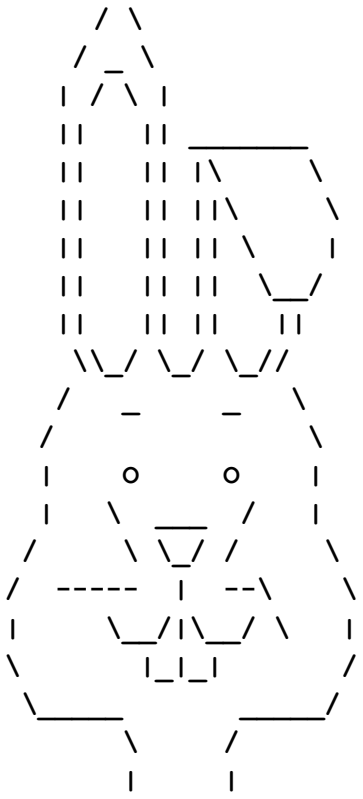
LEPELTIER Florentin
BENARD Theo

Mini-projet langage C

CROQUE-CAROTTE

B1-S2



SOMMAIRE

INTRODUCTION

Afin de nous initier et de nous familiariser au langage C, nous avions pour objectif de réaliser le jeu croque-carotte en binôme. Rentrons directement dans le vif du sujet.

CAHIER DES CHARGES

Le cahier des charges était le suivant :
-Le plateau de jeu devait comporter 25 cases
-Le nombre de lapins par joueur était au nombre de 4
-3 cases fixes devaient pouvoir s'ouvrir aléatoirement au cours d'une partie
-Une pioche de 24 cartes devait être mise en place, mélange des cartes compris
-La vérification des erreurs de saisie devait être effectué

ORGANISATION DU PROJET

L'avancement du projet a été entièrement géré par les fichiers "Hiérarchie du projet" et "Agenda", ces deux fichiers vous sont joints dans le fichier zip.
La hiérarchie du projet renseigne sur l'évolution du projet, il constitue la première approche que nous y avons fait, ensuite, des commentaires y ont été ajoutés au fur et à mesure, comme une sorte de "brouillon suivi" du projet.
L'agenda quant à lui, renseigne sur l'architecture du projet en elle même (fonctions/fichiers/etc.) et contient également la répartition des tâches au sein du binôme. Dans l'ensemble, si on jette un oeil à l'agenda, le travail a été complètement partagé au sein du groupe. Un membre du binôme créait une fonction que l'autre membre venait modifier/améliorer ou compléter, de cette manière aucune partie du projet n'est un "mystère" pour l'un ou l'autre et les différentes fonctionnalités ont été assimilées par les deux membres du groupe, ce qui est important.
Nous tenons à préciser que l'ensemble a été développé sous Mac OS, néanmoins, nous vous transmettons tout de même un exécutable Windows compilé en remplaçant les system("clear") par des system("cls"). Ne figure aucun warning lors de la compilation sur Mac, par contre, sous Windows apparaissent des warnings liés à la fonction "usleep", que nous n'avons pas "résolu".

ARCHITECTURE DU PROJET

Dans cette partie nous aborderons les grandes lignes de notre programme sans tout détailler, cela n'aurait pas grand intérêt.

Le plateau est un tableau d'entiers où chaque indice peut prendre 4 valeurs en fonction de l'état d'une case (Vide/Occupée par un joueur(2)/Ouverte=trou)
On affiche un caractère sur le terminal en fonction de la valeur stockée dans le tableau d'entiers.

Ensuite, chaque joueur dispose d'une structure "générique" par lapin comportant 3 champs (couleur,position,etat). Un joueur disposant de 4 lapins, nous avons créer un tableau de structures comme illustré ci- dessous :

J1

J1[0] = lapin n°1	J1[1] = lapin n°2	J1[2] = lapin n°3	J1[3] = lapin n°4
Structure générique d'un lapin : -couleur -position -etat	-couleur -position -etat	-couleur -position -etat	-couleur -position -etat

Ainsi nous avons autant de tableaux de structures qu'il y a de joueurs.

La pioche est un tableau d'entiers auquel on vient affecter la valeur des différentes cartes disponibles, lorsque l'on pioche, on récupère cette valeur qui est réutilisée dans la fonction "avancer" qui permet d'actualiser la position des lapins dans leur structure ci-dessus.

On lie ensuite la valeur de la position d'un lapin avec le plateau. Ainsi une valeur est attribuée au plateau, un caractère est affiché en conséquence via une fonction "affichage_plateau".

