Rapport de projet

INFORMATIQUE 1 : Langage C

Sujet : Ecrire un programme capable de jouer de façon autonome au jeu *Les Aventuriers du Rail*, en respectant les règles du jeu.

Auteure : Mme. Jade EVRARD

Encadrant : M. Thibault HILAIRE

Année universitaire : 2020/2021

Introduction :

Tout d'abord, le jeu *Les Aventuriers du Rail* se joue sur un plateau du réseau ferroviaire d'Amérique du Nord dans sa version originale (d'autres plateaux ont ensuite été développés). Lors de la mise en place du jeu en début de partie, le jeu est composé de 144 cartes wagon de 9 couleurs différentes (dont une couleur est considérée comme un joker, il s'agit de la carte Locomotive qui peut être utilisé à la place de n'importe quelle couleur). Les joueurs disposent de 45 wagons au début de la partie dont ils se serviront pour marquer les routes qu'ils prennent sur le plateau de jeu. Avant de commencer la partie, 3 cartes Destination sont distribuées à chaque joueur (il doit en garder minimum 2 pour commencer le jeu).

Le but du jeu est d'obtenir le maximum de points possible. Pour cela, les joueurs doivent réaliser les objectifs qu'ils ont choisi de gardé (si un objectif n'est pas réalisé, le joueur soustrait à son score le nombre de points attribués à l'objectif qu'il n'a pas pu réaliser). Pour obtenir de nouvelles cartes wagons, deux choix s'offrent aux joueurs : soit ils piochent une carte à l'aveugle soit ils ont à disposition 5 cartes retournées à côté du plateau (dès qu'un carte est prise, elle est immédiatement remplacée par une carte de la pioche). A la fin de la partie, le joueur qui a réalisé le plus long chemin remporte 10 points bonus.

I. Mise en place des structures et tableaux nécessaires

Avant de réfléchir à une stratégie de jeu, il est important de mettre en place les éléments dont on va avoir besoin par la suite. Nous disposons déjà des types et structures du fichier *TicketToRideAPI.h* (types : t_color et t_typeMove ; structures : t_objective, t_claimRouteMove, t_drawCard, t_drawBlindCard, t_drawObjectives, t_chooseObjectives et t_move).

Par ailleurs, j'ai ajouté plusieurs nouvelles structures (elles se trouvent dans le fichier *jeu.h*) :

- Une structure **t_board** qui contient deux entiers correspondants au nombre de villes et au nombre de routes présentes sur le plateau de jeu ainsi qu'un tableau d'entiers représentant toutes les routes du plateau.
- Une structure **t_track** qui est composé de 4 entiers (les deux villes qui composent la route, la longueur de la route et un booléen nous indiquant si la route est déjà prise ou non) et de 2 t_color (les deux couleurs de la route dans le cas où elle est double, ou bien la couleur de la route et NONE pour la deuxième couleur si elle est simple).
- Une structure **t_player** qui comporte le nom du joueur, le nombre de wagons et le nombre de cartes qu'il a encore en sa possession, les cartes initiales qu'il a au début du jeu, un tableau d'entiers indiquant le nombre de cartes que le joueur possède pour chaque couleur, le nombre d'objectifs que le joueur a en sa possession et enfin un tableau d'objectifs qui comporte le descriptif des objectifs que le joueur doit réaliser.
- Une structure **t_game** qui se constitue du nom de la partie en cours, des cartes visibles à côté du plateau de jeu, d'un entier player indiquant quel joueur est en train de jouer, d'un tableau de 2 t_player avec toutes les informations relatives à chaque joueur et enfin d'un t_board contenant les informations décrites au-dessus.

Ensuite, j'ai créé dans le fichier principal de mon projet (*jeu.c*) un tableau de t_track à partir du tableau arrayTracks généré par la fonction getMap et permettant d'avoir toutes les informations sur tous les chemins du jeu. J'ai également créé un tableau d'entiers G à 2 dimensions permettant de connaître la longueur de la route entre deux villes (tableau qui servira pour trouver le plus court chemin entre 2 villes).

II. Fonctions utilisées

En premier lieu, nous avons déjà à disposition toutes les fonctions réalisées dans le fichier *TicketToRideAPI.c*.

