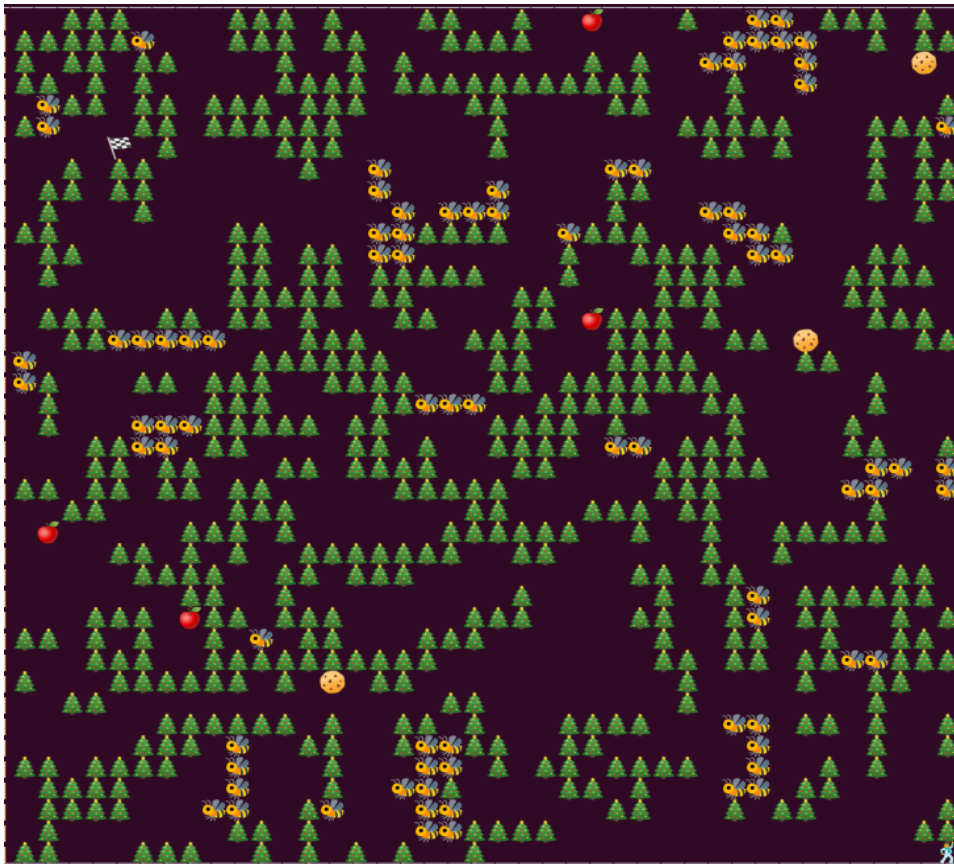


Rapport de Projet

Jeu 2D en C



BOUHRARA Adam

CASSIET Clément

CERF Fabien

GAUTIER Jordan

LEGRAND Joan

Sommaire

I.	Introduction	2
II.	Démarche	2
III.	Réponses Attendues au Cahier des charges	3
IV.	Structure du code	4
V.	Exemple	7
VI.	Conclusion	12

I. Introduction

Ce projet a pour but de réaliser un jeu en 2D dans lequel un joueur contrôle un personnage qui se déplace sur une carte avec des obstacles et des bonus à ramasser. Le personnage doit rejoindre son point d'arrivée avec l'énergie dont il dispose. Le programme est entièrement écrit en C, avec un makefile qui permet de le compiler et de l'exécuter facilement.

Afin de faire ce jeu, nous avons constitué un groupe de 5 personnes qui sont BOUHRARA Adam, CASSIET Clément, CERF Fabien, GAUTIER Jordan et LEGRAND Joan.

II. Démarche

Nous avons commencé par réfléchir sur une feuille de papier à la structure générale du code, aux différents fichiers et dossiers à créer, ainsi qu'aux fonctions dont nous aurions besoin pour développer le jeu.

Nous nous sommes fait une idée générale de ce qui nous attendait, ce qui nous a permis de nous répartir les fonctions à réaliser pour pouvoir travailler dessus de manière indépendante, tout en faisant régulièrement des points pour connaître l'avancée de chacun et le travail restant.

