DM - Réalisation du jeu Asteroids

Ce DM (Devoir à la Maison) est un travail en binôme destiné à évaluer vos connaissances en programmation fonctionnelle avec OCaml. Il sera noté et comptera comme note de contrôle continu pour PRG2. Il doit être rendu à votre enseignant par email pour le 24 mars. Au cours des séances de TP 5 ou 6, vous passerez un bref oral individuel, où vous devrez répondre à des questions sur votre solution. Le travail personnel et la qualité de votre solution (correction et présentation) sont aussi importants que son degré d'avancement.

1 Problème

Il s'agit de programmer le célèbre jeu Atari *Asteroids* en utilisant la librairie graphique[1] de OCaml. Une description et l'histoire du jeu sont disponibles sur Wikipédia[2].

La section "Cahier des charges" spécifie les contraintes que votre solution doit absolument vérifier. La section "Améliorations possibles" donne des pistes pour améliorer votre jeu. Celles-ci sont optionnelles, mais peuvent vous valoir des points supplémentaires.

2 Cahier des charges

Votre programme doit permettre de jouer au jeu Asteroids à un joueur selon les règles de base. L'espace de jeu doit afficher sur fond noir d'espace : le vaisseau du joueur, les astéroïdes et les éventuels projectiles. Le vaisseau est initialement au centre de l'espace de jeu, est vu de dessus et pointe dans une direction. Le joueur peut faire tourner le vaisseau sur lui-même pour changer sa direction. Le déplacement du vaisseau fait partie des améliorations possibles. Les astéroïdes sont de taille, de couleur, de position initiale, de direction et de vitesse variables. Vous utiliserez la fonction `Random.int` pour générer aléatoirement ces différents paramètres. Au début du jeu, un certain nombre d'astéroïdes sont créés puis ils se déplacent selon un mouvement uniforme (direction et vitesse constantes). Quand un astéroïde sort du cadre de jeu, il réapparaît de l'autre côté. Les projectiles sont émis par l'action du joueur (touche espace). Ils ont tous la même taille (petite) et la même vitesse (grande). Ils partent du vaisseau et se déplacent dans la direction qu'avait le vaisseau au moment du tir. Quand ils sortent du cadre, ils disparaissent. Vous devez gérer au moins deux types de collisions : (1) les collisions projectile-astéroïde et (2) les collisions astéroïde-vaisseau. Les premières doivent entraîner la fragmentation de l'astéroïde en deux/trois astéroïdes plus petits partant dans des directions et

1. http://caml.inria.fr/pub/docs/manual-ocaml-4.02/libref/Graphics.html
2. http://fr.wikipedia.org/wiki/Asteroids

vitesses aléatoires, ou bien la disparition de l'astéroïde s'il a une taille minimale. Les deuxièmes entraînent la destruction du vaisseau et donc la fin de la partie.

Un squelette de programme commenté `asteroids.ml` vous est fourni et contient entre autre une boucle d'animation et une boucle d'interaction. La boucle d'animation appelle toutes les vingtièmes de seconde la fonction `affiche_etat` pour afficher l'état courant du jeu et la fonction `etat_suivant` pour calculer l'état suivant selon les trajectoires du vaisseau, des astéroïdes et des projectiles. La boucle d'intéraction détecte les frappes clavier de l'utilisateur et appelle diverses fonctions pour modifier l'état du jeu en conséquence : `rotation_gauche` et `rotation_droite` pour faire pivoter le vaisseau, `tir` pour lancer un nouveau projectile, et `acceleration` pour accélérer le vaisseau (voir améliorations). L'appui de la touche 'q' termine le programme. Votre travail consiste à définir l'état de jeu pour qu'il contienne toutes les informations utiles, et à définir toutes les fonctions précédentes pour afficher et modifier ces états selon les règles du jeu. Il est recommandé d'utiliser des listes et les fonctions d'ordre supérieur telles que `List.map`, `List.fold_left` pour manipuler l'ensemble des astéroïdes ou projectiles.

Pour compiler le programme, exécuter dans un terminal la commande suivante.

```
ocamlc -o asteroids unix.cma graphics.cma asteroids.ml
```

Pour lancer l'exécutable produit, il suffit d'exécuter la commande `./asteroids`.

3 Améliorations possibles

S'il vous reste du temps, il y a de multiples façons d'améliorer le jeu.

- permettre au vaisseau de se déplacer selon les lois de la physique. La touche d'accélération permet d'augmenter la vitesse du vaisseau, qui est initialement à zéro. Comme le vaisseau n'a qu'un moteur arrière, il doit, pour freiner, pivoter à 180 degrés et accélérer. Lorsque le vaisseau sort du cadre, il apparaît de l'autre côté.
- détecter les collisions entre deux astéroïdes entraînant leur fragmentation. Pour éviter que les fragments ne se détruisent réciproquement, une astuce consiste à ignorer les collisions entre astéroïdes de même couleur.
- gérer un score. Chaque destruction d'un astéroïde apporte un nombre de points en fonction de sa taille.
- gérer un nombre de vies. Quand le vaisseau est détruit, une vie est perdue. Quand il n'y a plus de vie, le jeu est terminé.
- permettre la téléportation du vaisseau. À la demande du joueur, le vaisseau peut réapparaître à une position aléatoire de l'espace de jeu.
- gérer des niveaux de jeu. Quand tous les astéroïdes ont été détruits, on passe au niveau suivant (par ex. avec plus d'astéroïdes).

4 Livrables

Il y a 2 documents à rendre : un rapport et le fichier source. Le rapport devra contenir les éléments suivants :

- une introduction présentant votre compréhension du problème,
- votre analyse du problème, son découpage en sous-problèmes (fonctions intermédiaires),
- un bilan de ce que vous avez réalisé, si le cahier des charges est entièrement rempli, si vous avez réalisé des améliorations,
- la commande permettant d'exécuter votre programme,
- un rapport de test de votre programme, avec 2 ou 3 exemples illustratifs,
- une conclusion où vous indiquerez le temps que vous avez consacré à ce projet, les difficultés rencontrées, ce que vous avez appris.

Veillez à soigner la lisibilité du code (commentaires, nom des fonctions et des variables, indentation) autant que la présentation du rapport (orthographe, grammaire, style, mise en page).