R-TYPE — Comprehensive Technical Documentation (C++17 / ECS / UDP / SFML)

Version 2025/09/28 — This document consolidates and details the entire project

# Client Overview



*Figure 1 — Screen* ***Menu*** *: FPS, état Network (“Connected to the server”), starfield, Buttons SOLO/DUO/TRIO/SQUAD/Mode, accès Settings, PLAY, LOCKER.*

Visible details:

* **FPS: 60 :** top left (real-time counter).
* **Connected to the server** : green (indicator of established session).
* **Starfield** : multi-ray stellar particles in the background.
* **Buttons** : SOLO/DUO/TRIO/SQUAD/Mode for the choice of the format, **PARAMETER** (opening panneau), **PLAY** (confirm), **LOCKER** (inventory/skins).



*Figure 2 — Screen* ***Settings*** *: Resolution 800x600, Mode windowed, Quality graphic Medium, Mode color-blind Normal, button CONTROLS, Volume, FPS 30/60, Apply/Back.*

Expected behaviors:

* **Change** : cycling between predefined values.
* **Controls** : remapping keyboard/joystick (persistent).
* **Slider Volume** : 0–100 range with optional audio feedback.
* **FPS 30/60** : Client-side frame rate constraint, to be combined with V-Sync.
* **Apply** : writing the configuration, Back: Back without applying.

# 1. Contexte R-Type (key history & mechanics)

R-Type (Irem, 1987) is a cult classic horizontal shoot 'em up. The player pilots the R-9 “Arrowhead” against the Bydo Empire. The central mechanic is the Force, an indestructible detachable module that serves as both a shield and a weapon—the original design that defined part of the genre.

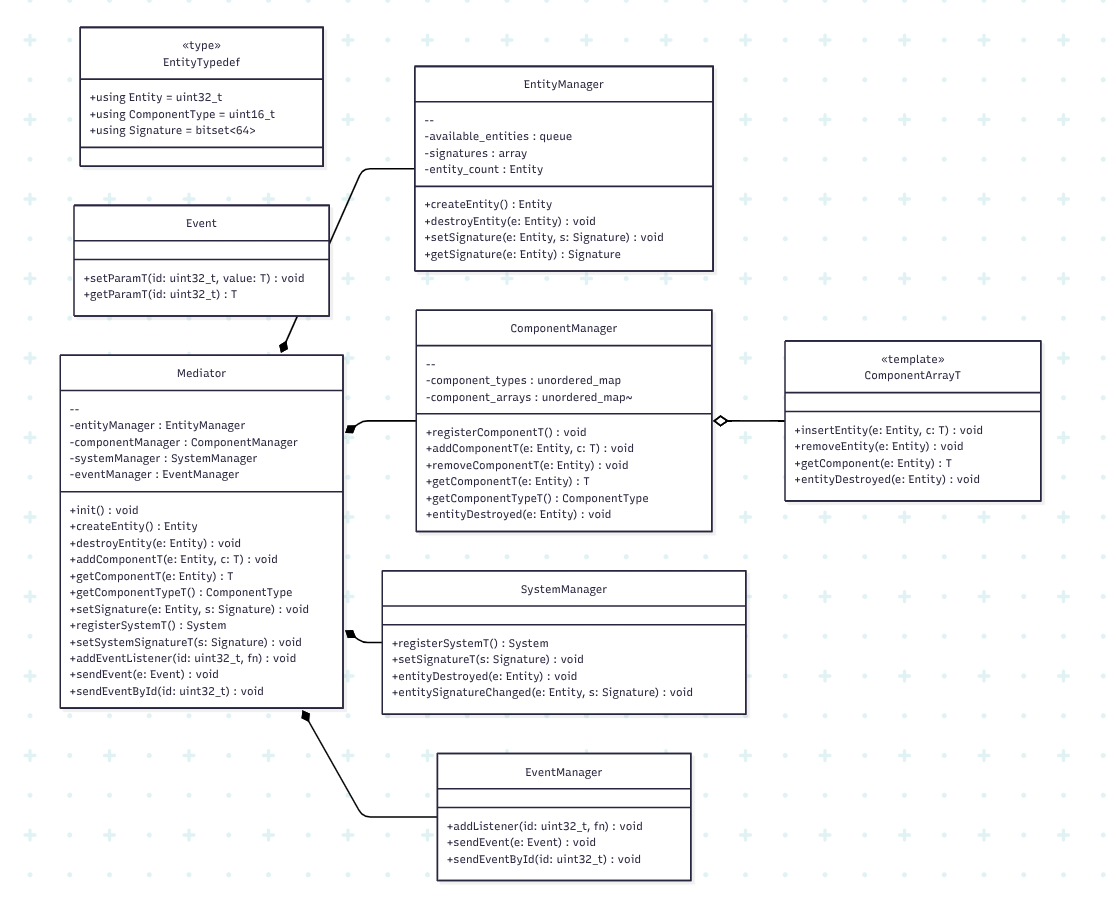
# 2. Requirements of the subject and scope

