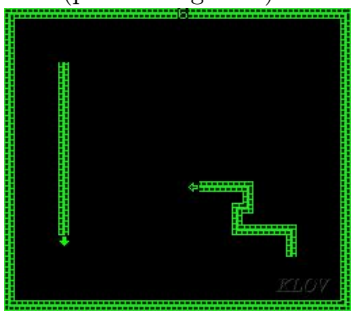
Projet de Programmation
Licence 1 Informatique UPEC 2023/2024
Snake-Blockade

Le jeu Snake-Blockade

Blockade est un ancien jeu arcade, précurseur de Snake, dans lequel deux joueurs (bleu et rouge) contrôlent deux serpents qui avancent de manière régulière. A chaque tour chaque serpent avance vers l'une des 4 cases voisines (pas en diagonale) selon la commande du joueur respectif.



Dans la version proposée pour implémentation, les serpents s'allongent d'une case tous les N tours. Si $N = 2$ par exemple, au premier et 2e tour le serpent n'a qu'une seule case (sa tête), puis au 3e et 4e tour il a deux cases (il s'est allongé d'une case au 3e tour) etc. N doit être fixé en début de partie de commun accord entre les joueurs. Lorsque la tête avance et c'est un tour où la longueur du corps n'augmente pas, la queue du serpent quitte sa case (qui devient libre).

Quand un joueur n'a plus de case libre pour avancer, il perd. Une autre façon de perdre est de jouer un coup qui fait rentrer la tête de son serpent dans un obstacle (le bord du tablier, son propre corps ou le corps de l'adversaire). Chaque joueur doit donc essayant de bloquer l'autre tout en faisant attention à se laisser suffisamment d'espace pour faire avancer son serpent. Le tablier peut contenir des obstacles. Des "fraises" peuvent apparaître aléatoirement et lorsqu'un serpent "avale" une fraise, son corps diminue d'une case (ou reste la même si c'était un tour multiple de N !).

Minimum syndical – pour une note de 10

Votre code doit être écrit en **Java** (pas en Robusta !). Il vous est demandé de fournir au moins un programme qui puisse faire jouer deux joueurs humains l'un contre l'autre. Votre programme devrait "arbitrer" le jeu, afficher la configuration courante, gérer le tour de rôle, demander au joueur courant quel coup jouer, augmenter ou pas la taille du serpent en fonction du tour et/ou si on a avalé une fraise et décider lorsque un joueur a perdu selon l'une des deux règles décrites ci-dessous :

1. soit parce que le coup joué mène son serpent dans un obstacle, dans son propre traînée ou dans la traînée de l'autre serpent,
2. soit parce qu'il n'y a aucune case libre autour de la tête de son serpent.

Attention ! Bien lire la section sur la qualité de votre code ci-dessous ! Soumettre du code de bonne qualité est aussi essentiel pour avoir cette note du minimum syndical !

Pour viser plus que 10 : graphisme, IA, tournoi, pouvoirs supplémentaires

Vous pouvez ne pas vous contenter du minimum syndical, voici les objectifs qui, si accomplis, peuvent vous apporter des points :

Affichage Dans le minimum syndical vous pouvez afficher la configuration courante (corps de chaque snake) dans la console, comme un tableau de caractères. Un affichage avec des pixels peut vous apporter 1-2 points de plus.

IA et stratégies Il vous est demandé d'implémenter des IA qui soient capables de jouer (tant bien que mal) contre un humain. Pour cela il faut imaginer et implémenter des *stratégies* : des "plans" d'action de l'IA pour jouer en fonction de la situation sur la table. Il faut au moins que l'ordinateur puisse choisir une case libre. Il faut au moins que l'ordinateur puisse choisir aléatoirement une case disponible. Ensuite l'ordinateur pourrait chercher à coincer l'autre joueur, à poursuivre un but assez simple comme avancer dans

une direction tant qu'il y a des cases libres, ou bien à se créer un grand espace de jeu. A vous de chercher... et de trouver.