Nous avons décidé d'utiliser Git (https://github.com/clementcst/projet_cookie_ing1) afin de pouvoir mettre plus facilement en commun notre travail et pour avoir les dernières modifications apportées sur notre travail. De cette façon, et au fur et à mesure du développement de notre jeu, on a pu garder un historique des différentes versions de notre programme, au cas où un changement, récent ou non, ne correspond plus à notre vision du jeu. Toute cette organisation dans le partage des données, l'utilisation des fichiers et l'implémentation de nouveaux morceaux de code a été bien pensée dès le début de la réalisation du projet. Ce qui nous a permis de rester serein dans notre méthode de développement, et ainsi réagir de manière rigoureuse et méthodique face aux différents problèmes rencontrés.

Lors de la mise en commun de nos fonctions, il a fallu effectuer beaucoup de débogage pour que l'apport de l'un ne gêne pas celui d'un autre. On a ainsi dû penser à d'autres solutions lorsqu'on a changé plusieurs fonctions qui ne fonctionnaient pas bien ou qui n'étaient pas optimisées. Toutes ces modifications effectuées nous ont permis au fur et à mesure de l'avancée du projet de rendre notre programme toujours plus adaptatif, et évolutif. Des éléments qui sont essentiels lorsque nous voulons donner une plus grande dimension à notre code, et d'y apporter des modifications par le futur.

Tout au long de la réalisation du projet, nous avons veillé à toujours rendre notre code lisible et compréhensible pour tout le monde, par le biais de commentaires explicites et des noms de fonctions/procédures claires pour comprendre directement leur but.

III. Réponses Attendues au Cahier des charges

Ce programme permet d'exécuter un jeu dans lequel on contrôle un personnage qui se déplace sur une carte en 2D générée aléatoirement, dont la faisabilité est vérifiée, et qui cherche à rejoindre un point d'arrivée (Drapeau).

- Sur son chemin, le personnage pourra rencontrer des obstacles qui lui font perdre de l'énergie (Arbres et Abeilles) et des bonus qui lui en font gagner (Pommes et Cookies).
- Si le joueur n'a plus d'énergie, la partie est perdue.
- Le personnage peut se déplacer de haut en bas, avec les touches z/x, de gauche à droite, avec q/d, mais aussi en diagonale avec a/e/w/c.



- Le joueur a également la possibilité de faire jusqu'à 6 retours en arrière pour annuler le dernier mouvement.

➤ Il est possible de reprendre une partie en cours après avoir quitté le jeu, si cette dernière a été sauvegardée.

➤ Il est également possible de revoir une ancienne partie.

À la fin d'une partie il est :

➤ Afficher la distance parcourue, l'énergie restante, l'énergie gagnée et l'énergie perdue.

➤ Afficher le chemin emprunté par le personnage.

➤ Afficher le meilleur chemin à prendre en terme de distance.

➤ Afficher le meilleur chemin à prendre en terme d'énergie.

IV. Structure du code

Structure générale

Le jeu est réparti en trois dossiers et un makefile. Il y a un dossier src qui contient tous les fichiers .c avec les fonctions du jeu, un fichier include qui contient tous les fichiers .h qui lient les fichiers entre eux ainsi qu'un include.h qui contient toutes les constantes utilisées au cours de la partie, comme la taille de la carte, l'énergie du personnage ou encore les icônes utilisées. Enfin il y a un dossier data qui permet de stocker les parties en cours ou terminées. Après compilation à l'aide du makefile et de la commande make, un dossier obj contenant tous les .o et un dossier bin contenant l'exécutable du jeu sont créés.

Main.c

Fonction principale, c'est le point d'entrée du programme et il définit le déroulement du programme.

Display.c

Fonctions d'affichage de la carte, des menus et des informations de fin de partie.

Game.c

Fonctions principales de jeu: fonctions de gestion de fin de partie, de début de partie, de gestion de l'historique et de gestion de la sauvegarde par l'utilisateur.

InitGame.c

Fonctions qui permettent d'initialiser la partie en créant la carte, en disposant des obstacles et des bonus. Permet aussi de placer le joueur, le personnage et le point d'arrivée. Contient aussi les fonctions qui vérifient la faisabilité de la carte

ManageFiles.c

Fonctions de gestion des fichiers de sauvegarde (Save.csv), d'historique (History.csv) et de partie en cours (Current_Game.csv).

Miscellaneous.c

Contient tout ce qui n'est classable dans les autres fichiers: les fonctions et initialisations relatives au maniement de différentes structures, fonction des listes par exemple.

On a par exemple, les allocations et libérations de mémoire, création de node etc.