Objective: a graphic client and an authoritative server communicating in UDP via a documented binary protocol. The engine must be modular (ECS). Minimal gameplay is required (star scrolling, distinct player entities, missiles, waves, collisions, sounds). Degraded network tests (latency/loss/duplication) must be planned.

# 3. General architecture

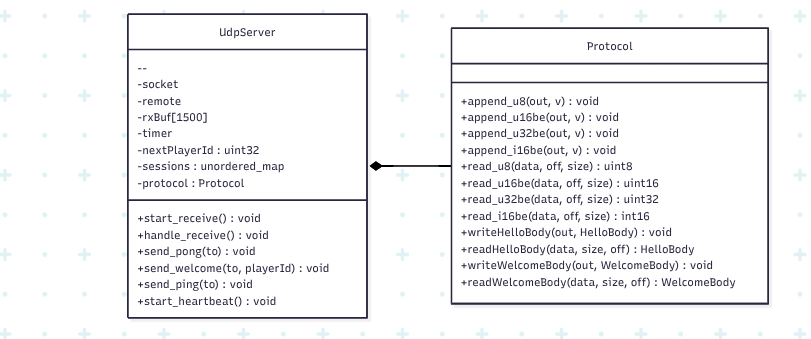
Three blocks: libengine (ECS), SFML Client (rendering/UX/input), UDP Server (authority, Sessions). The Mediator facade orchestrates EntityManager, ComponentManager, SystemManager, and EventManager.

## Diagramme — ECS (rendering image)



*Figure 3 — ECS View: Mediator, Managers, ComponentArray<T>, Event/EventManager*

## Diagramme — Network (rendering image)



*Figure 4 — Vue Network : UdpServer and Protocol (helpers big-endian, handshake, heartbeat).*

# 4. libengine (ECS) Details

* **EEntityTypedef:** `Entity` (uint32), `ComponentType` (uint16), `Signature` (bitset<64>).
* **EntityManager:** allocation/deallocation and signature management.
* **ComponentArray<T>:** compact container for O(1) access**.**
* **ComponentManager:** type registration and mapping to arrays.
* **SystemManager:** registration of Systems and signatures of interest.
* **Event/EventManager:** lightweight Event bus.
* **Mediator:** unified API for manipulating Entities, Components, and Systems.

# 5. Networking & Protocol

The server maintains sessions (endpoint→state). HELLO/WELCOME performs the handshake and assigns a `playerId`. Periodic HEARTBEAT and PING/PONG allow latency measurement and disconnection detection. Packets use big-endian endianness (helpers `append\_\*be` / `read\_\*be`).

