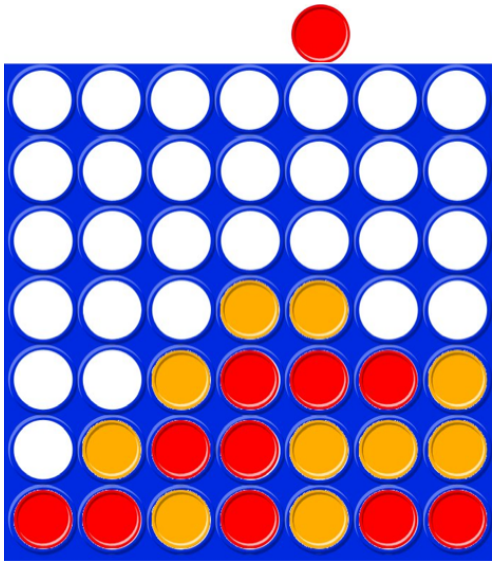
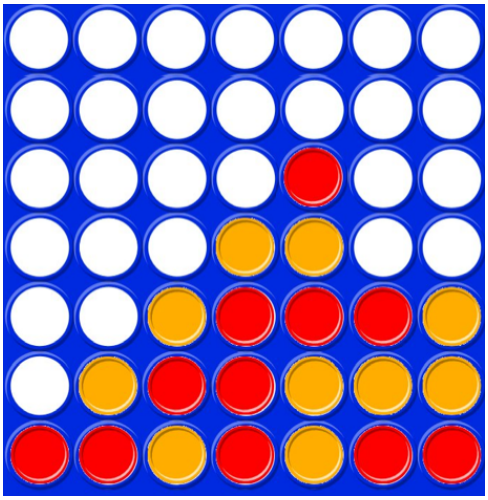
Puissance 4

Puissance 4 est un jeu de société combinatoire qui se joue à deux joueurs. Il a été édité pour la première fois en 1974 par MB.

Chaque joueur dispose de 21 jetons : rouges pour le premier joueur et jaunes pour le second. Une grille rigide comptant 7 lignes et 7 colonnes[1] est à disposition des deux adversaires dont le but est d'être le premier à aligner 4 jetons de même couleur horizontalement, verticalement ou diagonalement.

À chaque tour, un joueur choisit une colonne de la grille et lâche son jeton au sommet de celle-ci. Ce dernier est alors attiré par la gravité et tombe jusqu'en bas si la colonne est vide ou se positionne juste au-dessus du dernier jeton de la colonne, si celle-ci n'est pas vide. L'exemple ci-dessous illustre un coup joué par le joueur au jeton rouge.





La partie se termine dès qu'un joueur aligne 4 jetons. Il est alors déclaré gagnant. Si les joueurs ont placé tous leurs jetons, sans qu'aucun deux ne soit arrivé à en aligner quatre, alors la partie est déclarée nulle.

À la fin de chaque partie, le programme demande aux joueurs s'ils souhaitent rejouer ou pas. Si ces derniers décident de ne plus jouer, il leur affiche une synthèse : un message contenant le nombre de parties jouées et le nombre de parties gagnées par chaque joueur.

[1] Dans la version originale c'est 6 lignes et 7 colonnes mais cela complique légèrement l'algorithmique.

Travail à réaliser

IHM (Interface Homme Machine) : pour cette première partie du travail, vous avez deux alternatives. Selon vos préférences, vous pouvez :

- proposer une interface simple, en console, à l'aide des outils vus en cours/TD/TP, permettant aux joueurs de visualiser l'état de la grille au cours d'une partie.
- utiliser l'interface graphique (GUI, *Graphical User Interface*) que nous avons préparée et qui peut être pilotée très facilement par une petite bibliothèque de méthodes simples d'utilisation. Le code source (télécharger via le bouton *Download ZIP* à droite) et la documentation sont disponibles en ligne ici : https://github.com/blgatelierl2/projetl1p4.

Ce choix n'aura pas d'influence sur la note finale, seule sera évaluée l'exploitation qui en aura été faite (en particulier, une interface textuelle solide est préférable à un usage approximatif de l'interface graphique).

Le moteur du jeu : programmer le moteur du jeu de *Puissance 4* tel qu'il a été défini précédemment en respectant le schéma suivant :

- Au début de la partie la grille vide est affichée.
- Chaque joueur joue à tour de rôle.
- Dès qu'une combinaison de 4 jetons est réalisée la partie se termine.
- L'ordinateur propose de rejouer. En cas de refus, la synthèse est affichée.

Gestion de la rotation : désormais au début de chaque partie l'ordinateur vous demande si vous souhaitez intégrer la rotation dans cette partie. Ainsi à chaque tour de jeu, un joueur peut **soit** faire glisser un jeton dans une colonne, **soit** effectuer une rotation à 90° de la grille du jeu. Deux sens de rotation sont possibles : la première à droite et la seconde à gauche. La figure ci-dessus illustre l'enchaînement suivant : rotation à droite de la situation initiale (A), suivie d'une rotation à gauche.

