ENSSAT LANNION

UNIVERSITÉ DE RENNES 1

FISA Informatique – 2e année

Année 2021-2022

Projet contenu multimédia
Orchestre symphonique

1. Description du projet

Ce logiciel permet de simuler un orchestre symphonique.
Ainsi, il est possible de placer plusieurs sources audios (des instruments) dans un espace en 3 dimensions simultanément.
L'auditeur, qui se situe au centre de l'espace, va pouvoir composer un morceau en fonction de ses besoins.

Ainsi, l'auditeur va pouvoir déployer l'orchestre (Dans le cas de notre réalisation le serveur sera lancé sur un terminal de notre machine hôte). Il va également pouvoir utiliser les instruments situés sur la scène sur son ordinateur grâce au logiciel client (dans un second terminal sur la machine hôte). Pour ce faire, il pourra lancer et arrêter les instruments, mais aussi changer leur volume, leur position et leur direction.

2. Mise en œuvre du projet

Le projet à actuellement atteint le niveau de développement N°3.
Ainsi, cela signifie que le logiciel, en plus de permettre les fonctionnalités citées dans la description du projet, permet également d'exécuter plusieurs instruments en parallèle grâce à un système de partitions d'une part, et de lancer plusieurs instruments simultanément dans un script bash par exemple. Ainsi, les instruments vont s'exécuter de manière autonome.
De plus, il est possible de lancer un instrument en mode « interactif », ce qui permet de tester toutes ses possibilités.
Vous trouverez l'ensemble des fonctions et l'explication sur leur utilisation en vous référant au readme.md du projet.

Afin de pouvoir expliquer notre démarche de développement et de comprendre l'ensemble de ce dernier, nous avons décidé de repartir d'un code neuf. En effet, il nous paraissait plus aisé de comprendre un code que nous avions-nous-même développé plutôt que de repartir d'un exemple déjà donné (qui nous a cependant permis de comprendre les subtilités de développement en Open-AL).

Le cœur de notre logiciel consiste en l'utilisation et la mise en place de sockets. La subtilité de notre architecture vient du fait que nous avons un socket dit « d'écoute » instancié sur notre serveur (l'orchestre). C'est sur ce dernier que lorsqu'un client s'initialise il entre en communication avec le socket « d'écoute ». A partir de ce moment là, au niveau du serveur nous avons un nouveau thread qui est initialisé. Ce dernier génère un nouveau socket de communication bidirectionnelle entre le client et le serveur, ce qui permet que plusieurs clients puisse se connecter et contrôler indépendamment leur instrument dans notre orchestre via un thread et un socket de communication dédié. Le socket est en fait notre interface de communication réseau permettant d'envoyer de la donnée (ici nos commandes de gestion ainsi que notre demande de prise de contrôle d'un instrument).

Ainsi, nous avons également mis en place une gestion de l'exclusion mutuelle par l'utilisation d'un sémaphore. En effet, si deux instruments se connectent en même temps il pourrait subvenir un problème car ils accèdent tous deux à une ressource dite « critique ». De ce fait, pour éviter ce désagrément, lors de la création d'un thread nous avons le processus qui prend le jeton d'accès (le sémaphore) afin de bloquer l'accès à la ressource auquel il accède en copie. Puis, à la fin de son processus de copie (ici notre structure « t_information » qui comprend le descripteur de socket avec le contrôleur distant, un tableau de sources audio Open-Al et la liste des instruments), le processus libère le jeton (le sémaphore) afin de permettre à la seconde création mise auparavant en attente de s'exécuter.

3. Test du projet

Lors de notre développement, nous avons utilisé le débogueur GDB qui nous a permis de déboguer le programme en y insérant des points d'arrêt. Il permet donc de lancer le programme, d'arrêter l'exécution à un endroit précis, d'examiner et de modifier les variables au cours de l'exécution et aussi d'exécuter le programme pas-à-pas. Cet outil nous a réellement permis d'investiguer plus profondément chacune de nos erreurs afin de déceler les problèmes de notre code.

Afin de tester le bon fonctionnement de notre application nous avons réalisé des tests d'intégrité continus. C'est-à-dire qu'à chaque itération (nouvel ajout de fonctionnalité) à notre code, nous réalisons l'ensemble des test de fonctionnement précédemment passé plus celui de la fonctionnalité ajoutée.

4. Conclusion du projet

Ce projet a donc été pour nous très enrichissant. En effet, il nous a fait découvrir une nouvelle manière de voir la communication inter-machine (réseau local) et de développer un projet complet et vraiment technique.