# 6. SFML Client & UX in Detail

* **Boucle** : Events → update(dt) → render(). The starfield is based on **particles** (random positions/velocities, recycling at the bottom/top of the screen, pulsed intensity). The FPS counter is calculated on the sliding image medium.
* **Button states :** idle/hover/pressed/disabled, with visual (contrast/border) and audio (click) transitions. **PLAY** validates the selected mode, **PARAMETERS** opens the Settings scene, **LOCKER** opens the inventory (cosmetic).

**Settings** :

* Proposed resolutions, windowed/full screen mode, graphic quality (density of the starfield, possible post-processing).
* Color-blind mode: adapted palettes (protanopia/deuteranopia/tritanopia).
* FPS 30/60: target frame rate switching; V-Sync preferred to avoid tearing.

# 7. Gameplay — Entities, Components, Systems

* **Entities** : Player, Enemy, Projectile, Force, PowerUp, Explosion.
* **Components** : Transform, RigidBody, Sprite, Collider, Health, Weapon, Lifetime, Team, InputState.
* **Systems** : Input, Movement/Physics (dt fixe), Shooting, Collision (AABB grille), Damage, Lifetime, Animation, Render.

# 8. Build & Run

CMake (≥3.10), C++17. Dépendances : **SFML 2.6.x** côté Client, **Boost.Asio** côté Server via **CPM.cmake**. Commandes :  
 cmake -S . -B build -G Ninja -DCMAKE\_BUILD\_TYPE=Release  
 `./ci/build.sh`  
Run : `./r-type\_server` puis `./r-type\_client` (run from the root directory to load assets).

# 9. Quality & CI/CD

* Formatage clang-format, warnings stricts, clang-tidy, sanitizers (ASan/UBSan), tests unitaires (containers ECS, Protocol).
* CI GitHub Actions : build, tests, binary artifacts + assets, CPM dependency cache.

# 10. Network & Functional Tests

* Network (Linux `netem`, Windows `clumsy`) : latency 50/100/200 ms, loss 1–5%, duplication 1%, reordering. Measure p95/p99 of input→display delays, missed snap rates, visible corrections.
* Functional: Menu/Settings navigation, option persistence, spawns, shots, collisions, audio, end-of-game, return to menu.

# 11. Detailed Roadmap

M1 : 'HELLO/WELCOME', Sessions, `PING/PONG`, `HEARTBEAT`, `INPUT/STATE` minimal.  
M2 : Movement, Shooting, Collision, Damage, Lifetime + 3 types d’ennemis + power-ups.  
M3 : Prédiction + interpolation, Lobby/Join IP, tests Network/charge.  
M4 (bonus) : Boss 1, Force attach/detach, replays Client.

# 12. Risks & mitigation

Scope drift (set an MVG), desynchronization (timestamps + seq + logs), CPU load (profiling, batching), technical debt (small PRs, minimal test coverage).

