Choix technologiques – Projet R-Type

1. Introduction

Dans le cadre du projet **R-Type**, notre objectif est de développer un **jeu multijoueur temps réel** entièrement en **C++**, avec une architecture **client / serveur** performante, portable et maintenable.

Pour atteindre cet objectif, nous avons choisi les technologies suivantes :

- **C++** comme langage principal pour sa performance et son contrôle mémoire.
- **SFML** pour le développement du **client** (graphismes, son, entrées utilisateur, réseau simple).
- **Boost.Asio** pour le développement du **serveur réseau** (communication asynchrone, multithreading, fiabilité).
- **Lua** intégré côté serveur pour la logique de jeu et l'IA (scripting dynamique, équilibrage flexible, modifications sans recompilation).

Ce choix technologique répond aux contraintes de **performance**, de **portabilité** et de **réactivité** inhérentes à un jeu multijoueur en temps réel, tout en permettant une **itération rapide sur les mécaniques de jeu** grâce au scripting Lua.

2. SFML – Partie Client

2.1 Présentation

SFML (Simple and Fast Multimedia Library) est une bibliothèque C++ orientée multimédia, fournissant une interface simple pour :

- la création de fenêtres et **rendu 2D**,
- la gestion des **événements** (clavier, souris, manette),
- le **son** et la **musique**,
- ainsi que la **communication réseau** de base.

2.2 Avantages pour le R-Type

Fonctionnalité	Avantage
Simplicité d'utilisation	API claire et orientée objet, développement rapide du client.
Graphismes	Gestion 2D idéale pour le style rétro de R-Type.
Multimédia complet	Sons, musiques et sprites gérés sans dépendances supplémentaires.
Portabilité	Compatible Linux, Windows et macOS sans modification du code.
Intégration réseau basique	Permet de tester facilement la communication client/serveur.

2.3 Conclusion

SFML nous permet de créer un **client fluide, réactif et portable**, tout en limitant la complexité du code.

C'est un choix idéal pour un projet pédagogique et technique comme R-Type, où la priorité est la maîtrise du moteur de jeu et du réseau, sans réinventer toute la couche graphique.

3. Boost.Asio & Lua – Partie Serveur

3.1 Boost.Asio – Présentation

Boost.Asio est une bibliothèque C++ puissante dédiée à la **programmation réseau asynchrone**.
Elle offre des outils avancés pour :

- la gestion du **TCP et UDP**,
- les **threads** et **strands** (accès concurrents sécurisés),
- les **timers** (synchronisation de la boucle serveur),

- la programmation **non bloquante** via callbacks, futures ou coroutines C++20.

3.2 Boost.Asio – Avantages pour le R-Type

Fonctionnalité	Avantage
I/O asynchrone	Gestion simultanée de nombreux clients sans blocage.
Multithreading natif	Exploitation optimale des ressources CPU (strands, thread pool).
Timers précis	Synchronisation fiable des ticks du serveur.
Performance	Faible latence, parfait pour un jeu temps réel.
Évolutivité	Extensible (Boost.Beast, SSL/TLS, WebSocket…).

3.3 Lua – Présentation

Lua est un langage de **scripting léger et performant**, très utilisé dans l'industrie du jeu vidéo (World of Warcraft, Roblox, CryEngine…), permettant de définir :

- la **logique de jeu**,
- l'**IA des ennemis**,
- **Connexion avec L'ecs**
- les **vagues de monstres**,
- l'**équilibrage** du gameplay

3.4 Lua – Avantages pour le R-Type

Fonctionnalité	Avantage
Scripts éditables	Modifications du gameplay **sans recompiler** le serveur.
Flexibilité	Ajustement des paramètres du jeu en temps réel.
Itération rapide	Parfait pour prototyper et équilibrer les mécaniques.
Architecture propre	Sépare **moteur réseau (C++)** et **logique jeu (Lua)**.

3.5 Conclusion Serveur

Boost.Asio assure une base **stable, scalable et performante**, tandis que Lua apporte **flexibilité et modularité** pour la logique de jeu.

Ensemble, ils offrent un serveur :

- performant,
- configurable,
- facile à mettre à jour,
- et adapté au temps réel multijoueur.

4. Architecture globale

Côté	Technologie	Rôle principal	Avantages clés
Client	SFML	Affichage, sons, entrées, réseau simple	Simplicité, rapidité, portabilité
Serveur réseau	Boost.Asio	Networking, threads, synchro	Performance, fiabilité, async
Scripting gameplay	**Lua**	IA, mécaniques, équilibrage	Modification sans recompilation
Langage commun	C++	Cœur moteur & protocole	Contrôle total, performance native

La **séparation des responsabilités** est claire :

- le client SFML gère l'expérience utilisateur,
- le cœur serveur Boost.Asio gère le réseau et le temps réel,
- Lua contrôle la logique de jeu de façon flexible.

5. Complémentarité

Aspect	SFML (Client)	Boost.Asio + Lua (Serveur)
Rôle	Rendu & Input	Réseau & Logique de jeu
Complexité	Faible	Moyenne
Performance	Adaptée 2D temps réel	Haute performance + scripting
Scalabilité	Locale	Multi-clients, extensible

6. Pourquoi ces technologies sont meilleures que d’autres

(Section inchangée mais implicitement valide aussi pour Lua)

→ C++ pour la performance
→ SFML pour un rendu simple, léger et accessible
→ Boost.Asio pour le réseau temps réel scalable
→ **Lua pour un gameplay flexible sans sacrifier la performance**

7. Conclusion générale

Le choix **C++ / SFML / Boost.Asio / Lua** permet de créer un **R-Type multijoueur performant, flexible et modulaire**.

- **Performances** grâce au C++ & Asio
- **Gameplay modifiable à chaud** grâce à Lua
- **Client léger et portable** grâce à SFML
- **Architecture scalable & propre**

Une base technique solide, moderne et pédagogique respectant les objectifs du projet R-Type.