Comparative study of the project R-Type

OCTOBER 2025

EPITECH

Created by : Tiphaine Pautonnier, Nathan Cancel, Alexandre Odriosolo

Introduction

The aim of this project is to develop a multiplayer version of the famous game R-Type in C++, using an ECS (Entity Component System) architecture. This type of project involves making wise technological choices, both in terms of graphics rendering and network management, code organisation and dependency integration.

This comparative study aims to justify the technologies selected for the project by comparing them with other existing solutions. Each choice was motivated by specific criteria: performance, ease of use, compatibility with C++, and suitability for the specific needs of a real-time action game.

We will therefore explore the advantages of C++ as the main language, the benefits of ECS for structuring the game, the choice of the UDP protocol for the network, the use of the Asio library, dependency management with CPM and CMake, and finally graphics rendering with Raylib. Through this analysis, we will highlight the strengths of each technology and their relevance in the context of the project.

Choosing a language: why C++?

For this video game project, C++ was the natural choice. It is a language renowned for its performance, its proximity to hardware, and its ability to manage resources efficiently. In a game like R-Type, where responsiveness and fluidity are essential, C++ allows every aspect of gameplay to be optimised.

In addition to its performance, C++ benefits from a vast ecosystem of libraries and tools dedicated to game development. It is used in many professional engines such as Unreal Engine, which attests to its robustness.

Comparison with other languages :

Language	Performance	Ease of learning	Video game ecosystem
C++	5/5	2/5	5/5
C#	3/5	4/5	4/5 (Unity)
Python	2/5	5/5	1/5

ECS architecture: a modular and high-performance approach

The Entity Component System (ECS) is an architecture that allows games to be structured in a clear and efficient manner. It is based on three pillars: entities (game objects), components (associated data) and systems (processing logic). This separation promotes modularity, code reusability and, above all, optimal performance thanks to better memory management.

In the context of R-Type, where many entities interact in real time (ships, projectiles, enemies, etc.), ECS is particularly well suited.

Comparison of ECS libraries :

Library	Performance	Ease of use	Documentation
EnTT	5/5	3/5	4/5
Flecs	4/5	4/5	3/5
Anax	2/5	4/5	2/5

The network protocol: TCP or UDP?

In a real-time multiplayer game, such as R-Type, data transmission speed is crucial. What matters is not receiving all packets, but receiving the most recent ones quickly. This is where UDP becomes interesting: it is faster than TCP because it does not check for packet reception.

Of course, UDP is not perfect. There may be packet loss or duplication. But by adding an incremental counter and error checking, these problems can be managed effectively.

Comparison TCP vs UDP :

Criterion	TCP	UDP	Suitable for R-Type ?
Reliability	5/5	2/5	Yes (UDP)
Latency	2/5	5/5	Yes
Packet order	Guaranteed	Not guaranteed	Manually managed
Real-time performance	Average	Excellent	Yes

The network library: Asio

To manage network communications, we chose Asio, a powerful and flexible C++ library. It allows you to work in asynchronous mode, which is ideal for online games. It supports both TCP and UDP, and integrates easily into an ECS architecture.

With Asio, you can manage sockets in detail, optimise data exchanges, and build robust and scalable network communication.

Dependency management: CPM with CMake

To simplify the integration of external libraries, we opted for CPM.cmake. This lightweight dependency manager allows libraries to be added directly from their Git repositories, without having to manage submodules or complex configurations.

It works perfectly with CMake, which is the standard for C++ compilation, and makes the project cleaner and easier to maintain.

Comparison of dependency managers :

Tool	Ease of integration	Weight	Complexity
CPM.cmake	5/5	Lightweight	Low
vcpkg	3/5	Heavyweight	Medium
Conan	3/5	Medium	High

The graphics library: Raylib

For graphics rendering, we chose Raylib, a simple, lightweight library that is well suited to 2D games. It offers an intuitive API, clear documentation, and perfect compatibility with C++. It allows you to quickly create effective visuals without getting lost in complex configurations.

Raylib is ideal for a project like R-Type, where gameplay relies on dynamic but relatively simple visual elements.

Comparison with other graphics libraries :

Library	Ease of use	2D rendering	Setup complexity
Raylib	4/5	4/5	Low
SFML	3/5	4/5	Average
SDL	2/5	3/5	High

Conclusion

This project allowed us to make thoughtful technological choices adapted to the constraints of a real-time multiplayer game. The selected tools (C++, EnTT, UDP with Asio, Raylib, and CPM with CMake) form a coherent whole that meets the specific needs of developing a game like R-Type.

Each technology was chosen for its qualities: the performance and flexibility of C++, the modularity of the ECS architecture with EnTT, the speed of transmission offered by UDP, the simplicity and efficiency of Raylib for 2D rendering, and the ease of integration of dependencies thanks to CPM.