J'ai ensuite créé des fonctions qui permettent d'initialiser le jeu en début de partie dans le fichier *game.c* :

- void **createGame**(t_board* board,t_game* game,t_player* playerMe) permet de créer le jeu.
- int **createMap**(t_board board, t_game* game, t_player* player) permet de créer la mappe avec tous les éléments qui constituent le jeu comme par exemple les cartes visibles.
- void **createPlayers**(t_player* playerMe, t_player* playerOpponent) permet de créer les joueurs et d'initialiser leurs données.
- void **displayGame**(t_game game) permet d'afficher toutes les données que l'on veut et qui constitue le jeu.

Dans un fichier *move.c* j'ai ensuite créé les fonctions qui gèrent les coups des joueurs :

- void **askMove**(t_move* move) permet de récupérer le coup que le joueur souhaite réaliser et toutes les données qu'il doit saisir afin que ce coup puisse être réaliser correctement.
- int **needReplay**(t_move* move, t_color lastCard) permet de savoir si le joueur doit rejouer en fonction de son coup précédent (par exemple s'il a choisi de tirer des objectifs il doit rejouer pour pouvoir choisir ses objectifs).
- t_return_code **playOurMove**(t_move* move, t_color* lastMove) permet de réaliser le coup que le joueur a demandé grâce aux fonctions présentes dans *TicketToRideAPI.c.*
- void **updateMove**(t_move move, t_track* tracks, t_game* game, int** G) met à jour les données des tableaux et des structures créés précédemment en fonction du coup que notre joueur vient de jouer.
- void **updateOpponent**(t_move move,t_track* tracks, t_game* game, int** G) met à jour les données des tableaux et des structures créés précédemment en fonction du coup que le joueur adversaire vient de jouer.

En ce qui concerne l'autonomisation de mon joueur, j'ai créé les fonctions nécessaires à cela dans un nouveau fichier *playAlone.c* :

- void **shortTrack**(int Prec[], int src, int dest, int** G, t_game game, t_track* tracks) permet de remplir le tableau Prec qui nous permettra de connaître le plus court chemin entre le point de départ et le point d'arrivée de notre objectif.
- int **distanceMini**(int D[], int Visite[], int N) renvoie l'indice de la ville non visitée la plus proche (cette fonction est appelée dans la fonction shortTrack).

- int **nbTracksToDest**(int src, int dest, int Prec[]) renvoie le nombre de routes qu'il y a parcourir entre la ville de départ et la ville d'arrivée de notre objectif.
- void **tabTracksToDest**(int src, int dest, int Prec[], t_game game, t_track* tracks, t_track* tracksToDest) permet de remplir le tableau tracksToDest qui contiendra toutes les routes à prendre pour réaliser notre objectif.
- int **playAlone**(int fisrtPlay, int* previousReturn, t_move* move, t_move* lastMove, t_game game, int Prec[], int** G, t_track* tracks) permet de choisir le coup que notre joueur doit jouer en fonction des objectifs, des routes déjà prises, etc…

III. Stratégies choisies

Tout d'abord, à chaque fois que c'est à mon joueur de jouer, je m'intéresse au coup précédent. Si le coup précédent était de piocher une carte dans la pioche ou dans les cartes visibles, alors le joueur doit à nouveau piocher. En ce qui concerne cette pioche, je vérifie tout d'abord où en sont les objectifs que j'ai en main, si on a fini nos objectifs alors on choisit de piocher à l'aveugle. Cependant, si les objectifs ne sont pas encore atteints, alors on regarde si une carte présente dans les cartes visibles possède une couleur qui nous intéresse (par rapport aux routes que l'on doit prendre). Si c'est le cas, on choisit cette carte, sinon on pioche à l'aveugle.

Ensuite, si le coup précédent était « DRAW_OBJECTIVES » (c'est-à-dire tier de nouveaux objectifs), nous allons d'abord vérifier s'il s'agit du premier coup de la partie pour le joueur ou non. Si c'est le cas alors, il va garder les 3 objectifs qu'il aura piocher, sauf si dans ces 3 objectifs, il se trouve 2 objectifs ayant un score supérieur ou égal à 20 points (on considère que le joueur perdra car ce sont de gros objectifs, il n'aura donc pas le temps de réaliser le 3ème) ou bien s'il se trouve 2 objectifs ayant un score inférieur ou égal à 12 (on considère que le joueur perd son temps avec des objectifs ayant un score pas assez important pour marquer beaucoup de points, il vaut mieux en garder que 2 et re-piocher ensuite en espérant avoir de meilleurs objectifs). Dans les deux cas précédents, le joueur choisira les deux objectifs ayant les 2 plus grands scores. Enfin, si ce ne sont pas les premiers objectifs que le joueur prend alors le joueur gardera uniquement les 2 objectifs rapportant le plus de points.

