RAPPORT DU PROJET D'OCAML

LE MASTERMIND

Bovie Pierre-Edouard

Zabsonré Ahmonkou

Table des matières

I. Introduction

Le Mastermind est à la base un jeu de société dont le but est de trouver une combinaison. Généralement c'est à nous de trouver cette combinaison en jouant contre un ordinateur, mais le but de ce projet consiste à inverser ces rôles. Pour ce projet, il faut donc faire des fonctions permettant à l'ordinateur de trouver la combinaison secrète que l'utilisateur garde en tête en ayant choisi cinq couleurs parmi les huit disponibles, qui sont le rouge, jaune, blanc, bleu, violet, vert, orange et fuchsia.

II. Fonctionnement du programme

Au début du projet, nous avions décidé de créer le type couleur et le type pions. Chaque pion a une couleur et un chiffre. Dans une première approche, le chiffre des pions nous a paru être une bonne idée car on aurait pu choisir une combinaison de chiffre (donc de pions) aléatoirement grâce a la fonction rand et compris entre 1 et 8, le tout dans un tableau.

Mais rapidement nous laissâmes cette idée de côté car nous partions sur trop compliqué. Nous décidâmes donc d'utiliser la récursivité sur les listes pour faire le projet.

Le principe a été de créer une liste de listes de string, c'est à dire une liste ayant les 8 couleurs de base. Et à partir de cette liste, on aura les listes de toutes les combinaisons possibles de cinq couleurs parmi les huit en ajoutant à chaque tour une couleur différente.

Cette grande liste est la liste de toutes les combinaisons possibles que l'ordinateur utilisera pour déterminer la combinaison du joueur.

Ensuite à chaque tour et tant que la combinaison n'est pas la bonne, l'ordinateur proposera une combinaison et le joueur devra suivre les indications données pour indiquer les couleurs qui sont bien placées et celles qui ne le sont pas. Grâce à ça l'ordinateur pourra supprimer les combinaisons qui sont incorrectes dans la liste de combinaisons initiale, et ainsi de suite jusqu'à ce qu'il ne reste qu'une seule combinaison dans la liste.

La répartition du travail dans notre groupe a été très "brouillon", chacun écrivant une partie du code que nous envoyons ensuite à l'autre pour l'améliorer et la tester. Le travail s'est très vite accéléré quand les fonctions ont commencé à bien fonctionner.

III. Fonctions créées et utilisées

1. let rec construire taille ;;

Cette fonction a pour but de créer une liste contenant d'autres listes. Ces dernières forment, chacune, une combinaison de couleurs dont la taille est « taille ».

val construire : int -> string list list = <fun>

2. let rec afficher_combinaison combinaison ;;

Son but est de permettre à l'ordinateur d'afficher une combinaison à l'écran.

val afficher_combinaison : string list -> unit = <fun>

3. Let rec verification liste_en_memoire liste_reponse bp mp ;;

Cette fonction vérifie en fonction du nombre de couleur mal ou bien placée, si la combinaison donnée par l'ordinateur peut correspondre à celle de l'utilisateur. Si elle peut alors elle renvoie true, sinon false.

val verifcombinaison : 'a list -> 'a list -> 'a list -> int -> int -> bool = <fun>

4. Let rec suppression_combinaison propositions liste_combinaison bp mp ;;

Le rôle de cette fonction est de supprimer les combinaisons qui ne correspondent pas à celle de l'utilisateur, en fonction de la valeur du booléen retournée par la fonction let rec verification.

'a list -> 'a list list -> int -> int -> 'a list list = <fun>

5. Let rec jouer liste_combinaisons nombre_essai ;;

Cette fonction est en quelque sorte le tronc, la base du jeux. En effet, c'est à ce niveau que toutes les fonctions principales écrites sont utilisés, après qu'on ait choisi la version avec laquelle on veut jouer.

6. Let rec verification_redondances_combinaison ;;

Cette fonction vérifie si la combinaison donnée par l'ordinateur contient au moins deux fois la même couleur. Si oui alors elle renvoie true, sinon false.

7. Let rec suppression_redondances_combinaison ;;

Le rôle de cette fonction est de supprimer les combinaisons qui contiennent des couleurs identiques, en fonction de la valeur du booléen retournée par la fonction Let rec suppression_redondances_combinaison.

IV. Lancer le jeux

Taper dans le terminal ocamlc MASTERMIND.ml –o texte.exe .

Puis la touche entrée .

Ensuite taper texte.exe .

Et enfin la touche entrée .

V. Réponse à la question subsidiaire

Oui on peut se rendre compte que l'utilisateur triche. Plusieurs cas sont à signaler:

- L'addition du nombre de couleurs bien placées et mal placées ne peut être supérieure à 5;

- Si le nombre de couleurs bien placées est égal à 4 alors le nombre de mal placées ne peut être 1;