Movements.c

Toutes les fonctions qui permettent de faire les déplacements du joueur sur la carte, mettre la carte à jour et aussi d'actualiser les informations du personnage comme son énergie ou la distance parcourue.

Dijkstra.c

Tout le code relatif à l'implémentation de l'algorithme de Dijkstra dans le jeu, pour retrouver le meilleur chemin en énergie et en distance.

Current Game.csv

Fichier dans lequel on écrit les informations concernant la partie en cours de jeu.

Coordonnées x, coordonnées y, l'énergie du personnage, sa distance parcourue, l'énergie

gagnée en prenant un bonus, l'énergie perdue en rencontrant un obstacle, les informations de la carte et le chemin parcouru par le personnage. Ce fichier est supprimé à la fin de la partie.

Save.csv

Fichier dans lequel on fait la sauvegarde d'une game si le joueur le choisit. Il est une copie du Current_Game.csv avec en plus le

History.csv

Fichier dans lequel liste toutes les parties qui ont été terminées et peuvent être regardables. Il est à une structure identique à Save.csv mais avec plusieurs parties sauvegardées.

Structures de données principales :

La structure GameInformation qui contient toutes les informations du joueur et de la carte concernant la partie en cours. Cette structure n'est utilisée que dans main.c et game.c et regroupe la matrice de la map, la matrice des distances, les listes des chemins pris ou à prendre selon Dijkstra et la structure playerInfo.

Ensuite dans les variables importantes il y a matrice_Map et matrice_Distance. La première stocke les données de la map qui sont des int mais sous forme de char avec le code ASCII des données. La variable matrice_Distance est un tableau à trois dimensions qui permet d'avoir la distance entre chaque case sur la carte.

V. Exemple

Démarrage :

Afin d'exécuter le programme, il faut aller sur un invite de commandes, puis faire le chemin jusqu'au dossier contenant tous les fichiers fournis.

Ensuite, il faut compiler le programme en utilisant la commande : `make`

Après cela pour lancer une partie, il faut utiliser la commande : `make run`

Il existe également d'autres commandes pour :

-vider les dossier `obj` et `bin` : `make clean`

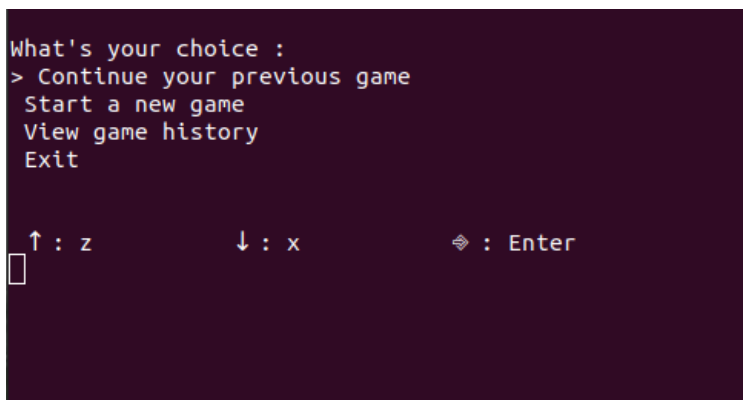
-supprimer les dossier `obj` et `bin` : `make hardClean`

-afficher les parties sauvegardées : `make save`

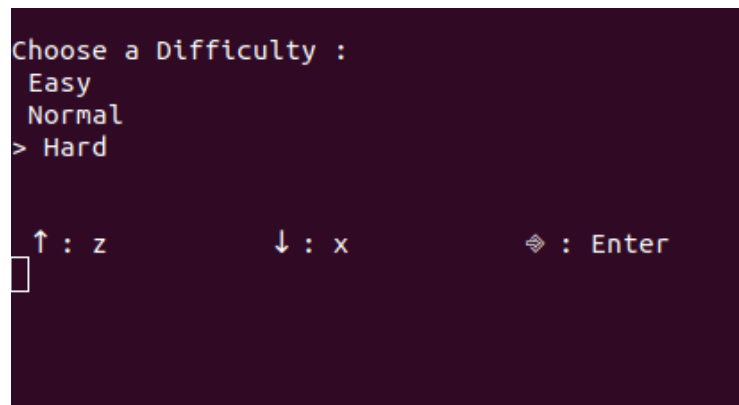
-vider le fichier des sauvegardes : `make cleanSaves`

-supprimer les dossier `obj` et `bin`, compiler le programme et lancer une partie : `make super`

Il existe également une option `make help` : permet d'afficher les instructions permettant l'exécution du programme.