# 13. Appendices

# Repository tree

R-TYPE-main/  
R-TYPE-main/.github/  
R-TYPE-main/.github/workflows/  
R-TYPE-main/.github/workflows/ci.yml  
R-TYPE-main/.gitignore  
R-TYPE-main/.idea/  
R-TYPE-main/.idea/.gitignore  
R-TYPE-main/.idea/R-TYPE.iml  
R-TYPE-main/.idea/misc.xml  
R-TYPE-main/.idea/modules.xml  
R-TYPE-main/.idea/vcs.xml  
R-TYPE-main/CI.md  
R-TYPE-main/CMakeLists.txt  
R-TYPE-main/Client/  
R-TYPE-main/Client/CMakeLists.txt  
R-TYPE-main/Client/include/  
R-TYPE-main/Client/include/Button.hpp  
R-TYPE-main/Client/include/ControlsConfig.hpp  
R-TYPE-main/Client/include/ErrorServer.hpp  
R-TYPE-main/Client/include/GameTypes.hpp  
R-TYPE-main/Client/include/Launch.hpp  
R-TYPE-main/Client/include/Lobby.hpp  
R-TYPE-main/Client/include/Locker.hpp  
R-TYPE-main/Client/include/Menu.hpp  
R-TYPE-main/Client/include/ParamButton.hpp  
R-TYPE-main/Client/include/Parameters.hpp  
R-TYPE-main/Client/include/ParticleSystem.hpp  
R-TYPE-main/Client/include/Player.hpp  
R-TYPE-main/Client/src/  
R-TYPE-main/Client/src/Button.cpp  
R-TYPE-main/Client/src/ControlsConfig.cpp  
R-TYPE-main/Client/src/ErrorServer.cpp  
R-TYPE-main/Client/src/GameTypes.cpp  
R-TYPE-main/Client/src/Launch.cpp  
R-TYPE-main/Client/src/Lobby.cpp  
R-TYPE-main/Client/src/Locker.cpp  
R-TYPE-main/Client/src/Menu.cpp  
R-TYPE-main/Client/src/ParamButton.cpp  
R-TYPE-main/Client/src/Parameters.cpp  
R-TYPE-main/Client/src/ParticleSystem.cpp  
R-TYPE-main/Client/src/Player.cpp  
R-TYPE-main/Client/src/main.cpp  
R-TYPE-main/Dockerfile.builder  
R-TYPE-main/README.md  
R-TYPE-main/Server/  
R-TYPE-main/Server/CMakeLists.txt  
R-TYPE-main/Server/src/  
R-TYPE-main/Server/src/main.cpp  
R-TYPE-main/Server/src/net/  
R-TYPE-main/Server/src/net/Protocol.cpp  
R-TYPE-main/Server/src/net/Protocol.hpp  
R-TYPE-main/Server/src/net/UdpServer.cpp  
R-TYPE-main/Server/src/net/UdpServer.hpp  
R-TYPE-main/Server/src/util/  
R-TYPE-main/Server/src/util/Log.cpp  
R-TYPE-main/Server/src/util/Log.hpp  
R-TYPE-main/assets/  
R-TYPE-main/assets/r-type.otf  
R-TYPE-main/assets/retro.ttf  
R-TYPE-main/assets/sprites/  
R-TYPE-main/assets/sprites/mouvement\_vaisseauxR.gif  
R-TYPE-main/assets/sprites/r-typesheet1.gif  
R-TYPE-main/assets/sprites/r-typesheet10.gif  
R-TYPE-main/assets/sprites/r-typesheet11.gif  
R-TYPE-main/assets/sprites/r-typesheet12.gif  
R-TYPE-main/assets/sprites/r-typesheet13.gif  
R-TYPE-main/assets/sprites/r-typesheet14.gif  
R-TYPE-main/assets/sprites/r-typesheet16.gif  
R-TYPE-main/assets/sprites/r-typesheet17.gif  
R-TYPE-main/assets/sprites/r-typesheet18.gif  
R-TYPE-main/assets/sprites/r-typesheet19.gif  
R-TYPE-main/assets/sprites/r-typesheet2.gif  
R-TYPE-main/assets/sprites/r-typesheet20.gif  
R-TYPE-main/assets/sprites/r-typesheet21.gif  
R-TYPE-main/assets/sprites/r-typesheet22.gif  
R-TYPE-main/assets/sprites/r-typesheet23.gif  
R-TYPE-main/assets/sprites/r-typesheet24.gif  
R-TYPE-main/assets/sprites/r-typesheet25.gif  
