RODRIGUES Samuel
MEUNIER William
PAESA Théo

Algorithmique et programmation procédurale

Jeu de traverse

3 mai 2020

École Internationale des Sciences du Traitement de l'Information

Table des matières

1 Introduction **2**

2 Description du programme **2**
2.1 Description du jeu . . . 2
2.2 Le menu de démarrage . . . 2
2.3 Gestion des mouvements . . . 3
2.3.1 1re phase . . . 3
2.3.2 2e phase . . . 3
2.3.3 3e phase . . . 3
2.3.4 4e phase . . . 3
2.3.5 5e phase . . . 3

3 Démarche et structure du programme **4**
3.1 les structures . . . 4
3.2 IA . . . 5
3.3 Peaufinage & Problèmes . . . 5
3.3.1 Sauvegarde d'une partie . . . 5

4 Répartition des tâches **6**

5 Bilan personnel **6**
5.1 RODRIGUES Samuel . . . 6
5.2 PAESA Théo . . . 6
5.3 MEUNIER William . . . 6

1 Introduction

Ce projet consiste à coder le jeu de Traverse.

- Tout d'abord, afin de pouvoir jouer à 2 joueurs sur le terminal : le problème étant de réussir à coder un programme qui créé un plateau de jeu et initialise une partie. Puis il faut demander aux joueurs leur coup chacun leur tour. Il faut vérifier si les coups sont conformes aux règles du jeu, sinon il faut demander une nouvelle saisie. Enfin, il faut que le programme termine la partie sur un match nul ou la victoire d'un des 2 joueurs.
- Puis afin de pouvoir jouer contre une IA : le vrai défi de ce projet est cette deuxième partie. En effet, nous n'avions aucune connaissance en IA avant le projet. De plus, ce n'est pas une IA de base qui joue un coup random, mais qui réfléchit du meilleur coup à jouer selon l'algorithme MinMax.

2 Description du programme

2.1 Description du jeu

Le jeu de traverse que nous avons codé se joue uniquement à deux joueurs, joueur VS IA ou bien joueur contre joueur. Le jeu est composé d'un plateau carré de 100 cases et de 16 pions, 8 pour un joueur et 8 pour l'autre joueur. Sur ces 8 pions il y a 4 types de pions :

- Le carré : que nous avons représenté par un "C" peut se déplacer d'une case sur chaque latérale, nord, sud, est et ouest en entrant dans le terminale le chiffre dédié à la direction voulue.
- Le triangle : que nous avons représenté par un "A" peut se déplacer d'une case sur ses deux diagonales avant ou bien d'une case latérale en arrière, sud, nord-est et nord-ouest (et nord, sud-est et sud-ouest pour l'autre joueur).
- Le losange : représenté par un "L" peut se déplacer d'une case sur toutes ses diagonales, nord-est, nord-ouest, sud-est et sud-ouest.
- Le rond : représenté par un "O" peut se déplacer d'une case sur toutes ses cases limitrophes.

A chaque tour le joueur peut décider de bouger un de ses pions d'une case en sélectionnant la coordonnée du pion voulu puis en indiquant la direction de déplacement voulu, si une mauvaise sélection des commandes est entré, une autre sélection est demandé. Il a la possibilité de déplacer le même pion une seconde fois dans un tour si le pion en saute un autre. Le joueur peut décider de sauter son tour en entrant "-1" dans le terminale, ou bien "-2" pour sauvegarder la partie. Le but du jeu est d'être le premier à avoir déplacé tout ses pions de l'autre coté du plateau. Si au bout de 30 tours un joueur possède encore un pion dans sa zone de départ ou s'il passe son tour pendant 3 tours consécutif le joueur est perdant.

2.2 Le menu de démarrage

En exécutant le programme l'utilisateur à le choix entre 4 possibilités :

- Possibilité n°1 : En entrant "1" dans le terminale le mode "normal" est lancé deux joueurs ont la possibilité de s'affronter.
- Possibilité n°2 : L'utilisateur peut jouer contre l'ordinateur.
- Possibilité n°3 : Charge des parties permettant de tester certaines fonctionnalités du programme.
- Possibilité n°4 : La partie ultérieurement sauvegardée est relancée.

2.3 Gestion des mouvements

La gestion des mouvements est quelque chose d'assez simple. Il suffit, pour un pion donné, de récupérer ces coordonnées. Puis calculer ces coordonnées d'arrivées. il ne reste plus qu'à l'enlever du plateau et le replacer au point correspondant aux nouvelles coordonnées.
Nous avons donc découpé le déplacement en 5 phases :

1. Saisie des informations
2. Calcule des coordonnées d'arriver
3. Vérification de la validité des données
4. Suppression du pion sur le plateau
5. Placement du pion aux nouvelles coordonnées

2.3.1 1re phase

Pour la 1re phase on demande au joueur de sélectionner le pion à déplacer en entrant ses coordonnées sur le plateau ainsi que la direction dans laquelle il veut le déplacer.

