# TME numéro 10

**Exercice 1** Écrivez une fonction `init_client : in_channel -> out_channel -> Graphics.color` qui réalise l'initialisation du client : celle-ci consiste à lire sur le canal d'entrée les couleurs proposées par le client. Le client envoie sa couleur grâce à la chaîne « `PUT COULEUR VAISSEAU` *couleur*`\r` », votre fonction vérifie alors que la couleur n'est pas déjà allouée. Le cas échéant, elle rajoute le vaisseau du client à l'ensemble des vaisseaux du jeu (à une position aléatoire dans l'espace de jeu), elle envoie au client la chaîne « `VAISSEAU ENREGISTRE\r` » et retourne la couleur sélectionnée par le client. En revanche, si la couleur est déjà utilisée, votre fonction doit renvoyer la chaîne « `ERREUR COULEUR UTILISEE\r` ». Enfin, pour tous les cas restants, le serveur renvoie la chaîne « `ERREUR SYNTAXE\r` ». De plus, tant que les couleurs sont déjà utilisées ou bien que le protocole OcamlPilot n'est pas respecté, votre fonction boucle sur les actions ci-dessus. Réfléchissez bien si vous avez besoin d'un mutex et, le cas échéant, où vous en avez besoin.

**Exercice 2** Écrivez une fonction `client : Unix.file_descr -> unit` prenant en argument une socket de service et gérant entièrement le client pour lequel cette socket a été créée jusqu'à la déconnexion de celui-ci. Cette fonction doit donc commencer par récupérer la couleur du client puis, une fois cette phase d'initialisation achevée, elle répond à toutes les requêtes du client, que ce soient des « `GET` » ou des actions (`PUT ACTION TIRE`, etc). Pensez bien aux éventuels mutex que vous devez utiliser. Faites aussi attention à rattraper toutes les exceptions. Lorsque le client s'est déconnecté, n'oubliez pas de fermer la socket de service.