R-TYPE-main/assets/sprites/r-typesheet26.gif  
R-TYPE-main/assets/sprites/r-typesheet27.gif  
R-TYPE-main/assets/sprites/r-typesheet28.gif  
R-TYPE-main/assets/sprites/r-typesheet29.gif  
R-TYPE-main/assets/sprites/r-typesheet3.gif  
R-TYPE-main/assets/sprites/r-typesheet30.gif  
R-TYPE-main/assets/sprites/r-typesheet30a.gif  
R-TYPE-main/assets/sprites/r-typesheet31.gif  
R-TYPE-main/assets/sprites/r-typesheet32.gif  
R-TYPE-main/assets/sprites/r-typesheet33.gif  
R-TYPE-main/assets/sprites/r-typesheet34.gif  
R-TYPE-main/assets/sprites/r-typesheet35.gif  
R-TYPE-main/assets/sprites/r-typesheet36.gif  
R-TYPE-main/assets/sprites/r-typesheet37.gif  
R-TYPE-main/assets/sprites/r-typesheet38.gif  
R-TYPE-main/assets/sprites/r-typesheet39.gif  
R-TYPE-main/assets/sprites/r-typesheet40.gif  
R-TYPE-main/assets/sprites/r-typesheet41.gif  
R-TYPE-main/assets/sprites/r-typesheet43.gif  
R-TYPE-main/assets/sprites/r-typesheet44.gif  
R-TYPE-main/assets/sprites/r-typesheet7.gif  
R-TYPE-main/assets/sprites/r-typesheet8.gif  
R-TYPE-main/assets/sprites/r-typesheet9.gif  
R-TYPE-main/assets/sprites/vaisseaux.gif  
R-TYPE-main/ci/  
R-TYPE-main/ci/build.sh  
R-TYPE-main/ci/check\_clang\_format.sh  
R-TYPE-main/ci/test.sh  
R-TYPE-main/cmake/  
R-TYPE-main/cmake/CPM.cmake  
R-TYPE-main/include\_common/  
R-TYPE-main/include\_common/ComponentArray.hpp  
R-TYPE-main/include\_common/ComponentManager.hpp  
R-TYPE-main/include\_common/Components/  
R-TYPE-main/include\_common/Components/Gravity.hpp  
R-TYPE-main/include\_common/Components/RigidBody.hpp  
R-TYPE-main/include\_common/Components/Sprite.hpp  
R-TYPE-main/include\_common/Components/Transform.hpp  
R-TYPE-main/include\_common/Entity.hpp  
R-TYPE-main/include\_common/EntityManager.hpp  
R-TYPE-main/include\_common/Event.hpp  
R-TYPE-main/include\_common/EventManager.hpp  
R-TYPE-main/include\_common/Mediator.hpp  
R-TYPE-main/include\_common/System.hpp  
R-TYPE-main/include\_common/SystemManager.hpp  
R-TYPE-main/include\_common/Systems/  
R-TYPE-main/include\_common/Systems/Physics.hpp  
R-TYPE-main/include\_common/Systems/PlayerControl.hpp  
R-TYPE-main/include\_common/Systems/Render.hpp  
R-TYPE-main/include\_common/Utils.hpp  
R-TYPE-main/libengine/  
R-TYPE-main/libengine/CMakeLists.txt  
R-TYPE-main/libengine/src/  
R-TYPE-main/libengine/src/ComponentArray.cpp  
R-TYPE-main/libengine/src/ComponentManager.cpp  
R-TYPE-main/libengine/src/EntityManager.cpp  
R-TYPE-main/libengine/src/Mediator.cpp  
R-TYPE-main/libengine/src/SystemManager.cpp  
R-TYPE-main/sound/  
R-TYPE-main/sound/Insert\_coin\_sound.mp3  
R-TYPE-main/sound/Menu\_sound.mp3  
R-TYPE-main/sound/Stage2\_sound.mp3  
R-TYPE-main/sound/Stage3\_sound.mp3  
R-TYPE-main/sound/Start\_sound.mp3  
R-TYPE-main/sound/Win\_level\_sound.mp3  
R-TYPE-main/sprite/  
R-TYPE-main/sprite/ErrorServerConnection.png  
R-TYPE-main/sprite/R\_type\_name.png  
R-TYPE-main/sprite/plateforme1.png  
R-TYPE-main/sprite/vaisseau3.png