Un tel programme devrait donc permettre de faire jouer un humain contre le programme lui-même (qui employera donc l'IA que vous avez conçue), ou de faire jouer deux programmes entre eux – et c'est le but du point suivant !

Une intelligence artificielle donnera lieu à un bonus de 1 à 4 points.

Tournoi

Pour les groupes qui développent des IA plus intelligentes que l'IA aléatoire, nous allons organiser un tournoi : l'idée est de lancer en exécution deux programmes proposés par des binômes différents, qui communiqueront à travers internet (pour cela il faudra respecter un protocole très précis en Java), chacun considérant l'autre programme comme son adversaire.

Nous fournirons des classes Java qui implémentent des tampons permettant la communication entre deux programmes. Votre programme devra être capable d'écrire et de lire les coups dans ces tampons. On vous donnera le format pour lire et écrire les coups.

Chaque programme calculera son prochain coup et l'écrira dans le tampon, puis il attendra que l'autre programme écrive son coup dans dans le tampon pour répondre. On réalisera un tournoi entre les binômes (et peut-être on inscrira dans ce tournoi nos programmes à nous, les enseignants...), et que le meilleur gagne. Le tournoi se fera sur un tablier sans obstacles ni fraises.

La participation au tournoi donnera lieu à un bonus de 2 points. La gagnant du tournoi aura un prix spécial !

Obstacles et fraises Un bonus de 2 points sera donné aux projets qui ajouteront des fonctionnalités, comme des obstacles et/ou des "fraises" qui, une fois "mangées", diminuent la longueur de son snake. D'autres types de fraises peuvent être proposés.

Critères d'évaluation de la qualité de votre code

- Votre programme principal (c'est à dire la fonction `main`) doit comporter moins de 30 lignes et ne doit comporter que des appels de fonctions et des déclarations.
- Chaque fonction doit contenir moins de 10 lignes. Si une de vos fonctions contient plus de 10 lignes, vous devez alors écrire au moins une sous-fonction afin de découper votre programme. (Cette règle n'est pas stricte : une fonction de 11 lignes sera acceptée par exemple.)
- Vous devez séparer le traitement des données des interactions avec l'utilisateur, c'est à dire que chaque sous-programme (fonction ou procédure) effectuant une action sur le jeu doit prendre en paramètre toutes les variables nécessaires et non demander à l'utilisateur de lui donner. De cette manière, vous pourrez tester plus facilement chaque sous-programme. Toutes les interactions avec l'utilisateur seront donc programmées dans les menus.
- Vous devez commenter votre code : avant chaque sous-programme et plus généralement, avant chaque bloc, un commentaire devra expliquer ce que fait le code.
- Vous devez respecter les conventions de nommage : les noms des variables et des sous-programmes devront toujours être en minuscules, seules les premières lettres des noms composés pourront être en majuscules (ex : `int nombre, nombreJoueurs;`) ; les constantes devront être en majuscules (ex : `MAXUSER`), et les noms des structures s'écrivent en minuscules, sauf la première lettre (ex : `Utilisateur`). De plus, les noms doivent être significatifs pour la clarté du code.
- Chaque menu sera un sous-programme et quitter le menu reviendra à quitter ce sous-programme.

Dates

Vous devrez choisir votre binôme avant **lundi 8 avril**. Un espace EPREL sera créé pour nous communiquer votre binôme et pour le suivi du projet. Le code du projet devra être soumis sur EPREL au plus tard **mercredi 22 Mai, à 23h59**. Aucun retard ne sera permis, les soumissions en retard seront notées 0/20.

Suivi du projet

Vous devez présenter l'état d'avancement sur votre projet au moins 3 fois pendant le mois de mai, pendant des séances de suivi de projet. Ces séances de suivi seront signalées début avril et vont apparaître sur ADE. À chaque séance de suivi, un plan de travail à accomplir pour la séance prochaine vous sera délivré. Si vous ne participez qu'à 2 séances, vous serez pénalisés de 3 points. Une seule participation c'est 6 points de moins. Aucune participation c'est 9 points de moins.