# Jeu Streams - Analyse de l’application

## Infos administratives

Numéro de groupe : 43

Etudiant 1 : Qi Joachim

Etudiant 2 : Sallé Robin

Etudiant 3 : Schellekens Arnaud

## Communication

Donnez ici le type des messages échangés (sur les pipes et les sockets).  
Donnez les scénarios d’exécution.

Nous allons mettre en place une structure « message » qui aura comme attributs un code en int et un message en string. Cette structure sera présente dans un fichier entête « message.h ». Ce fichier entête contiendra également différentes constantes représentant les différents codes de communication à mettre comme valeur dans l’attribut « code ».

## Découpe de l’application

L’application sera découpée de la manière suivante :

* Network (tcp-ip)
* Ipc
* Game
* Serveur (main)
* Client (main)

### Modules

Donnez ici le nom des modules et une brève description.   
Donnez les signatures des fonctions et une brève description de ces fonctions.

Modules :

Utils [utils\_v1.h, utils\_v1.c)] -> module contenant toutes les méthodes utilitaires du cours permettant de faciliter les appels systèmes.

Main [message.h, serveur.c (main), client.c (main)] -> module comportant les 2 fichiers exécutables pour l’application et un fichier entête message.h reprenant une structure message qui nous permettra d’envoyer les différents codes de communication et les messages entre les exécutables.

Ipc [ipc.h, ipc.c] -> module composé d’un fichier d’entête ipc.h contenant toutes les clés nécessaires à la création des sémaphores et de la mémoire partagée, et d’un fichier d’implémentation ipc.c contenant toutes les implémentations utiles pour les sémaphores et pour la mémoire partagée.

Network [network.h, network.c] -> module composé d’un fichier d’entête network.h comprenant le port du serveur, l’ip… et un fichier d’implémentation network.c contenant toutes les implémentations utiles pour le réseau.

Game [game.h, game.c] -> module composé d’un fichier d’entête game.h comprenant les signatures des méthodes et certaines constantes utiles pour le jeu.Il y aura aussi un fichier d’implémentation game.c contenant les implémentations utiles pour la logique et le déroulement du jeu en lui-même.

Fonctions :

Network :

int initSocketServer(int port) -> méthode pour initialiser un socket (création, bind et listen) et renvoie le file descriptor du socket.

Int connectClient() -> méthode permettant de créer une nouvelle connexion entre le socket et un joueur et d’ajouter ce joueur dans le tableau de joueurs. La méthode renvoie le file descriptor de cette connexion.

void closeSocket() -> méthode pour fermer le socket et les différentes connexions avec les joueurs à la fin du jeu.

Ipc :

int createScoresTab() -> méthode pour créer le tableau des scores et le placer dans la mémoire partagée en début de partie. La méthode renvoie l’id du shm.

void placeScore(Player player) -> méthode pour placer le score de partie d’un joueur dans le tableau des scores de la mémoire partagée.

void closeIPC() -> méthode pour fermer la mémoire partagée et le sémaphore.

Game :

char\* createTilesTab() -> méthode pour créer le tableau contenant toutes les tuiles possibles pour le jeu et le renvoyer.

char\* drawTile() -> méthode pour piocher une tuile du tableau de tuiles aléatoirement et la renvoyer.

Bool placeTile(int position, int tile) -> méthode pour placer une tuile et renvoie un boolean si la méthode s’est bien réalisée ou non.

Void sendScore(Player player, int score) -> méthode pour envoyer le score du joueur vers le serveur.

Player\* sortTabScores() -> méthode pour trier le tableau de joueurs en fonction de leurs scores et et le renvoyer.

### Types Utilisateurs

Donnez ici les types utilisateurs que vous comptez créer et une brève description.

On mettra en place une structure Player avec comme attributs son pseudo, son sockfd et son score et la structure Message pour les communications.