfaces, exceptions, ...). A ce sujet, la section 6 donne la liste des critères qui nous serviront à évaluer votre projet.

Votre application doit permettre de jouer à trois jeux différents (Othello, Puissance 4 et Tic-tac-toe) via une interface graphique et en utilisant la souris sur une taille de grille standard et en utilisant les règles classiques qui sont disponibles sur les pages wikipedia suivantes :
- Othello : `http://fr.wikipedia.org/wiki/Othello_%28jeu%29` ;
- Puissance 4 : `http://fr.wikipedia.org/wiki/Puissance_4` ;
- Tic-tac-toe : `http://fr.wikipedia.org/wiki/Tic-tac-toe_%28jeu%29`

Seules les règles énoncées sur ces pages Wikipedia à la date du 3 février 2014 à 12h (GMT) seront considérées. Vous pouvez consulter la page telle qu'elle apparaissait le 3 février 2014 à 12h (GMT) en cliquant dans l'onglet « Afficher l'historique » et en vérifiant si et quand celles-ci ont été modifiées.

La sélection du jeu actif (parmi les trois disponibles) et les différentes options de jeu (voir ci-dessous) pourront être sélectionnées via des menus.



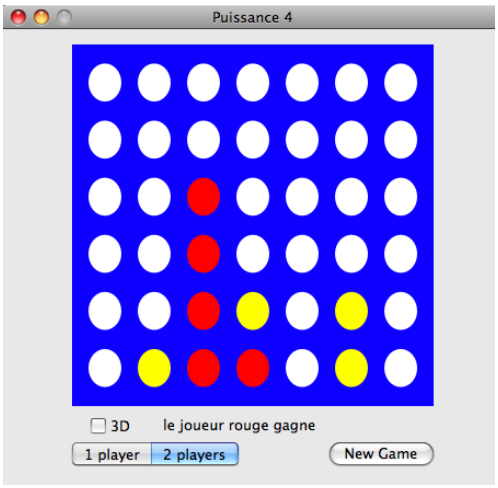


FIGURE 1 – Exemples de parties d'Othello et de Puissance 4 en cours. (Sources : wikipedia)

7.1 Intelligences Artificielles et modes de jeux

Pour chaque jeu, vous déterminerez au moins 2 intelligences artificielles (IA). La première peut être très simple et naïve. Votre deuxième IA sera plus subtile et essayera par exemple d'exploiter une stratégie (comme par exemple celles décrites brièvement sur wikipedia). N'oubliez pas de décrire les algorithmes utilisés par vos IA dans le rapport. Vos deux IA de base devront jouer dans un délai raisonnable (de l'ordre de moins de 5 secondes pour un coup). Vous pouvez envisager autant d'IA supplémentaires (facultatives) que vous le voulez en plus de vos deux IA de base. Les IA supplémentaires n'ont pas de contraintes (donc pas de limite de temps).

Votre programme doit permettre tous les modes de jeux :
- joueur humain contre joueur humain ;
- joueur humain contre IA ;
- IA contre IA.

De plus, dans le cas d'une IA, l'utilisateur doit pouvoir choisir parmi les IA disponibles (par exemple « aléatoire » ou « difficile »).

Dans le cas d'un jeu « IA contre IA », vous prévoirez deux modes : un mode « continu » où les IA jouent directement l'une après l'autre ; et un mode « coup par coup » où l'utilisateur appuie sur un bouton « suivant » pour afficher le coup suivant.

7.2 Comparateur d'IA

Votre programme doit permettre de comparer deux IA en demandant à l'utilisateur de rentrer un nombre n de parties et de sélectionner deux IA parmi celles disponibles. Ensuite, le programme lancera n parties entre les deux IA consécutivement et affichera ensuite les résultats sous la forme d'un pourcentage de parties gagnées par chacune des IA sélectionnées. Il n'est pas nécessaire que l'utilisateur visualise les parties, ce qui importe dans ce mode étant d'obtenir le résultat statistique final.

7.3 Archivage

Il doit être possible à tout moment de sauvegarder la partie en enregistrant l'état du jeu courant. Egalement, l'application doit permettre de charger une partie précédemment sauvegardée pour pouvoir la continuer.

7.4 Tests unités

Vous devez **obligatoirement** fournir des tests unités pour tester quand une partie est terminée et que les conditions de victoires (partie gagnée, perdue ou nulle) sont correctement gérées. Vous pouvez, de manière facultative, tester tout autre comportement ou fonctionnalité de votre programme.

7.5 Eléments supplémentaires

Les éléments décrits ci-dessus constituent la version de base de votre projet. Une version de base qui serait *parfaitement* conçue et qui attesterait du souci de bien faire (bonne utilisation du paradigme orienté objet, clarté du code, jouabilité, pas de bug détecté, documentation complète, respect des consignes, rapport clair et complet, etc. : voir Section 6) vous permettrait d'avoir une note globale de maximum 16/20.

Pour augmenter la note, diverses choses supplémentaires peuvent être ajoutées au projet. Par exemple :

- Possibilité pour l'utilisateur de modifier la taille de la grille ou le nombre de jetons à aligner pour gagner.
- Affichage d'un conseil pour le prochain coup, pour un joueur. Vous pouvez par exemple vous baser sur ce que ferait votre IA la plus forte pour donner ce conseil. Ce conseil ne devrait être affiché que si on en active un mode « triche ».
- Chronomètre et scores basés sur le temps de jeu et le nombre de coups.
- Tableau des meilleurs scores.
- Efforts particuliers pour les graphismes (par exemple : effet visuel du jeton qui tombe dans sa rangée au Puissance 4).
- Ajout de statistiques. Exemple : compter le nombre de fois qu'un joueur était forcé de jouer la position i sous peine de laisser l'adversaire jouer en i et de gagner.
- Coloration optionnelle des endroits où il est pertinent de jouer « pour ne pas perdre »(soit au prochain tour, soit par la suite si le jeu est capable de le détecter) et coloration des endroits où il est pertinent de jouer « pour gagner » (idem).
- Variantes de jeux : par exemple un mode « Tetris » au Puissance 4 : quand une ligne entière est remplie, elle disparait et les jetons au-dessus redescendent.

Cette liste n'est pas exhaustive et les initiatives personnelles de qualité sont encouragées (il est cependant conseillé d'avoir d'abord une version de base solide avant de vous lancer dans les éléments optionnels).

Bon travail !