Bonus Javass

Auteurs: Loïc Houmard (297181)
Antoine Masanet (288366)

Dans cette partie bonus, nous avons mis en œuvres deux améliorations au projet de base: la possibilité de choisir l'atout ainsi qu'un menu graphique pour initialiser les paramètres ayant trait au lancement d'une partie.

Choisir l'atout

Point de vue de l'utilisateur

Grâce à cette amélioration, l'utilisateur a la possibilité, quand c'est à lui de faire atout, de cliquer sur celui qu'il désire sélectionner ou bien de cliquer sur le bouton <chiber> s'il souhaite laisser son partenaire faire atout. Ce bouton <chiber> n'est que visible et actif si le partenaire de l'utilisateur n'a pas déjà chibé. Ce processus fonctionne aussi bien avec un partenaire local que distant ainsi qu'un joueur simulé. Ce dernier possède un algorithme lui permettant de choisir l'atout ou chiber si son espérance de points gagné lors de ce tour n'est pas suffisamment élevée. De plus, nous avons rajouté une croix à côté du nom du joueur ayant décidé de chiber ainsi qu'une image de l'atout choisi à côté du nom du joueur ayant choisi l'atout.

Mise en œuvre

Chaque joueur devant choisir l'atout quand c'est à lui de jouer, nous avons rajouté la méthode chooseTrump(Cardset hand, boolean canPass) en public abstract à l'interface Player pour que chaque classe implémentant Player ai à la redéfinir.

Pour la mise en œuvre d'un point de vu graphique, nous avons rajouté un StackPane contenant la partie en cours ainsi que l'affichage du choix de l'atout. Ainsi, nous pouvions "binder" la visibleProperty de l'affichage de l'atout de manière à ce que le Pane du choix de l'atout ne s'affiche seulement lorsque c'est au joueur de faire atout. Par la suite, les joueurs savent aussi qui a pris atout (l'atout à la droite de son nom) et qui a chiber (croix à la droite de son nom) lors du tour actuel. Pour cela, nous avons rajouté les commandes FTPL (firstTurnPlayer) et TRCH (trumpChooser) qui indique aux joueurs les informations correspondantes.
Pour le cas du remotePlayer, nous avons rajouté la commande CHTR (chooseTrump) ainsi qu'un booleanSerializer et deserializer pour le boolean canPass.

La partie intéressante de ce bonus est de permettre au joueur simulé de faire atout. Pour choisir l'atout, le MctsPlayer simule donc "itérations/4" parties pour chaque atout, avec la même création de nœuds sous-jacent que la méthode cardToPlay. A la fin de la simulation, on récupère pour chaque atout, le nœud avec la plus grande espérance et puis en les comparant, on détermine quel atout présente le plus de potentiel. Pour l'option chiber, nous avons fait le choix de simuler sur un ordinateur un très grand nombre de tour où un MCTS choisit l'atout et joue la partie correspondant avec des MCTS de même niveau. Suite à cette expérience, nous avons pu déterminer qu'en choisissant l'atout avec notre algorithme, un MCTS fait en moyenne 100,46 points par tour. Par conséquent, le MCTS chibe si il en a le droit (canPass) et si sa meilleur espérance en prenant l'atout est en dessous de 100,46 et choisit l'atout sinon.

Menu de lancement

Point de vue de l'utilisateur

Le menu de lancement permet à un utilisateur de lancer une partie local ou remote sans avoir à modifier les arguments dans les Run Configurations. Pour cela, lors du lancement du programme, le joueur lance une seule classe "Play" qu'il soit remote ou local. Après le lancement de cette classe, une fenêtre graphique Javass-Launcher s'ouvre où le joueur peut rentrer les paramètres de configuration de sa partie puis cliquer sur le bouton <start> quand il a fini de rentrer les paramètres. Similairement à la partie précédente, les mêmes valeurs par défaut sont conservées lorsque le joueur ne rentre rien. Finalement, le menu est dynamique, c'est à dire que lorsque l'utilisateur sélectionne une partie remote, tous les "textField" et "choiceBox" disparaissent pour seulement laisser

Mise en œuvre

Lors du lancement de la main de Play, celle-ci crée un graphicalLauncher et l'affiche. Celui-ci s'occupe de récupérer les infos fournies par le joueur lorsque celui-ci clique sur <start>. Après les paramètres de partie récupérer, le graphicalLauncher appelle la méthode startGame de Play. Cette dernière, à l'aide du premier argument décide de lancer une remoteGame ou une localGame qui se comportent respectivement comme remoteMain et localMain. Nous avons conservé les classes remoteMain et localMain pour pouvoir lancer des parties plus configurées avec des paramètres telles que la graine qui n'est pas accessible lors d'un lancement standard.