Rapport de projet en IN104

Alexandre Laleu et Matéo Martin

21 mai 2024

Table des matières

1 Introduction

Le jeu du Bobail est un jeu de société traditionnel originaire d'Afrique de l'Ouest. Il se joue à deux joueurs, chacun disposant d'un ensemble de pions qu'il déplace sur un plateau à cinq lignes et cinq colonnes. Une partie du jeu est remportée lorsqu'un joueur parvient à coincer le bobail entre les pions ou à l'amener dans sa zone de jeu.

Dans le cadre de ce projet, l'objectif est d'implémenter le jeu du Bobail africain en langage C. Cette démarche vise principalement à acquérir la capacité de travailler de manière autonome, dans ce langage de programmation, sur un projet complet. En outre, ce projet sert de plateforme pour explorer plusieurs aspects du développement logiciel, tels que la gestion des cas d'échec, l'écriture de fonctions de test pour assurer la fiabilité du code, ainsi que l'utilisation de GitHub pour le contrôle de version et la collaboration. À travers cette initiative, nous cherchons à renforcer nos compétences en programmation tout en découvrant les meilleures pratiques de développement logiciel.

2 Détails du projet

2.1 Structure du code

2.1.1 Structures de données utilisées

Le Bobail se jouant sur un plateau, le choix d'une matrice allouée dynamiquement pour le représenter a logiquement été fait. Les pions des joueurs sont alors repérés par des entiers de 1 à 10 tandis que le bobail est représenté par la valeur -1. Cela permet de localiser chaque pion dans notre code, de sorte à modifier le plateau au cours du temps et l'afficher dans le terminal de l'utilisateur.

2.1.2 Arborescence de fichiers

Le dossier a été organisé avec un fichier principal nommé bobail.c, des fonctions annexes séparées en un ficher .h (head) de déclaration et un fichier contenant leur script, ainsi qu'un makefile pour la compilation et l'éxécution.

2.2 Algorithmes

Plusieurs fonctions différentes sont nécessaires au bon déroulement du jeu, à savoir celles qui permettent d'examiner la légalité d'un coup, celles qui vérifient les conditions de victoire, celles qui initialisent le jeu, l'affichent et le ferment, et enfin les autres qui ne sont que des intermédiaires pour raccourcir et simplifier les codes.

2.2.1 is_legal_bis

Cette fonction est la plus longue du projet, en ce qu'elle fait elle-même appel aux fonctions *get* et *is_path_empty*. C'est celle qui examine la légalité des coups joués. *get* permet d'obtenir la position d'un pion sur le plateau, elle est simplement constituée de deux boucles for, tandis que *is_path_empty* permet de vérifier que le mouvement d'un pion se fait bien par la traversée de cases toutes vides. En effet, les pions ne peuvent pas sauter les uns par dessus les autres. Ainsi, *is_legal_bis* vérifie que le joueur déplace bien sa pièce, qu'elle ne passe que par des cases vides, qu'elle reste bien sur le plateau, que les déplacements sont maximum (c'est à dire qu'ils se font jusqu'au bout d'une ligne ou d'une colonne si possible, ou jusqu'à un autre pion) et enfin que les mouvements du bobail qui sont soumis à des règles particulières sont eux aussi corrects.

2.2.2 print_board

Ici se fait l'affichage du plateau de jeu dans le terminal de l'utilisateur. Les pions du joueur humain sont des pions verts, ceux de la machine des pions rouges et le bobail un B jaune. Les cases sont quadrillées de A à E et de 1 à 5.

FIGURE 1 – Plateau de jeu initial

2.2.3 destroy_board

Libération de l'espace mémoire utilisé pour la matrice contenant les données du plateau de jeu.

2.2.4 modify_board

Modification du plateau de jeu en fonction de la position initiale de la pièce passée et de sa position finale passées en argument. Cette fonction fait appel à la fonction *get*.

2.2.5 swap

Échange les joueurs via un passage par adresse, utilisée à la fin de d'un tour de jeu dans le code principal *bobail.c*.

2.2.6 test_end

Fonction vérifiant, à l'issue de chaque tour de jeu, si les conditions de victoire sont remplies.

2.2.7 bobail

Algorithme principal qui éxécute le jeu et coordonne toutes les autres fonctions. C'est ici que se fait l'allocation mémoire de la matrice du plateau de jeu, et que le joueur et l'ordinateur s'affrontent. Ils rentrent les pions qu'ils veulent déplacer dans le terminal grâce au quadrillage affiché, et indique l'endroit où ils veulent le placer. Chacun des pions possède un numéro devant être indiqué par l'utilisateur pour pouvoir être déplacé. Les coups joués par la machine sont systématiquement commentés, et si l'utilisateur tente un coup interdit ou fait une erreur, il en est alerté.

