SCIENCES SORBONNE UNIVERSITÉ

POLYTECH SORBONNE

RAPPORT DE PROJET

Programmation C

Jeu de société :

LES AVENTURIERS DU RAIL

But :

Codage d'un joueur automatique

ROYER Jérémy EI-SE 3

Introduction

Pour réaliser un joueur complètement autonome du jeu *Les Aventuriers du Rail* (*Ticket To Ride*), le développement est passé par plusieurs étapes. Tout d'abord la réalisation de plusieurs structures de données afin de récupérer toutes les informations du jeu comme le plateau, les joueurs, etc... Puis il a fallu faire l'initialisation de toutes ces données au début de chaque partie. Après l'initialisation, il y a eu la mise à jour des données à chaque tour. Enfin avant de réaliser un joueur automatique, il fallait d'abord créer de quoi jouer nous-même grâce au terminal. Et seulement après tout ça, la création du joueur automatique en lui-même a pu se faire.

Fichiers du projet

Les différentes phases d'écriture décrite brièvement précédemment vont être repréciser ici.

Pour les **structures de données** que j'ai créées donc sans parler de celles mise à disposition préalablement, il y a une structure représentant **les routes** contenant la ville de départ et d'arrivée, la longueur, les deux couleurs et si elle est prise par un joueur (-1 non, 0 nous, 1 adversaire). La deuxième structure représente **le plateau de jeu** contenant simplement le nombre de villes et de routes et un tableau de routes. Ensuite une structure est dédiée pour caractériser **un joueur**, elle permet de stocker son nombre de wagons et de cartes, ses quatre premières cartes, son nombre de carte pour chaque couleur (dans un tableau), son nombre d'objectifs et la liste de ces derniers. Et enfin la dernière structure et la plus important est celle représentant **le jeu** et contenant le nom de la partie, les cinq cartes face visible de la pioche, le plateau de jeu, un entier représentant le joueur du tour (0 nous, 1 adversaire), un tableau de 2 joueurs (l'adversaire et nous), un tableau contenant toutes les routes du jeu, un tableau contenant la longueur de chaque route identifiée par ses villes aux extrémités, un entier représentant le nombre de route nécessaire à prendre pour réaliser nos objectifs et un dernier tableau de route contenant toutes les routes à prendre pour réaliser nos objectifs.
Toutes ces structures se retrouvent dans le fichier **structGame.h** .

Pour **l'initialisation des données**, tout est fait dans le fichier **createGame.c** avec une fonction générale qui crée le jeu avec les fonctions mise à disposition (getMap, waitForT2RGame, connectToServer, …) et qui appelle deux autres fonctions qui initialise les joueurs et les routes.

La **mise à jour des données** du jeu est réalisée par les fonctions se trouvant dans le fichier **update.c**, de manière analogue à l'initialisation dans une fonction principale est appelé la fonction correspondante au coup du joueur. Et donc il y a une fonction pour mettre à jour les données en fonction du coup joué (tirer des objectifs, choisir des objectifs, tiré une carte visible, une carte cachée, prendre une route).

Ensuite dans un fichier, il y a les fonctions qui permettent de **jouer nous-même** avec le terminal. Une fonction qui demande le coup qu'on veut jouer, une qui rempli et envoie le coup à jouer (elle sera utilisée également par le joueur automatique) et une dernière qui vérifie s'il

faut qu'on rejoue (après avoir pioché des objectifs ou piocher une carte si c'est la première). On peut retrouver ces trois fonctions dans **move.c** .

Et enfin la plus grande partie et celle la plus intéressante également est le fichier contenant toutes les fonctions pour faire **un joueur automatique**. Ce fichier s'appelle **autoMove.c** . Il est décomposé en un total de 7 fonctions dont une principale. Dans l'ordre d'apparition dans le ficher, **les trois premières permettent de trouver le chemin à prendre entre deux villes**. On retrouve une fonction permettant de trouver la ville la plus proche qui n'a pas encore été visité (elle est utilisée dans un l'algorithme de la prochaine fonction). La deuxième est donc un algorithme permettant de trouver le plus court chemin entre deux villes avec la liste de toutes les villes par lesquelles il faut passer. La troisième fonction permet de déduire du résultat de l'algorithme la liste des routes à prendre pour relier deux villes (et donc accomplir un objectif). Maintenant **les 4 dernières fonctions permettent de choisir quel coup on va jouer et comment**. Il y a donc, toujours dans l'ordre, une fonction permettant de choisir les objectifs que l'on sélectionne selon certains critères (la stratégie sera expliquée dans un autre paragraphe). Ensuite une fonction qui nous permet de piocher soit dans les cartes visibles soit dans la pioche. Puis une fonction qui permet de prendre une route. Et enfin la fonction principale permettant de choisir selon la stratégie qui sera évoquée en dessous le coup à jouer par notre joueur automatique.