Dans le cas où le coup précédent n'est ni un DRAW_OBJECTIVES ni un DRAW_CARD ni un DRAW_BLIND_CARD alors il est forcément soit un CHOOSE_OBJECTIVES soit un CLAIM_ROUTE. Dans ces deux cas, on étudie d'abord si les objectifs sont terminés ou pas. Dans le cas où les objectifs ne sont pas encore tous atteints alors on essaie d'abord de le réaliser. Pour cela, on regarde en priorité si on a la possibilité de poser des wagons pour prendre une route non « multicolor ». Si c'est le cas, on la prend, sinon on regarde si on peut prendre une route « multicolor » avec les cartes que l'on a. Si c'est le cas, on la prend, sinon on va devoir piocher. On choisira alors de la même manière que décrit au premier paragraphe de cette partie si l'on prend une des cartes visibles ou si l'on pioche à l'aveugle. Dans le cas où tous les objectifs que l'on a sont réalisés, il faut alors se demander si l'on reprend des objectifs ou pas. J'ai choisi de re-piocher des objectifs dans le cas où mon nombre de wagons restants est supérieur ou égal à 20 et que le nombre de wagons restants de mon adversaire est supérieur ou égal à 15. Ces chiffres sont assez élevés parce que j'ai choisi la stratégie de m'arrêter de réaliser des objectifs assez tôt pour ensuite prendre des routes un peu partout sur le plateau de jeu pour bloquer mon adversaire dans la réalisation de ses objectifs. Dans le cas où nos nombres de wagons restants ne sont pas

suffisants pour que je re-pioche de nouveaux objectifs, je vais alors chercher à prendre le maximum de routes avec mes cartes restantes afin de marquer des points et bloquer mon adversaire. Je vais alors parcourir toutes les routes restantes qui ne sont pas encore prises et je vais regarder si je peux en prendre une parmi elles (en gardant la priorité sur les cartes non « multicolor »). Si c'est le cas, je la prends, sinon je pioche une carte à l'aveugle.

IV. Problèmes rencontrés

Au cours de la réalisation de ce projet, j'ai connu surtout un problème majeur qui se trouve être la gestion de l'allocation dynamique. J'ai bloqué pendant un long moment sur l'allocation de la mémoire du tableau G utilisé dans la fonction shortTrack. Après plusieurs erreurs corrigées, j'ai pu enfin arriver à allouer le bonne mémoire.

Par ailleurs, j'ai eu beaucoup de mal au début du projet à bien m'imprégner de toutes les fonctions, structures et types déjà à notre disposition étant donné que cette partie n'a pas réalisé par nous-mêmes (malgré une bonne lecture du fichier *TicketToRideAPI.h*).

V. Conclusion

En définitive, j'ai réussi à écrire un programme capable de faire jouer mon joueur de manière autonome au jeu *Les Aventuriers du Rail* en respectant les règles du jeu. Mon programme permet de faire gagner mon joueur 90% du temps (9 parties sur 10 remportées) contre PLAY_RANDOM et 80% du temps (8 parties sur 10 remportées) contre NICE_BOT.

Cependant, des améliorations restent toujours à faire. Au niveau stratégie, je pense que le problème majeur de mon programme c'est qu'il ne traite qu'un objectif à la fois (il n'essaie pas de réaliser les autres objectifs tant que l'objectif courant n'est pas rempli). Au niveau code, l'erreur majeure que j'ai faite est dans le codage de la fonction playAlone qui est mal optimisé et dont la lisibilité est compliquée. J'aurais dû faire plusieurs fonctions pour chaque cas cité dans la partie « stratégie » et faire des appels aux fonctions réalisées. Le code aurait été plus propre et plus clair (je m'en suis rendu compte un peu trop tard).

Néanmoins, je suis fière du résultat final obtenu. Au début du projet, je ne m'imaginais pas arriver jusque là et parvenir à mettre en œuvre la stratégie que j'avais en tête.