RE-TYPE, UNE REFONTE DU JEU RETRO R-TYPE

Théo Regheasse, **Corentin Pilard** & **Yacine Yaddaden (Professeur)**

Département de mathématiques, informatique et génie, Université du Québec à Rimouski

Contexte

Notre projet consiste à repenser le jeu d'arcade classique R-Type, sorti en 1987, en C++ tout en intégrant un mode multijoueur coopératif.

Cette initiative s'appuie sur une précédente réalisation similaire dans un laps de temps plus court, qui n'avait pas atteint le niveau de finition souhaité.

Cette fois, nous visons à perfectionner le jeu, avec un code plus performant, en abordant les défis spécifiques de la synchronisation des actions des joueurs, tout en conservant l'esprit et l'expérience authentique du jeu original.

Objectifs

- Réaliser une copie du jeu d'arcade R-Type.
- Implémentation du mode multijoueur coopératif.
- Utilisation de l'architecture ECS pour une gestion efficace du jeu en définissant par avant tous les composants assignables en tant qu'entité.
- Intégration d'une connexion TCP pour réaliser les échanges Serveur/Clients dans le lobby.
- Utilisation d'UDP pour une communication optimisée en jeu.
- Gestion des lags et optimisation du réseau pour limiter les ralentissements.

Méthodologie suivie

- Utilisation de Notion pour la gestion et le détail des tâches via un tableau Kanban.
- Séparation du développement des fonctionnalités majeures par branches sur Git.
- Gestionnaires de ressources : client (Loader / Handler), capacité d'affichage.
- Utilitaires pour simplifier la création d'interfaces utilisateur.
- Réalisation de l'architecture ECS pour manipuler les éléments.
- Création de composants, de systèmes et de scripts pour créer, interagir et contrôler les entités.
- Incorporation du réseau : serveur et client.
- Finition avec le gameplay pour finaliser le projet.

Architecture

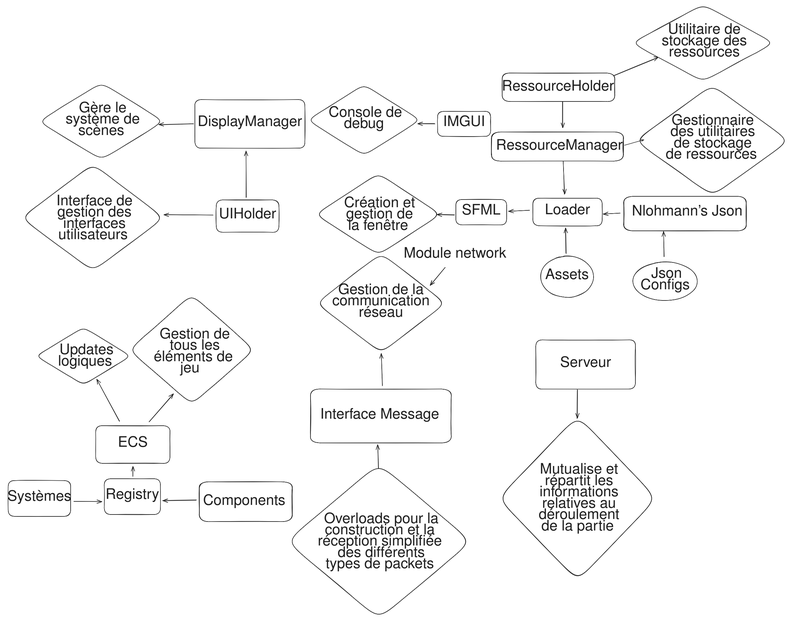

FIGURE 1 – Architecture

Technologies utilisées

- **Langage de programmation**
  - C++ 20 : Langage de programmation moderne et performant, offrant de nombreuses fonctionnalités pour le développement.
- **Système de contrôle de version**
  - GitHub : Plateforme de gestion de version, facilitant la collaboration et la gestion du code source.
- **Librairies**
  - Nlohmann's JSON : Bibliothèque C++ pour travailler efficacement avec les données JSON.
  - SFML : Bibliothèque multimédia en C++ pour le développement de jeux, offrant des fonctionnalités graphiques, sonores et réseau.
  - IMGUI : Bibliothèque C++ pour créer des interfaces utilisateur interactives flexibles rapidement.
- **Environnement de Développement Intégré (IDE)**
  - CLion : IDE conçu par JetBrains pour le développement C++ avec des outils avancés pour le débogage et l'optimisation.

Aperçu

FIGURE 2 – Menu du jeu.

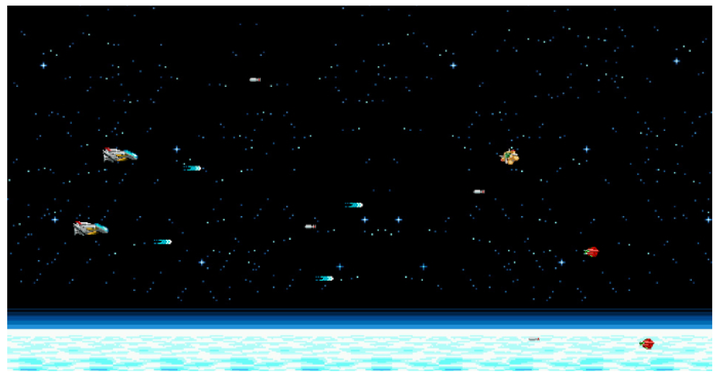

FIGURE 3 – Aperçu du jeu. Deux joueurs qui font feu sur des ennemis qui répliquent.

Conclusion

Au terme de son développement, le jeu :
✓ Reproduit le comportement du jeu d'origine.
✓ Permet de jouer en réseau avec d'autres joueurs.
✓ Utilise l'ECS pour régir les principes fondamentaux de gameplay, comme les actions, la physique et les ennemis.
→ L'ajout d'un mode solo est envisagé pour offrir plus d'options de jeu.

Références

[1] R-type. https://fr.wikipedia.org/wiki/R-Type.

[2] Omar Cornut. Dear imgui. https://github.com/ocornut/imgui.

[3] Laurent Gomila. Sfml. https://www.sfml-dev.org/documentation/2.5.1.

[4] Niels Lohmann. Nlohmann's json. https://github.com/nlohmann/json.

[5] Bjarne Stroustrup and Herb Sutter. cppreference.com. https://en.cppreference.com/.