A screenshot of a terminal window with a dark purple background. The text is white. At the top, it says "What's your choice :". Below that, there is a list of options: "> Continue your previous game", "Start a new game", "View game history", and "Exit". The first option is preceded by a greater-than sign. At the bottom of the menu, there are three lines of text: "↑ : z", "↓ : x", and "↵ : Enter". Below these, there is a small white square cursor.

Voici le menu interactif d'entrée du jeu, proposant de choisir entre reprendre la partie sauvegardée, démarrer une nouvelle partie, voir les parties en historique ou fermer le jeu.



Si on choisit de créer une nouvelle partie, le jeu nous propose de choisir la difficulté (Facile, Normal ou Difficile) ainsi que la taille (Petit, Moyen ou Grand).



Une fois la difficulté sélectionnée, la partie se lance. Dans l'exemple ci-dessus, la taille est grande et la difficulté : difficile. On peut voir que chaque déplacement possible est associé à

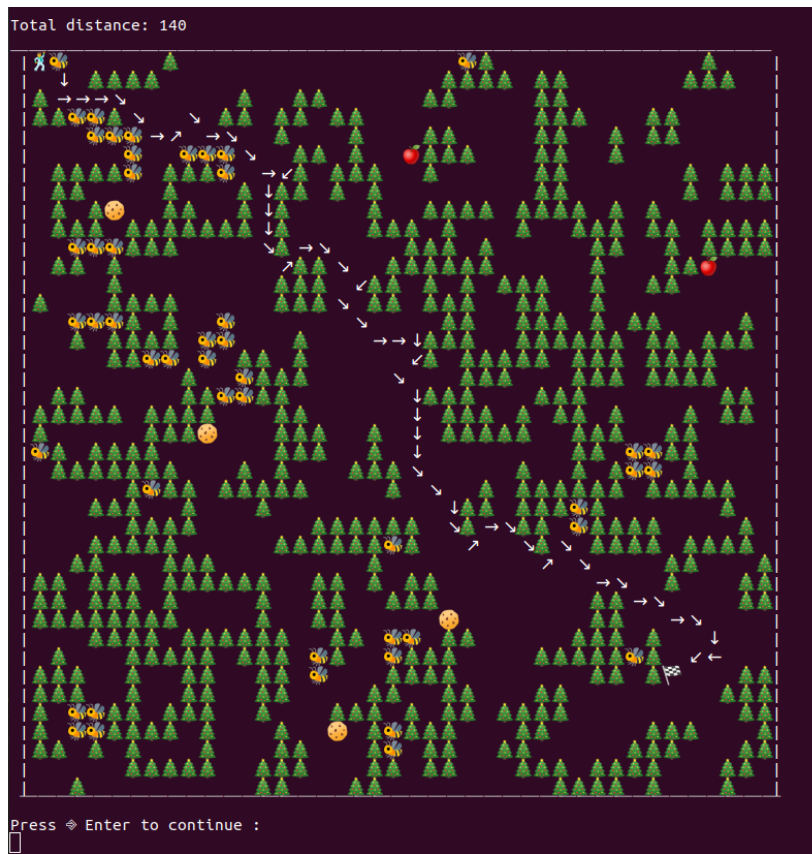
une distance : droite = 7km, bas = 9km et diagonale (bas/droite) = 1km. Ces distances sont générées aléatoirement au début de la partie et permettent de jouer en cherchant le chemin le plus court en distance. On peut également chercher le chemin le plus court en énergie.



