**TP Réseau – Snake en réseau**

Par Jérôme Leclercq pour l'IIM – 2022

Pour terminer ce module, après avoir fait un magnifique serveur de chat et désamorcé une terrible bombe, il est temps de rentrer dans le vif du sujet : la mise en réseau d'un jeu vidéo.

En partant d'un jeu vidéo solo (dans notre exemple, un snake), vous allez devoir mettre en place un programme serveur qui aura pour rôle de permettre à plusieurs joueurs de jouer ensemble au jeu du Snake.

Les fichiers que vous allez recevoir comprendront un générateur de projet que nous avons déjà utilisé, dont le rôle est de gérer les dépendances pour nous et générer un projet Visual Studio automatiquement pour que vous n'ayez pas à le faire. Vous devriez donc avoir un projet Client représentant un jeu du snake fonctionnel, conçu à l'aide de la SFML, et un projet Serveur minimal.

Pour vous faciliter la tâche, vous trouverez des fichiers préfixés cl_, sv_ et sh_. Ceux-ci correspondent respectivement à des fichiers clients, serveurs et communs / les deux (shared), ceux ci sont automatiquement ajoutés aux projets Visual Studio par le générateur de projet (xmake).

Pour générer ou actualiser des projets Visual Studio, exécutez le fichier xmake_vs.bat (ou exécutez la commande `xmake/xmake.exe project -k vsxmake` dans le répertoire du projet).

**Objectif :**

Il faut que le jeu soit jouable en réseau, autrement dit que plusieurs joueurs puissent se connecter au serveur, y faire apparaître un serpent qu'ils puissent ensuite contrôler.
Pour ce faire, lors du lancement du programme celui-ci va demander à l'utilisateur d'entrer une adresse IP, et si elle est valide (et que la connexion peut être établie), ouvrir une fenêtre. Les règles du jeu solo doivent être conservées : un serpent grandit lorsqu'il mange une pomme et réapparaît s'il touche un mur, un autre serpent ou une partie de son corps.
Bien entendu, si un joueur se déconnecte, son serpent disparaîtra instantanément de la partie.

Il s'agit d'un projet de groupe, la démocratie ayant décidé des groupes de quatre et vous êtes seize, je m'attends donc à un total de quatre projets.

Le projet sera hébergé sur GitHub.

**Contraintes :**

- Il faut que l'API Socket soit utilisée (pas d'utilisation de sfml-network ;o).
- Le projet serveur ne doit pas incorporer de notion superflue (affichage, fenêtrage, audio, etc.)
- Les bibliothèques externes sont autorisées tant qu'elles ne touchent pas au réseau ou à la sérialisation
- Vous ne devez commit sur GitHub que vos propres modifications, cela afin de pouvoir

identifier qui est responsable de chaque changement (faites des messages de commit clairs).

Quelques indices :
- Découpez votre code en fonctions réutilisables, cela permet d'aider à structurer son programme.
- Pensez votre protocole en termes d'actions simples, et faites usage d'opcodes pour les différencier (communiquez en équipe à ce sujet).
- Certains de vos messages réseau vont devoir être transmis régulièrement, d'autres seulement en cas d'événement.
- Allez-y au plus simple, même si moins efficace au début pour complexifier après.
- Il est courant / plus pratique de faire une structure par type de message qu'on veut transmettre pour faciliter la manipulation.
- Posez des questions en cas de doute / de flou, demandez-moi de réexpliquer les notions que vous ne comprenez pas.

**Évolutions possibles :**
Une fois que vous disposez d'un prototype fonctionnel, vous pouvez l'améliorer avec quelques idées pour des points supplémentaires :
- Écrire un message dans la console lorsqu'un joueur rejoint la partie ou la quitte (Difficulté 2 / 5)
- Donner à chaque joueur une couleur aléatoire qu'il conservera de sa connexion à sa déconnexion, alternativement vous pouvez faire en sorte que le joueur voit toujours son serpent dans une couleur définie (ex : vert) et les serpents ennemis dans une autre. (Difficulté 2 / 5).
- Demander via la console au lancement du programme un pseudonyme au joueur qui sera affiché au-dessus de son serpent pour tout le monde (Difficulté 3 / 5).
- Faire en sorte que plus un serpent grandit, plus celui-ci devient rapide. (Difficulté 3 / 5)
- Optimiser le netcode de votre jeu pour éviter de transmettre plusieurs fois des données inutilement. (Difficulté ? /5)
- Écrire un message dans la console lorsqu'un joueur meurt dans la partie, et de quelle façon (mur ? Joueur ?) Si pseudonymes affichez le pseudo du joueur responsable de la mort du serpent. (Difficulté 3 / 5)
- Comme bon vous semble, faites-en votre jeu !

Pour finir, ce travail est avant tout collaboratif, mais vous pouvez utiliser les branches de git pour expérimenter des choses dans votre coin si vous le souhaitez (privilégiez néanmoins le travail de groupe).

Bonne chance !
**ET POSEZ DES QUESTIONS.**