Evidemment, un **fichier principal play.c** permettant de regrouper tous ces fichiers et contenant le *main()* est présent. Il contient la boucle de jeu principale et l'appel aux fonctions nécessaires.

Stratégie mise en place

La stratégie déployer par le joueur automatique se déploie surtout sur deux points, le choix des objectifs et la prise des routes.

Tout d'abord, pour le **choix des objectifs** au départ on prend tout le temps les trois objectifs tirés sauf si la ville de Miami est dans les objectifs car je l'estime trop compliqué (souvent le joueur automatique ne réussissait pas à réaliser les objectifs avec Miami). Ensuite lors des prochains choix d'objectifs au cours de la partie, le raisonnement se base sur le nombre de wagons à poser pour réaliser l'objectif et le nombre de wagons restant. Et si par rapport à ces critères je choisis les trois objectifs ou zéro, dans le premier cas j'enlève celui rapportant le moins de points et dans le deuxième cas je prends celui rapportant le moins de points.

Je vais présenter brièvement la stratégie pour **piocher des cartes** qui n'a pas été la plus travailler. Si le joueur automatique doit piocher, le choix se déroule en 3 points, d'abord pour les deux premières routes qu'il doit poser, il regarde s'il y a la couleur 1 des routes face visible puis la couleur 2 en ne prenant pas en compte les locomotives. Ensuite il regarde s'il a moins de 2 locomotives puis s'il a plus de 3 cartes de la même couleur et dans ce cas il pioche la locomotive. Sinon il pioche une carte face cachée.

Maintenant le plus gros de la stratégie se déroule dans la **pose des routes**. Beaucoup de conditions sont vérifiées dans un ordre précis pour savoir la route que le joueur automatique

va poser. Tout d'abord à chaque tour, on recalcul toutes **les routes qu'on doit prendre** en prenant le **plus court chemin** possible car il peut changer si notre adversaire pose une route. Puis avant de commencer à voir si on peut poser une route, on regarde si on doit piocher un objectif (plus qu'une route ou zéro à prendre et un nombre de wagons suffisant chez l'adversaire et nous). Après ça, **différentes conditions** sont examinées. On parcourt toutes les routes que l'on doit prendre, et on s'occupe d'abord des **routes qui ne sont pas multicolores**. Sur ces dernières on regarde si on peut la poser avec la couleur 1 puis la couleur 2 sans locomotive et seulement si on ne peut pas, on regarde si avec des locomotives on peut la prendre dans le cas où la route a une longueur d'au moins 4. Si après ça on n'a pas pu poser de route, on regarde **les routes multicolores** et on pose une route comme précédemment sans locomotives puis avec.
Maintenant, si le joueur automatique n'a malheureusement toujours pas pu poser de route, on regarde si on peut poser une **route de 6 wagons** (le maximum) de couleurs différentes de la première route que l'on souhaite théoriquement poser. Et si après ça, on n'a toujours pas pu poser de routes et qu'il nous reste moins de 6 wagons ou plus d'objectifs alors on regarde si on peut prendre une route de la plus grande longueur possible.
Après toutes ces conditions vues, il nous reste plus qu'à piocher.
Mais si nous sommes au **dernier tour** parce que l'adversaire à moins de 5 wagons alors on vérifie si on ne peut pas **poser n'importe quelle route** (la plus grande possible) quand même.

Voilà comment la stratégie du joueur automatique fonctionne. Elle pourrait être amélioré avec une modification de l'algorithme ne cherchant pas le plus court chemin mais le plus rentable entre la distance et les points. Par exemple préférer poser 2 wagons de plus si c'est pour prendre une route de 4 et de 5 plutôt que prendre deux routes de 2 et une de 3.