2.3.2 2e phase

Une fois les informations nécessaires connues on peut calculer les coordonnées d'arrivées. Pour cela, nous avons implémenter la fonction **convertDirection()** qui prend en paramètre les coordonnées actuelles du pion ainsi que la direction dans laquelle le déplacer (cf : deplacement) et renvoie les coordonnées où le pion sera déplacé.

2.3.3 3e phase

Ensuite, il nous faut vérifier que les coordonnées renvoyées sont valides. Pour cela, on test si les coordonnées renvoyées ne pointent pas en dehors du plateau ou vers une case déjà occupée, auquel cas, on redemandera d'entrer les coordonnées au joueur ou on initiera un saut si cela est possible.

2.3.4 4e phase

Une fois que les coordonnées ont été validées, on s'occupe de simplement retirer l'adresse vers le pion à déplacer du plateau.

2.3.5 5e phase

Finalement,il nous suffit de mettre à jour les coordonnées du pion ainsi que de le replacer en ses nouvelles coordonnées sur le plateau.

On réitère ces opérations pour chaque tour. En changeant de joueur à chaque fois.

3 Démarche et structure du programme

Pour pouvoir réaliser notre projet à bien, il a fallu décider d'une démarche sur comment nous allons implémenter l'algorithme du jeu de Traverse. Il à donc été décidé de s'occuper en priorité du mode joueur VS joueur. nous nous sommes par la suite occupé de l'IA et enfin des bonus.

3.1 les structures

Le jeu de traverse nécessite de connaître en permanence l'état du plateau. C'est-à-dire, les pions de chaque joueur (positions, types, appartenance). Ainsi nous avons décidé de stocker l'état du jeu dans la structure suivante :

```
typedef struct
{
    /*! tableau de joueur */
    joueur *joueurs;
    /*! Plateau de jeu (tableau 2D d'adresse
        de pion) */
    pion ***plateau;
    /*! adresse ver le joueur courant */
    joueur *joueurCourant;
    /*! numeros du tour */
    int tour;
} partie;
```

- **joueurs** : un tableau qui stocke une liste de structure de joueurs
- **plateau** : un tableau à 2 dimensions qui en chaque case stocke sois rien (aucun pion n'est à cette endroit sur le plateau) sois un pointeur vers le pion situé en ces coordonnées.
- **joueurCourant** : Le pointeur qui renvoie l'adresse du joueur courant situé dans le tableau joueur.
- **tour** : un entier qui permet de connaître le numéro du tour actuel.

Doxygen : Partie

Ensuite, la structure joueur nous permet de stocker l'état d'un joueur selon la structure suivante :

```
struct joueur
{
    /*! Identifiant du joueur */
    int id;
    /*! Adresse vers le tableau contenant
        les pions du joueur */
    pion *pions;
    /*! Adresse vers le tableau contenant
        les pions du joueur */
    couleur couleur;
    /*! Permet de connaître la zone d'
        arrivée du joueur */
    facePlateau zoneArr;
    /*! Permet de connaître le nombre de
        tours inactif*/
    int inactivite;
};
```

- **id** : est un entier qui nous permet d'identifier le joueur stocké dans cette structure.
- **pions** : est un tableau contenant chaque pion sous la forme d'une structure pion.
- **couleur** : est la couleur du joueur qui de type couleur, une énumération des couleurs disponibles dans le jeu.
- **zoneArr** : est une énumération de chaque coin du plateau (sous forme de direction).
- **inactivite** : est un entier qui nous permet de compter le nombre de tour consécutif ou le joueur n'a pas bougé de pion.

Doxygen : Joueur

Puis la structure pion nous sert à stocker toutes les données relative à un pion :

```
typedef struct
{
    /*! Joueur au quel le point appartient
        */
    joueur *joueur;
    /*! Type du pion */
    typePion type;
    /*! Type du pion */
    listeCoupPossiblePion coupsPossibles;
    /*! Coordonnées du pion sur le plateau
        */
    coord coord;
} pion;
```

- **joueur** : est le joueur auquel le pion appartient que l'on connaît grâce au pointeur joueur qui renvoie vers le joueur lié au pion.
- **type** : est le type de pion qui est de type Pion, soit un losange, un carré, un rond ou un triangle.
- **coupPossibles** : est la structure liste des coups possibles par ce pion c'est-à-dire, les directions qui lui sont possible de réaliser.
- **coord** : est une structure contenant les coordonnées du pion sur le plateau de type (x,y).

Doxygen : Pion

3.2 IA

L'intelligence artificielle a été implémentée grâce à l'algorithme **MinMax** qui nous a était proposé. Bien que lent, il nous suffit car notre but n'est pas de faire de notre IA la plus performante mais plutôt de l'implémenter correctement. L'IA a permis de découvrir de nombreux problèmes d'allocations mémoires. Grâce à Valgrind, on a pu détecter les fuites mémoires. Cependant, à cause de ces différents problèmes, notre IA n'a qu'un seul niveau de difficulté.

