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** (communication asynchrone, multithreading, fiabilité).

Ce trio technologique répond aux contraintes de **performance**, de **portabilité** et de **réactivité** inhérentes à un jeu multijoueur en temps réel.

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 – Partie Serveur

3.1 Présentation

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

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

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

3.2 Avantages pour le R-Type

Fonctionnalité	Avantage
I/O asynchrone	Gestion simultanée de nombreux clients sans blocage.
Multithreading natif	Meilleure exploitation des ressources CPU grâce aux *strands* et au *thread pool*.
Timers précis	Synchronisation fiable des ticks du serveur.
Performance	Faible latence, haut débit, parfait pour un jeu temps réel.
Évolutivité	Compatible avec Boost.Beast pour HTTP / WebSocket, extensible vers SSL/TLS.

3.3 Conclusion

Boost.Asio offre une base solide pour un **serveur stable et scalable**.
Son approche asynchrone permet de supporter plusieurs joueurs sans perte de réactivité, ce qui est essentiel dans un jeu comme **R-Type**.

4. Architecture globale

Côté	Technologie	Rôle principal	Avantages clés
Client	SFML	Affichage, sons, entrées, envoi des commandes réseau	Simplicité, rapidité de développement, portabilité
Serveur	Boost.Asio	Réseau asynchrone, gestion des connexions, logique de jeu	Performance, stabilité, multithreading
Langage commun	C++	Communication directe, structures partagées	Rapidité d'exécution, contrôle bas niveau, compatibilité totale

Cette architecture permet une **séparation claire** entre les rôles :

- le **client SFML** se concentre sur l'expérience utilisateur,
- le **serveur Asio** gère la logique réseau et le temps réel.

5. Complémentarité des deux bibliothèques

Aspect	SFML (Client)	Boost.Asio (Serveur)
Utilisation	Simplicité, développement rapide	Haute performance, robustesse
Type de communication	Envoi/réception de packets (TCP/UDP simples)	Communication asynchrone multi-clients

Complexité du code	Faible	Moyenne à élevée
Scalabilité	Limitée (1 joueur)	Élevée (plusieurs clients simultanés)

En combinant SFML et Boost.Asio, on obtient une architecture **cohérente, performante et bien équilibrée** entre facilité de développement et puissance technique.

6. Pourquoi ces technologies sont meilleures que d'autres

6.1 Comparaison avec d'autres langages

Langage	Avantage	Inconvénients dans notre contexte
Python (ex: Pygame, asyncio)	Simple à apprendre	Trop lent pour le temps réel et la gestion fine des threads.
Java (ex: JavaFX, Netty)	Portable, orienté objet	Gestion mémoire automatique → latence imprévisible (GC).
C# (Unity)	Environnement complet	Trop lourd pour un projet bas niveau type R-Type.
C++	Performances natives, contrôle total	Courbe d'apprentissage plus raide, mais adaptée à Epitech.

✅ **Conclusion** : C++ reste le meilleur choix pour un jeu temps réel nécessitant de la performance et un contrôle précis sur le réseau, la mémoire et les threads.

6.2 Comparaison avec d’autres frameworks graphiques

Framework	Points forts	Raisons de non-choix
SDL2	Très portable, bas niveau	API procédurale plus complexe, moins intuitive que SFML.
OpenGL pur	Contrôle total du rendu	Trop verbeux pour un projet d’école, nécessite un moteur maison complet.
Unreal / Unity	Outils complets, moteur 3D	Surdimensionnés pour un projet 2D multijoueur, manque de maîtrise sur le code bas niveau.

Conclusion : SFML offre l’équilibre idéal entre **simplicité, performance et maîtrise du code**, contrairement aux moteurs tout-en-un.

6.3 Comparaison avec d’autres bibliothèques réseau

Bibliothèque	Avantage	Raisons de non-choix
SFML Network seul	Simple à utiliser	Pas conçu pour gérer plusieurs centaines de connexions simultanées.
Enet	Très rapide en UDP	API plus complexe, moins standard que Boost.
RakNet	Haute performance	Non maintenu, dépendances lourdes.
Boost.Asio	Standard Boost, moderne, compatible C++20	Idéal pour un serveur asynchrone modulaire et évolutif.

Conclusion : Boost.Asio est le standard industriel pour le **C++ réseau asynchrone**, offrant robustesse et compatibilité avec l’écosystème moderne (Beast, SSL, coroutines…).

7. Conclusion générale

Le choix de **C++**, **SFML** et **Boost.Asio** repose sur trois axes majeurs :

1. **Performance et réactivité** — indispensables pour un jeu en temps réel.

2. **Maîtrise complète du code** — aucune boîte noire, gestion fine du réseau et de la mémoire.

3. **Évolutivité et portabilité** — compatible sur tous les systèmes et extensible à d'autres protocoles (WebSocket, HTTPS…).

Ainsi, ces technologies sont à la fois :

- **pédagogiques** (comprendre le fonctionnement interne du réseau et du moteur de jeu),

- **performantes** (latence faible, exécution native),

- **modulaires** (chaque partie du projet peut évoluer indépendamment).

En résumé, **C++ / SFML / Boost.Asio** forment une **base solide, moderne et maîtrisée** pour construire un **R-Type complet, fluide et performant**, tout en respectant les objectifs d'apprentissage et de rigueur technique du projet.