Mini-Projet OCaml

Simon Mauras

29 octobre 2014

Table des matières

1 Introduction **1**

2 Interfacer avec le terminal **1**

3 Avant de jouer... **2**

4 Blackjack **2**

5 Bataille **3**

6 Trichons... **3**

1 Introduction

Ce mini-projet est l'aboutissement de plusieurs TP de programmation sur le langage fonctionnel OCaml. Il a lieu dans le cadre du Projet de premier semestre de L3 Informatique à l'ENS de Lyon. L'objectif est de proposer une sorte de Game Center permettant de jouer au Blackjack et à la Bataille. Ce rapport expose les différents choix qui ont pu être faits lors de l'implémentation.

2 Interfacer avec le terminal

Le point d'entrée dans notre programme est le `let _ = ... ;;`. En effet toutes les commandes sont exécutées une seule fois et aucune variable intermédaire (comme `let p = read_line () in`) n'est déclarée de manière globale ce qui est plus propre. Le programme commence par demander quelques informations au joueur :

```
Quel est ton prénom ? Simon
```

Par la suite la fonction récurcive `menu_principal` affiche les choix possibles (même si ici ce n'est pas utile, elle est tail-récurcive pour qu'un accro du blackjack ne puisse pas faire déborder la pile...). La fonction auxiliaire `aux` boucle en attendant une réponse valide de l'utilisateur.

```
*****************************************
*  Credits 000       Batailles gagnees 000 *
*  Menu principal :                       *
*  1. Jouer au blackjack                  *
*  2. Jouer à la bataille                 *
*  3. Quitter                             *
*****************************************
Que voulez vous faire ? Rien
Choix invalide
Que voulez vous faire ? Bataille
Vous n'avez plus de jetons
```

```
Que voulez vous faire ? Quitter
Au revoir Simon
```

Le joueur peut rentrer soit un chiffre (ex : 1) soit un mot clé (ex : blackjack, BlackJack, ...) insensible à la casse. Différentes fonctions sont appelées selon le jeu choisi. Lorsque le joueur quitte, le programme le salue par son nom.

3 Avant de jouer...

Dans cette section, il nous était demandé de coder plusieurs fonctions basiques afin de faciliter l'implémentation des jeux de Blackjack et de Bataille. Une carte étant représentée par un entier et une couleur. Nous avons fait le choix d'avoir un As faible au BlackJack (vaut 1 point) et d'un As fort à la bataille (meilleur que le Roi).

Cela est en pratique implémenté avec deux sortes d'As, le 1 (As faible) et le 14 (As fort). La fonction `genere_jeu` génère un jeu de l'As faible au Roi pour le Blackjack et la fonction `genere_mini_jeu` génère un jeu dont la valeur des cartes est entre les deux entiers passés en paramètre (possibilité d'avoir un As fort à la Bataille).

Pour la fonction de mélange des cartes, il est intéressant de dire que si on suppose que la fonction `rand` de la bibliothèque possède une répartition uniforme, alors toutes les permutations sont équiprobables. Nous avons utilisé une implémentation naïve du mélange de Fisher-Yates [1].

La fonction `distribue` est identique à la fonction permettant de partitionner une liste en deux dans le tri fusion. La fonction `empiler_cartes` consiste en concaténer deux liste en un temps linéaire en la taille de la seconde.

4 Blackjack

La fonction principale du Blackjack est `jouer_blackjack`. Elle fait séquentiellement appel aux fonctions `faire_jouer_joueur` puis `faire_jouer_banque`. À la fin d'une partie un jeton est attribué au joueur en cas de victoire (champ mutable dans le paramètre de type `joueur`).

La fonction `faire_jouer_joueur` est récurcive, affiche l'état actuel du jeu et termine dès que la défaite est acquise (Un mauvais perdant pourrait avoir l'envie de vider la pioche et faire planter le programme, petite victoire en soit). La fonction auxiliaire `aux` s'assure que le choix fait est valide.

```
*****************************************
*                 Blackjack             *
*****************************************
***** Banque ******
2 de Carreau
Total = 2
***** Joueur ******
As de Pique
Dame de Trefle
Total = 11
******************
Voulez vous piocher (oui/non) ? Je sais pas
Choix invalide
Voulez vous piocher (oui/non) ? oui
***** Banque ******
2 de Carreau
Total = 2
***** Joueur ******
As de Pique
Dame de Trefle
Roi de Pique
```

1. http://en.wikipedia.org/wiki/Fisher-Yates_shuffle

```
Total = 21
******************
Voulez vous piocher (oui/non) ? non
***** Banque ******
2 de Carreau
2 de Coeur
3 de Trefle
5 de Carreau
2 de Pique
Valet de Trefle
Total = 24
***** Joueur ******
As de Pique
Dame de Trefle
Roi de Pique
Total = 21
******************
Vous avez gagne
Vous avez désormais 1 credit(s).
```

Nous avons amélioré "l'intelligence artificielle" de la banque. Elle ne s'arrête que quand elle a perdu (strictement plus de 21 points) ou quand elle a gagné (au moins autant de points que le joueur). En effet il n'est pas dans son intérêt de s'arrêter à 18 points si le joueur gagne, car continuer ne peut qu'améliorer les chôses.

5 Bataille

Pour la bataille nous avons choisis la règle suivante : si a un moment de la partie un joueur n'a plus de cartes pour continuer (même si une bataille est en jeu ou doit être complétée) celui-ci perd. Si les deux joueurs perdent en même temps, l'avantage est donné à "l'humain".

Après une première implémentation nous nous sommes retrouvés face au problème des parties qui bouclent et ne se terminent pas. Pour le résoudre nous nous sommes inspiré de la bataille entre deux humains au cours de laquelle les cartes gagnées sont mises de manière aléatoire sous le paquet. Nous avons donc décidé d'utiliser la fonction `melanger` en amont de l'utilisation de la fonction `empiler_cartes`.

Afin de pouvoir jouer avec des jeux de différentes tailles, le choix est laissé au joueur pour la valeur de la carte la plus faible. L'affichage consiste en un petit diagramme représentant le nombre de cartes restantes à chacun des deux joueurs. Une partie de Bataille se déroule donc de la manière suivante :

```
*****************************************
*                 Bataille                *
*****************************************
Jeu de carte de X à As, X = ? 10
Simon |||||  ||||| Adversaire    10 de Coeur   VS   Dame de Pique
Simon ||||  |||||| Adversaire    Dame de Coeur   VS   10 de Pique
Simon |||||  ||||| Adversaire    As de Pique   VS   As de Coeur  BATAILLE !
Simon ||| ???? ||| Adversaire    Valet de Pique   VS   Valet de Coeur  BATAILLE !
Simon | ???????? | Adversaire    10 de Pique   VS   Dame de Pique
Simon   |||||||||| Adversaire
Vous avez perdu...
Vous avez désormais 4 credit(s).
```

6 Trichons...

Depuis le menu principal, d'autres choix (non affichés) sont possibles. Le choix 4 permet à l'utilisateur de fixer la graîne utilisée par la fonction `rand`. Le choix 5 permet de gagner un jeton sans avoir à jouer au Blackjack.

Quelques graînes utiles pour tester notre programme :

1	Égalité au Blackjack
11113	Mégabataille pour X = 8 (14 cartes)
1190927	Mégabataille pour X = 7 (16 cartes)
5936020	Mégabataille pour X = 6 (18 cartes)