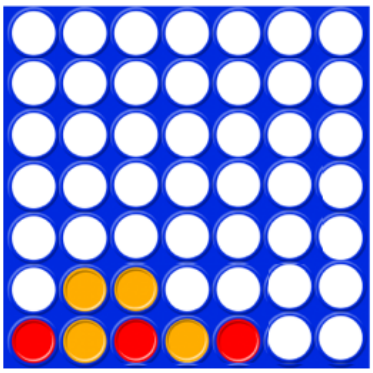


Figure A : situation initiale

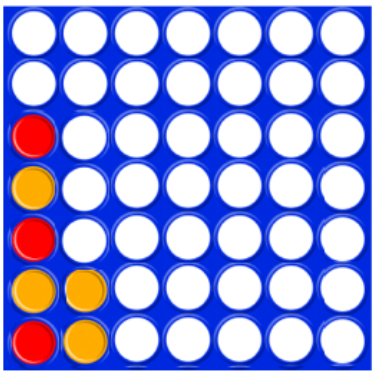


Figure B : rotation à droite de la figure A

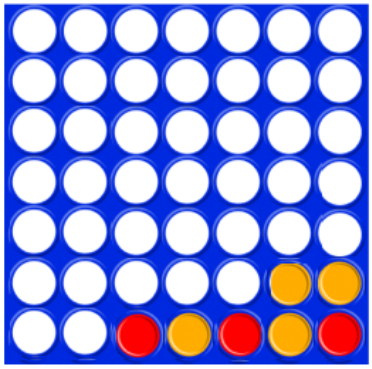


Figure C : rotation à gauche de la figure B

Le nombre de rotations par partie est limité à quatre par joueur. Notez que le choix du type de partie (avec ou sans rotation) se fait en début de partie. On peut donc faire une partie avec rotation puis enchaîner sur une autre sans rotation et ainsi de suite jusqu'à la fin complète du jeu, qui sera matérialisée par une réponse négative à la demande de refaire une nouvelle partie.

L'aperçu avant validation : lors d'une partie avec rotations, le joueur doit faire preuve d'un bon niveau d'imagination afin de visualiser les conséquences de ces rotations. On proposera alors à chaque joueur une option "aperçu avant validation" qui est disponible **UNIQUEMENT** pour les parties avec rotation. Chaque joueur n'a droit qu'à deux aperçus par partie. Il peut les utiliser comme bon lui semble. Il peut par exemple, en demander un pour voir l'effet d'une rotation à gauche, puis enchaîner sur un autre aperçu pour voir l'effet d'une rotation à droite et décider finalement de ne pas faire de rotation mais d'insérer un jeton.

Un mode IA : l'ordinateur peut suivre différentes stratégies de la plus naïve à la plus complexe. Malheureusement les outils théoriques et pratiques que vous avez vus jusque là vous limitent à des stratégies de base non optimisées.

Par ailleurs, sachez que la version sans rotation a été étudiée par L. Victor Allis dans sa thèse *A Knowledge-based Approach of Connect-Four* (1989). Il a montré que la personne qui commence a toujours la possibilité de gagner (sur la grille standard 7x6) si elle joue selon une certaine stratégie définie dans la thèse.

Pour le moment, on pourra se contenter de la stratégie naïve suivante :

Supposons que l'IA ait les jetons jaunes. Si une rotation permet de gagner, l'IA la joue. Sinon, pour chaque case libre, l'IA calcule combien au max elle peut aligner de jetons en mettant un de couleur jaune et combien l'adversaire peut au max aligner en mettant un jeton de couleur rouge.

On score maintenant les cases de la manière suivante :

7 si l'IA peut aligner 4 sinon
6 si l'adversaire peut aligner 4 sinon
5 si l'IA peut aligner 3 sinon
4 si l'adversaire peut aligner 3 sinon
3 si l'IA peut aligner 2 sinon
2 si l'adversaire peut aligner 2
1 sinon

L'IA ordonne alors les coups possibles selon le score de chacun. Elle commence ensuite par le coup X ayant le plus grand score et teste si en le jouant l'adversaire pourrait gagner en faisant de suite une rotation. Si ce n'est pas possible alors l'IA joue le coup X sinon elle refait la même chose avec le coup ayant le second meilleur score et ainsi de suite.

Remarque : l'IA proposée ne sera pas extrêmement forte. Afin d'éviter qu'un joueur puisse se contenter de mémoriser une partie gagnée contre l'IA et de la répéter pour gagner à chaque fois, il est utile d'introduire un peu d'aléatoire dans la stratégie ! On recommande par exemple de tirer aléatoirement l'ordre dans lequel seront considérées les cases de même score dans la stratégie précédente. Le choix parmi deux cases équivalentes (il y en a beaucoup en début de partie) sera ainsi aléatoire et les parties seront moins facilement reproductibles.

Organisation de votre travail

(1) Commencer par réfléchir à la structure de données dont vous aurez besoin.
(2) Décomposer votre travail en sous-programmes.
(3) Écrire l'algorithme de chaque sous-programme.
(4) Programmer le tout en Java.

La présence aux séances est **OBLIGATOIRE** et la note du projet tiendra compte du travail réalisé en séances en plus du programme rendu. Votre code doit être clair, lisible et bien commenté.

Travail optionnel proposé

Ceux qui terminent le projet avant la fin des séances de TP pourront aller encore plus loin dans le développement de l'IA. Attention, cette extension ne peut **rapporter des points en plus uniquement si tout le travail avec IA de base est parfaitement propre et complet**.

Voici par exemple quelques points de réflexion dont vous pourriez ajouter la prise en compte par votre IA :

- Est-il possible de poursuivre l'alignement jusqu'à 4 : ça ne sert à rien de jouer une case qui permet d'aligner 2 ou 3 si on sait déjà qu'on ne pourra pas aligner 4 car les positions sont déjà occupées par l'adversaire.
- Maximiser le nombre d'alignements : il vaut mieux jouer une case qui réalise 3 alignements de 2 qu'un seul.
- Le nombre d'alignements de 4 qui seront possibles : plus cela ouvre d'opportunités, mieux c'est.
- Possibilité d'évaluer N coup à l'avance : vous pouvez alors choisir votre niveau de difficulté dans chaque partie ! Plus N est grand, plus l'IA est performante.