Principes de programmation 2.
Projet 2023-24 : modélisation et implémentation d'un jeu de société.

Objectifs :

- **Apprendre à bien choisir une modélisation pertinente et efficace.**
- **Apprendre à bien choisir une découpe algorithmique simple et efficace.**
- **Jouer avec les structures.**
- **Manipuler les fonctions et procédures et les passages des paramètres.**
- **Améliorer sa rédaction de spécifications.**
- **Réaliser des batteries de tests unitaires.**
- **S'amuser !**

Contexte :

Rush Hour est un jeu de société qui se joue sur un plateau où l'on place des véhicules selon un modèle présent sur une carte. Le but du jeu est simple : il s'agit de se frayer un chemin hors de l'embouteillage avec la voiture rouge. Pour ce faire, le joueur peut déplacer les autres véhicules présents sur le plateau en les déplaçant uniquement vers l'avant ou vers l'arrière. Selon la direction dans laquelle chaque véhicule a été placé au départ, ça le fera soit monter ou descendre, soit aller de gauche à droite. Le plateau de jeu est divisé en cases et chaque véhicule couvre un certain nombre de cases (2 cases pour les voitures et 3 cases pour les camions). Il y a plusieurs niveaux de difficulté : Débutant, Intermédiaire, Avancé, Expert. La figure 1 ci-dessous nous montre un exemplaire du jeu :

Figure 1 : jeu Rush Hour

À la figure 2, voici un exemple de défi résolu étape par étape afin de faire sortir la voiture rouge du plateau. La première étape consiste à faire avancer la voiture orange (elle monte). La seconde étape déplace le camion vert en arrière (il va donc vers la gauche). La troisième étape recule le camion bleu (ici il descend de 2 cases). Et la dernière étape consiste à déplacer la voiture rouge jusqu'à la sortie du plateau.

Figure 2 : sortir la voiture rouge, étape par étape.

Ce jeu existe aussi en ligne, testez-le : https://www.mahjongfun.com/fr/rush-hour/

Avec cette version numérique, le nombre de déplacements réalisés pour atteindre son but est comptabilisé. On peut ainsi le comparer au nombre minimal de déplacements nécessaires pour atteindre le but et donner la possibilité d'améliorer sa performance.

Règles du jeu pour notre application informatique

Premièrement, le joueur choisit une carte. Cette carte sera lue dans un fichier. Ceci initialise le plateau en y plaçant les divers véhicules.

Ensuite, le joueur déplace un véhicule à la fois. Il répète cela jusqu'à parvenir à faire sortir la voiture rouge du plateau. Pour déplacer un véhicule, le joueur précise la couleur du véhicule souhaité, le sens de déplacement et le nombre de cases du déplacement. Si l'utilisateur entre des données non valides ou un déplacement impossible, le mouvement est ignoré. Sinon le déplacement est réalisé et comptabilisé. Une partie s'arrête quand l'utilisateur a sorti la voiture du plateau ou qu'il a montré son souhait d'abandonner. Le programme s'arrête quand l'utilisateur demander de le quitter.

Remarques :

- Le joueur doit avoir, à tout moment, la possibilité de remettre le plateau dans son état initial. Ceci remet le compteur du nombre de déplacements à 0.
- En cas de fin de partie (donc soit une victoire, soir un abandon), le joueur doit avoir la possibilité de rejouer le même plateau ou de passer à un autre plateau.
- Si le joueur parvient à sortir la voiture rouge, alors son score est sauvegardé dans un fichier. Le score est le nombre de déplacements de véhicule que l'utilisateur a réalisé afin de gagner la partie. Ceci permettra au joueur rejouer un même plateau en espérant s'améliorer afin de se rapprocher du nombre de déplacements minimal.
- À chaque fois que le joueur parvient à sortir la voiture rouge du plateau, un classement de ses scores pour le plateau concerné doit être affiché.

- Les plateaux sont sauvegardés l'un à la suite de l'autre dans 4 fichiers (un fichier par niveau). Ainsi, tous les plateaux de niveau « débutant » sont sauvegardés l'un à la suite de l'autre[1] dans le fichier nommé `rushHourDebutant.txt`.
  Et nous avons la même configuration pour les fichiers `rushHourIntermediaire.txt`, `rushHourAvance.txt` et `rushHourExpert.txt`

Ce que vous devez faire en pseudocode :

1. Définir les structures dont vous avez besoin.
2. Réaliser une découpe en procédures et fonctions (donc uniquement penser aux signatures).
3. Spécifier les procédures et fonctions le plus clairement possible (on peut les écrire assez facilement formellement).
4. Réaliser une batterie de tests unitaires par fonction/procédure.
5. Réaliser le programme principal qui appelle vos fonctions/procédures.
6. Implémenter en pseudocode les procédures et fonctions.

Échéance 1 :

Attention : pour cette échéance, vous ne devez **pas coder le contenu** des procédures et fonctions. Il ne faut que les signatures.

Définir des groupes de 2 étudiants. Le projet peut aussi se faire seul.

Pour le vendredi **22 mars** vous devez rendre un rapport au format PDF incluant

- Une page de garde comprenant, entre-autres, le nom des membres du groupe.
- Une introduction.
- Votre modélisation. C'est-à-dire vos choix de structures avec explications de celles-ci.
- Un programme principal en pseudocode qui montre l'enchainement du processus de jeu avec appels à vos fonctions et procédures.
- La liste des fonctions et procédures. Pour chacune, vous devez donner :
  - La spécification avec pré-post-résultat. Cette spécification ne doit pas être formelle mais vous pouvez tenter malgré tout.
  - La signature.

Vous pouvez ajouter ce que vous souhaitez à ce rapport que vous jugeriez pertinent.

[1] dans un format que nous décrirons plus tard