On peut noter que dans le programme le champ couleur n'a finalement pas été utilisé, nous l'avons tout de même laissé, nous pensons qu'il est encore possible d'exploiter le champ de cette structure.

DIFFICULTÉS RENCONTRÉES

La principale difficulté a été l'utilisation des structures, n'ayant jamais manipulé ce type de données, il nous a été nécessaire de nous renseigner sur le sujet et de comprendre leur fonctionnement. Tout comme les pointeurs qui étaient une nouveauté, mais qui désormais s'avèrent bien pratiques.
Les modulos sont également un peu embrouillants avec les tableaux, et les boucles, il est nécessaire de bien se poser pour ne pas commettre d'erreurs avec les décalages d'indices à droite et à gauche, pouvant conduire à des bugs parfois longs à repérer…
Et pour finir, nous n'avons pas réussi à écrire un code entièrement "configurable" , nos idées étant déjà floues au départ, et ne maîtrisant pas vraiment le langage, il était compliqué de prendre du recul et d'adapter le code en fonction. Il est donc nécessaire de repenser en grande partie ce dernier afin d'obtenir un jeu entièrement configurable.

NIVEAU ATTEINT

Cahier des charges de base	oui
Cahier des charges (extensions)	non (recherches sans réel aboutissement)
Cahier des charges pour les plus forts	un peu (quelques suppléments)

Nous avons réalisé le cahier des charges de base et avons ajouté quelques fonctionnalités :

-Les cases qui s'ouvrent changent aléatoirement d'une partie à l'autre
-Nous demandons aux deux joueurs de saisir leurs noms au début de la partie
-Ajout des règles du jeu dans le menu
-Nous avons essayé de rendre l'affichage un peu plus clair et agréable
-Des animations ont été ajoutées lors du déplacement des lapins

Nous avons également tenté de manipuler la SDL, nous sommes parvenus à ouvrir une fenêtre mais avons préféré nous concentrer sur la correction de bugs, ces derniers étant assez nombreux…
Ne voyant pas réellement comment réaliser un jeu jouable entre 2 postes différents et par manque de temps, nous avons préféré laisser cela un peu de côté. Néanmoins pas mal de recherches intéressantes ont étés menées sur la partie réseau et communication.
Nous avons inclus ces recherches/tests dans notre fichier zip auquel cas vous voudriez y jeter un oeil.

BUGS PERSISTANTS

Quelques bugs persistent encore, voici ceux que nous avons relevés :

-Si un joueur n'a plus qu'un lapin en vie et qu'il pioche une carte le faisant avancer plus loin que la case carotte il ne pourra pas sélectionner son lapin, aucune solution n'est donc possible pour terminer la partie…
Ce bug aurait sûrement pu être corrigé mais les règles à ce sujet n'étant pas exprimées nous ne l'avons pas résolu, de plus, cela aurait compliqué encore plus la fonction "selection_lapin".
-La pioche fonctionne presque, en effet 25 cartes sont piochées au lieu de 24 et il survient souvent que des cartes carotte soient piochées à la place d'autres…
-Il est possible que d'autres bugs soient présents, si c'était le cas, nous ne les avons alors pas rencontrés ou avons oublié de les noter...

AMÉLIORATIONS POSSIBLES

Il est toujours possible de faire mieux, dans notre cas, nous aurions pu créer un code plus configurable (en ajoutant un tableau de joueurs par exemple). Une lecture plus approfondie des règles du jeu aurait pu nous éviter de perdre du temps "inutilement", tout comme la lecture des annexes que nous n'avons pas lues au départ, et avons donc oublié de lire jusqu'au dernier week end avant la soutenance…
La connexion en réseau, afin de jouer soit sur deux terminaux différents d'un même écran sur un réseau local, ou en réseau sur deux écrans différents à l'aide de l'adresse IP internet ...
La partie graphique avec SDL
La pertinence des commentaires pourrait également être revue comme la factorisation du code.

AVIS PERSONNELS

Dans l'ensemble nous sommes plutôt contents du travail réalisé, le fait de travailler sur un projet est vraiment entraînant et motivant, au début, on ne voit pas trop comment procéder, puis, cela finit par venir jusqu'à parfois même vous rendre accro. Cette expérience nous a permis de prendre du recul et de maîtriser les bases du langage C, certes, il reste beaucoup à apprendre mais ce genre d'approche est plaisante et vraiment enrichissante.
Un grand merci d'avance pour le temps que vous consacrerez à étudier notre projet, et, pour la qualité de vos enseignements formateurs auxquels nous nous devons d'être reconnaissants !