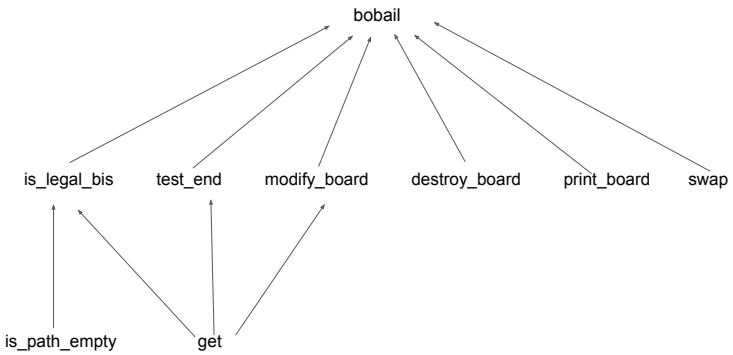

FIGURE 2 – Arborescence de fichiers

2.3 Intelligence artificielle

L'ordinateur joue chacun de ses coups au hasard. Comme c'est le joueur humain qui commence la partie, l'ordinateur joue toujours le bobail et une de ses pièces. Nous avons pensé à utiliser l'algorithme *Min-Max* pour une intelligence artificielle plus poussée mais nous avons manqué de temps pour l'implémenter.

2.4 Difficultés rencontrées

La plus grande difficulté que nous avons rencontré réside sans aucun doute dans la manipulation de Git. Au début, nous ne parvenions pas à identifier les fonctions utiles, mais ce problème s'est logiquement résolu rapidement. Le plus complexe, c'était de bien s'organiser de sorte à accepter toujours la version la plus avancée des fonctions sur lesquelles nous travaillions simultanément. L'un de nous pouvait apporter des modifications différentes de l'autre, il fallait bien s'entendre sur la justesse de celles-ci ou non, et la résolution des conflits seules sur Vs Code n'était pas suffisante. Il était préférable d'en parler à vive voix. Néanmoins, une fois une bonne organisation trouvée (cf. paragraphe suivant), manipuler Git nous a permis de gagner beaucoup de temps et de suivre l'avancement de notre projet au fil des séances avec facilité.

Le reste du projet s'est bien déroulé. *is_legal*_bis était la fonction la plus difficile à implémenter en raison des nombreux cas à traiter. Pour l'affichage du jeu, nous avons pensé à utiliser SDL, mais nous avons du mal à l'installer et l'utilisation de ce logiciel ne nous semblait pas primordial, tant que le déroulement du jeu se faisait correctement.

Outre la difficulté à traiter de manière exhaustive tous les cas de déplacements dans *is_legal_bis* et les problèmes d'installation de SDL sur notre machine locale, c'est la gestion des boucles do-while qui nous a pris un certain temps. En effet nous n'étions pas du tout familier à ce type de boucle et il nous a donc fallu apprendre à bien les utiliser.

Enfin dans ces mêmes boucles nous nous sommes aperçus que, lors de la première itération, la commande *scanf* laisse des caractères dans le buffer d'entrée, notamment le caractère de nouvelle ligne "\n". Ce phénomène, difficilement prévisible nous a beaucoup posé problème et nous avons finalement trouvé compe solution de rajouter un espace avant de lire les entrées : " %c%d" au lieu de "%c%d"

2.5 Organisation du travail et gestion de git

Pour bien s'organiser, nous avons décidé de travailler sur les fonctions du code séparément, en vérifiant, à deux, à chaque fois que l'une d'entre-elles étaient terminées, que tout était juste. Cela nous a permis de gagner du temps en évitant les conflits les plus pénibles et c'est précisément ici que les commandes de Git pour mettre à jour notre dossier ont été très avantageuses.

2.6 Conclusion

Ce projet était intéressant pour nous apprendre à la fois à travailler sur un projet de A à Z et à collaborer avec un partenaire. La complexité de l'exercice était plutôt équilibrée. Nous sommes satisfaits d'avoir atteint un point final dans notre travail, de pouvoir jouer au Bobail dans le terminal de notre ordinateur, même si l'un contre l'autre nous serions sans aucun doute incapables de terminer une partie entièrement...