3.3 Peaufinage & Problèmes

Nous avons, sur les derniers jours qui nous restaient, passé beaucoup de temps à nous assurer qu'il n'y avait pas de fuite mémoire autrement dit que tout ce qui a était alloué de manière dynamique a bien été libéré quand il faut. Cela a été très fastidieux en particulier avec l'IA qui génère beaucoup de plateau de jeu constitué de très nombreux éléments alloués dynamiquement.
De ce fait, nous avons passé la fin du temps impartis à essayer de coder un IA VS IA, cependant nous avons rencontré des problèmes de segmentation que nous n'avons pas pu résoudre part manque de temps. De même, nous avons préféré nous concentrer sur une interface terminale plutôt que graphique, qui aurait demandé beaucoup de temps.

Enfin, des bonus ont été proposé, notre programme :

1. permet de sauvegarder la partie
2. utilise un makefile pour compiler

3.3.1 Sauvegarde d'une partie

Pour sauvegarder une partie, il nous suffit d'écrire les données d'une partie dans un fichier puis de pouvoir le recharger plus tard. Cependant nous n'avons pas besoin de stocker toutes les données contenues dans la structure partie. Ainsi nous n'écrivons que le numéro du tour, l'id du joueur courant et la liste des joueurs et des pions ainsi que leurs positions sur le plateau. Pour charger une partie, il nous suffit donc d'initialiser une nouvelle partie et d'écrire par-dessus les informations de la partie sauvegarder.

4 Répartition des tâches

Au vu du confinement et de la distance nous séparant, nous avons opté pour GitLab afin de stocker et échanger nos travaux respectivement attribués. De plus, Grâce à VScode, nous pouvions éditer le projet en même temps lorsque nous travaillions au même moment sur ce dernier. Ce qui s'est avéré très pratique en travaillant à distance. Pour la répartition des tâches, nous avons fait en fonction des préférences de chacun : Tout d'abord, pour la vérification d'entrée et autres fonctions de base, nous avons utilisé la bibliothèque de Samuel. Ce dernier s'est également occupé du main. Pour ce qui est du programme principal, nous l'avons découpé en plusieurs parties :

- Théo s'est chargé de l'affichage, de réaliser les déplacements de pions et tester leur validité, différencier les joueurs, tester
- William s'est occupé de placer et enlever (temporairement) un pion du plateau, initialiser le plateau et la partie, tester et les sauts.
- Samuel a opté pour le déplacement de pions en fonction du types et des possibilités(nord,sud,etc...), calculé les coordonnées, l'allocation mémoire, la sauvegarde, charger une partie sauvegardée.

Pour la partie IA, nous avons opté sur un travail plus groupé car les problèmes d'allocations étaient trop compliqués pour un seul d'entre nous.

5 Bilan personnel

5.1 RODRIGUES Samuel

Pour ma part, le projet m'a paru un peu long pour le temps qui nous a été impartis en particulier car nous n'avons jamais eu de cours sur l'Intelligence Artificielle et donc découvrir le fonctionnement et les principes liés à cette fonctionnalité a été plutôt fastidieux. D'un autre côté cela m'a permis de m'améliorer dans la recherche d'informations et aussi j'ai appris à mes dépends qu'il était important de bien structurer son code à la fois pour qu'on puisse mieux se retrouver sois même mais aussi pour que d'autres puissent le comprendre facilement.

Ce projet m'a aussi permis de découvrir de nouveaux outils comme Valgrind et Git qui sont très utiles pour programmer en particulier Git qui nous a permis de faciliter l'échange du code et d'informations liées au projet. Valgrind est aussi un outil très puissant en C qui nous a permis de corriger nos problèmes de fuite mémoire.

5.2 PAESA Théo

De mon point de vue, ce projet était très long car nous avons eu moins de temps qu'au premier semestre mais moins compliqué que l'algorithme de Huffman. Malgré une bonne répartition des tâches, nous n'avons pas pu faire beaucoup de bonus. Même si nous avons terminer la partie principale à temps. Les problèmes d'allocations mémoire ont été très long à résoudre et m'ont permis de me rendre compte de la rigueur du codage d'un jeu de ce type. La manipulation des pointeurs et des structures tout au long du projet m'a permis d'avoir une plus grande aisance. J'ai eu un peu de mal avec les règles, au début, mais une fois comprises le code en découle sans grandes difficultés. Le vrai challenge a été la réalisation de l'IA. Vu que nous n'avons pas eu de cours dessus, il a fallu comprendre seul. Heureusement Samuel était là pour nous expliquer. C'est donc grâce à une concertation sur le travail de chacun que nous avons pu avancer. La fin du projet consistant à tester le jeu que nous avions codé a été amusante et très satisfaisante.

5.3 MEUNIER William

La partie principale du projet était abordable vis à vis de la complexité et du temps mais il nous a resté très peu de temps pour faire les bonus, du à une organisation et une répartition des tâches à distance et certaines explications manquantes. L'outil en ligne GitLab, nous a permis un partage des travaux efficace avec la possibilité de modifier le code de chacun facilement. J'ai trouvé ce projet polyvalent, complet et utile par sa nécessité de savoir maîtriser plusieurs notions du langage plus ou moins abordé au cours de l'année (tableaux, IA, traitement de fichier).