TME numéro 1

(affichage de objets graphiques)

Exercice 1 (installation) Pour vous aider à réaliser votre projet, l'équipe enseignante a créé des fichiers LISP spécifiques pour xemacs, des fichiers OCaml, un toplevel contenant toutes sortes d'aides, etc. Pour en bénéficier, vous devez exécuter la commande suivante dans une console :

```
/Infos/lmd/2004/licence/ue/li260-2005fev/g7/bin/install
```

Une fois cette commande effectuée, vous pouvez lancer xemacs. Ce dernier utilisera un mode tuareg customisé pour votre projet. En particulier, vous aurez accès à une commande, programmée, je dois le dire, de main de maître par Guénaël Renault, et vous aidant à insérer vos commentaires OcamlDoc (commande « Insert header OcamlDoc » du menu tuareg, ou bien taper « M-o » pour les hackers — « M-o » = Méta-o = Alt-o), à utiliser sans modération. Remarquez en outre que vous avez désormais dans votre home directory un répertoire `li260-2005fev` qui est prévu pour que vous puissiez y déposer les programmes de cette UE.
Si vous voulez avoir le même environnement pour programmer chez vous (sous Linux bien entendu, personne n'aurait l'idée saugrenue de vouloir utiliser un autre système d'exploitation !), no problemo : il suffit de lancer dans votre home directory sur les machines de l'UFR la commande suivante :

```
tar cvjf ocamlpilot.tar.bz2 /Infos/lmd/2004/licence/ue/li260-2005fev/g7
```

puis de copier sur disquette ou clé USB le fichier « `ocamlpilot.tar.bz2` ». Une fois chez vous, connectez-vous en tant que `root`, ouvrez une console et tapez « `cd /` », recopiez la disquette ou la clé USB là où vous êtes et copiez le fichier « `ocamlpilot.tar.bz2` » dans le répertoire courant puis tapez :

```
tar xvjf ocamlpilot.tar.bz2
```

Maintenant, si vous tapez la commande « `ls` », vous devez avoir un répertoire `Infos`. Ce dernier contient tous les fichiers OcamlPilot. Supprimez le fichier « `ocamlpilot.tar.bz2` », déconnectez-vous et reconnectez-vous avec votre login habituel. Vous pouvez maintenant lancer le programme `install` décrit plus haut et vous avez le même environnement qu'à l'UFR. Top moumoute non ?

Exercice 2 Écrivez la ou les ligne(s) nécessaire(s) pour ouvrir une fenêtre graphique OCaml, puis la ou les ligne(s) pour la fermer. Réouvrez la et tracez une ligne, puis un cercle. Dessinez un autre cercle avec une couleur différente.

Exercice 3 Écrivez une fonction `affiche : int * int * Graphics.color -> unit` qui prend en argument un triplet dont les deux premiers éléments correspondent aux coordonnées (x, y) d'une balle, et qui affiche cette balle aux coordonnées indiquées avec la couleur spécifiée dans le troisième argument. Vous déclarerez au préalable une variable `rayon_balle`, que vous initialiserez à la valeur `ref 4`, et vous utiliserez cette variable dans votre fonction pour réaliser votre affichage. Le fait que ce soit une référence n'est pas très importante pour l'instant, mais vous verrez par la suite que ce sera utile. N'oubliez pas d'inclure les commentaires ocamldoc, i.e., pensez à la commande « M-o » ou bien à « Insert header OcamlDoc » du menu « tuareg », ça me ferait tellement plaisir (à moi et, je pense aussi, à Guénaël qui s'est donné du mal pour vous programmer cette commande xemacs).

Exercice 4 Modifiez la fonction précédente de manière à ce qu'elle prenne en argument un élément de type `eltJeu`, cf. le fichier `jeu.mli` se trouvant dans la rubrique « ressources » de la page ouèbe d'OcamlPilot :

`http://www-sysdef.lip6.fr/~gonzales/li260-2005fev`

Vous utiliserez un filtrage pour différencier les balles des autres objets (cube, vaisseau, plateau) : lorsque vous avez une balle, vous l'affichez, sinon vous ne faites rien.

Exercice 5 Dans le jeu que l'on veut obtenir, l'affichage est toujours centré sur votre vaisseau spatial. Rajoutez donc à la fonction précédente deux arguments (vous placerez ceux-ci avant l'élément de type `eltJeu` de manière à ne pas être embêtés pour faire votre filtrage) représentant les coordonnées de votre vaisseau spatial et modifiez l'affichage en conséquence. Testez abondamment.

Exercice 6 Modifiez la fonction précédente de manière à ce qu'elle ne prenne plus en troisième argument seulement une balle, mais une liste de balles (toujours au format `eltJeu`).

Exercice 7 Rajoutez dans votre filtrage l'affichage des cubes (toujours au format `eltJeu`).

Exercice 8 Rajoutez enfin dans votre filtrage l'affichage des vaisseaux ainsi que du plateau de jeu. Les tailles de ces objets (rayon des vaisseaux, et largeur et hauteur du plateau (arène)) seront stockées dans des variables dont les valeurs sont laissées à votre discrétion.

Exercice 9 Lancez ocamldoc sur votre fichier. Je rappelle comment lancer OcamlDoc : soit vous avez patché votre fichier `.bashrc` et vous devez juste exécuter dans une console la commande :

```
ocamldoc nom_fichier.ml
```

soit vous ne l'avez pas fait (well, shame on you) et vous devez lancer :

```
ocamldoc -d docs -g \
/Infos/lmd/2004/licence/ue/li260-2005fev/g7/ml/ocamldoc_projet.cmo -I +threads \
nom_fichier.ml
```

Inutile de dire la solution que je préfère. Pour éviter la deuxième commande que je considère comme très sous-optimale, je vous suggère de relire le poly précisant le style de programmation à adopter pour réaliser `ocamlpilot`.

Exercice 10 Bravo, vous êtes arrivés au terme de ce TME. Maintenant, il s'agit de faire un peu de ménage. Tout d'abord, sauvegardez le fichier que vous venez d'écrire quelque part. Le répertoire `li260-2005fev` de votre home directory est probablement un bon endroit pour cela. OK, une fois que c'est fait, réfléchissons un peu à ce que l'on veut programmer : finalement, seule la fonction de l'exercice 8 est intéressante pour la suite du développement d'OcamlPilot. Conservez donc cette fonction ainsi que les variables globales dont elle a besoin dans un fichier que je vous suggère d'appeler au hasard `client.ml`. Lancez `ocamldoc` sur ce fichier.