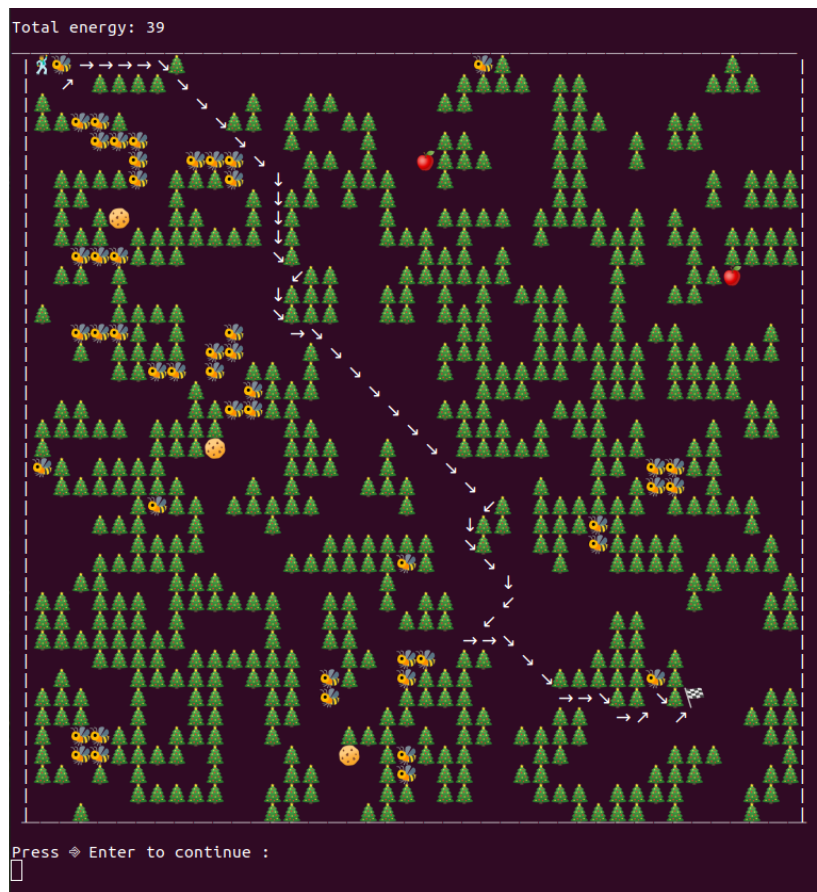
Lorsque le joueur se déplace, les cases déjà parcourues sont marquées par des points, afin que le joueur sache s'il est déjà passé par une case. Si le joueur marche de nouveau sur une de ces cases, un message d'alerte s'affiche pour le prévenir. On peut voir au bas de l'écran qu'un message s'affiche aussi lorsque le joueur heurte un obstacle.

```
Bravo, vous avez gagné avec 33 d'énergie restante !  
Vous avez parcouru 245.  
Vous avez gagné 80 d'énergie grâce au bonus et perdu 44 d'énergie à cause des obstacles.  
Press ⬅ Enter to continue :  
□
```

Une fois la partie gagnée, ce message de victoire s'affiche, indiquant l'énergie restante, la distance parcourue, l'énergie gagnée grâce aux bonus et l'énergie perdue à cause des obstacles.



Après avoir confirmé, le chemin de moindre coût (en distance) s'affiche (calculé avec l'algorithme de Dijkstra).



Ensuite, le chemin de moindre coût (en énergie) est affiché. S'ensuit un visionnage de la partie terminée.

VI. Conclusion

Ce projet de réalisation d'un jeu en C a été une expérience très enrichissante pour l'équipe en termes de compétences techniques mais aussi sur des aspects de gestion de projet. Composée de cinq personnes, nous avons dû faire preuve de collaboration mais aussi d'écoute pour mener à bien ce projet.

C'est tout d'abord l'aspect visuel et ludique dans la création d'un jeu qui a été une source de motivation pour l'équipe, qui a travaillé avec enthousiasme pour améliorer la jouabilité et les fonctionnalités du jeu. Ce projet a été l'occasion pour chaque membre de l'équipe de développer ses compétences en programmation C sur plusieurs aspects. En effet, nous avons dû mettre en pratique nos connaissances et aller au-delà, spécialement sur la gestion de pointeurs, de structures, de listes, la manipulation de fichiers et les affichages graphiques dans une console.

L'équipe a également dû faire face à des évolutions du code au cours du développement, et a su gérer ces changements de manière efficace grâce à l'utilisation de différents outils de partage de données, et une communication en continue tout au long du projet.

La seule pointe de regret que nous pouvons avoir serait de ne pas avoir parfaitement géré le timing de la fin de projet. En effet nous n'avons pas totalement eu le temps de finir les dernières modifications et les améliorations que nous aurions voulu ajouter au jeu. Cela l'aurait rendu encore plus complet, attrayant et esthétique notamment avec une meilleure visualisation de l'historique, indiquant si la partie avait été gagnée ou perdue avant de la visionner et un historique affichant le chemin parcouru derrière le personnage.

En résumé, la réalisation de ce projet sur la réalisation d'un jeu en C a été une expérience très positive pour l'ensemble de l'équipe. Chacun a pu améliorer ses compétences en programmation et en communication, tout en travaillant de manière collaborative sur le projet.