**DUGAUQUIER Thibault L3BI**

**HERRMANN Lara**

***Projet de Programmation en C 2018-2019***

***Barca***

1. **Choix de conception et d’implémentation**

La base de notre projet est notre plateau, codé par une matrice pointant sur une structure Case. Ce tableau nous sert de support pour toutes les informations relatives au jeu au cours de la partie. La structure Case va comporter toutes les informations relatives aux pièces jouées et à leur état, tel que :

* Le type (soit une pièce, soit une case vide)
* La couleur,
* Si la pièce est positionnée sur un point d’eau,
* Si la pièce est effrayée.

Le plateau est alloué dynamiquement en fonction des dimensions choisies par l’utilisateur (cf 2. Extensions réalisées). Quant aux points d’eau, ils sont toujours positionnés aux mêmes endroits au centre du plateau. Nous avons décidé que leur écart n’était pas proportionnel à la taille du plateau.

Les pièces sont disposées une à une sur le plateau, toujours aux mêmes places, centrées sur les extrémités supérieures et inférieures du plateau. Il existe trois types d’animaux, ceux de base : l’éléphant, la souris et le lion. Elles sont capables de s’effrayer mutuellement selon les règles du jeu.

Les joueurs sont des structures définies par :

* un identifiant,
* une couleur,
* son nombre de points d’eau pour mettre fin à la partie,
* son tour pour déterminer quel joueur doit jouer,
* un booléen ordinateur, dans le cas où le jeu est en mode 1 joueur.

Tous les déplacements sont gérés par la fonction déplacementPossible, très conséquente mais permettant de vérifier si un joueur peut effectuer un déplacement. Elle permet de vérifier le positionnement des pions dans les dimensions du plateau, si le pion sélectionné est de la bonne couleur, si le pion ne sera pas effrayé etc … Si toutes les conditions ne sont pas réunies, un message d’erreur correspondant est affiché.

Finalement, notre programme étant conséquent, nous avons décidé de réaliser une répartition du code en plusieurs fichiers avec en-tête (cf 2. Extensions réalisées).

1. **Extensions réalisées :**
   1. **Définition du plateau**

Nous avons fait en sorte que le joueur puisse définir les dimensions du plateau dès le début de la partie. Pour cela, nous avons imposé que les dimensions soient paires pour pouvoir disposer aisément les pièces. Il a ensuite été possible d’allouer dynamiquement le plateau en fonction des dimensions choisies par l’utilisateur dans la fonction allouerPlateau. Nous n’avons donc pas eu à utiliser un DEFINE au début du programme.

* 1. **Répartition du code en plusieurs fichiers avec en-têtes**

Une autre extension que nous avons réalisée est la répartition du code en plusieurs fichiers avec en-têtes. Cette extension a été nécessaire face à la taille de notre programme. Nous avons décidé de le répartir en 7 fichiers .c regroupant les principales unités de notre programme, à savoir :

* l’affichage (du plateau, des règles, de certains messages relatifs à l’avancée de la partie),
* les fonctions usuelles non essentielles,
* le jeu permettant le suivi de l’avancée de la partie,
* les joueurs et la modification de leurs paramètres,
* le plateau et tout ce qui permet son initialisation,
* les vérifications permettant le codage du déplacement d’une pièce et ses restrictions,
* le main.

1. **Algorithmes non triviaux utilisés**

Nous avons réalisé différentes fonctions peu évidentes, telles que :

* la fonction d’affichage, très conséquente, qui permet d’afficher chaque case indépendamment pour former le plateau. Pour afficher proprement les nombres en abscisse et en ordonnée, nous avons utilisé la fonction nbEspaces, qui renvoie le nombre de caractères d’un nombre, ce qui permet à posteriori de savoir combien d’espace mettre dans l’affichage pour éviter un décalage.
* la fonction CoordonnéesOrdi, permettant le déplacement des pièces de l’ordinateur. Elle détermine d’abord s’il y a des pièces effrayées à déplacer. Si ce n’est pas le cas, elle détermine les pièces appartenant au joueur ordinateur et déplace l’une d’entre elles aléatoirement. Dans un second temps, en choisit une case du tableau au hasard pour la déplacer. Cette fonction nécessite ainsi de nombreux parcours du plateau.

1. **Problèmes non résolus**

Nous avons testé notre programme tout au long du projet. Cela nous a permis de vérifier toutes les possibilités de jeu envisagées pour corriger les différentes erreurs. Ainsi nous pensons avoir résolu toutes les erreurs de notre algorithme. Notre jeu est donc capable de dérouler une partie sans problème.

Beaucoup de nos fonctions effectuent des parcours du plateau, il aurait sûrement été possible de les limiter.

Cependant, certaines de nos fonctions sont redondantes, notamment celles du jeu contre l’ordinateur puisque nous avons réécrit les fonctions coordonnées et déplacementPossible pour les adapter à ce type de parties, ce qui surcharge nos fichiers de code.

De même, des fonctions ne sont pas optimisées, en particulier la fonction coordonnéesOrdi, car la boucle qui permet de vérifier le déplacement tourne tant que les coordonnées fournies ne sont pas correctes. Il y a une faible probabilité que celle ci tourne à l’infini.