



AVIGNON
UNIVERSITÉ

Rapport Personnel

Étudiant
Amadou BASS

10/06/2025

Master Informatique
ilsen

UE ALGORITHME ET MODÉLISATION AVANCÉE

Encadrants

Pierre Jourlin
Lahcène Belhadi

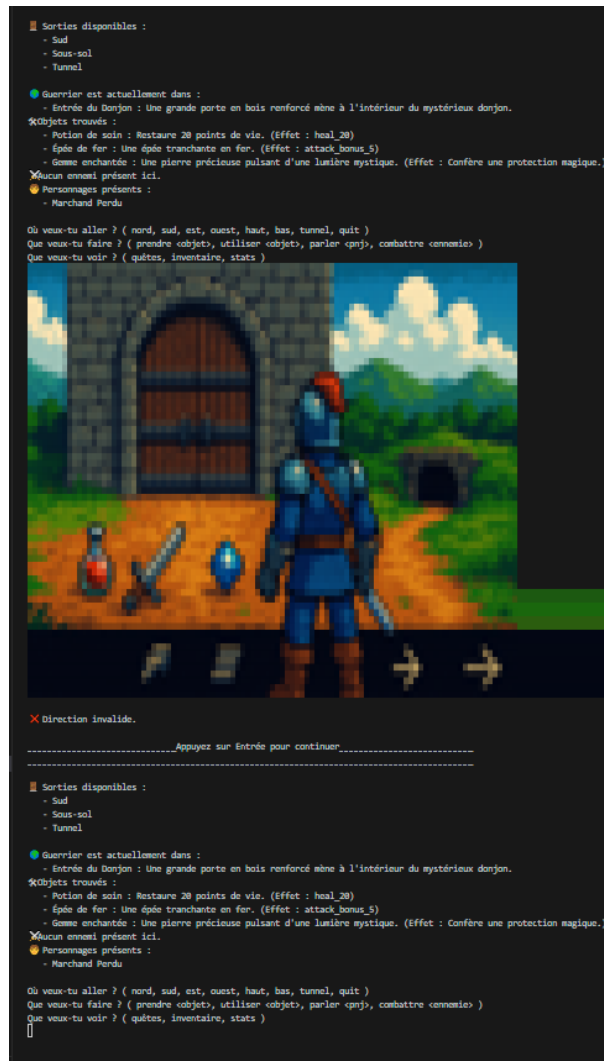
UFR
SCIENCES
TECHNOLOGIES
SANTÉ



CENTRE
D'ENSEIGNEMENT
ET DE RECHERCHE
EN INFORMATIQUE
ceri.univ-avignon.fr

Sommaire

Titre	1
Sommaire	2
1 Présentation du projet	4
2 Architecture générale et organisation de l'équipe	4
3 Introduction	6
4 Objectifs et Responsabilités	6
5 Implémentation technique dans <code>game.rs</code>	6
5.1 Affichage des statistiques du joueur	6
5.2 Affichage de l'inventaire	7
5.3 Affichage des quêtes	7
6 Défis techniques rencontrés	8
7 Réflexion personnelle et acquis	8
8 Conclusion	8



Affichage console du jeu à son lancement

1 Présentation du projet

Dans le cadre de l'UE *Algorithme et Modélisation Avancée* du Master en Génie Logiciel, nous avons développé un jeu de rôle en ligne de commande codé en **Rust**. Ce projet visait à mettre en œuvre des concepts avancés de modélisation algorithmique à travers la conception d'un système interactif cohérent.

Le choix de Rust s'explique par ses avantages techniques : sécurité mémoire sans garbage collector, performance comparable au C/C++, et gestion stricte du typage et de l'emprunt. Ces contraintes ont orienté nos choix d'architecture vers une conception modulaire, robuste et extensible.

Objectifs pédagogiques et techniques

- Comprendre la modélisation de systèmes complexes et interactifs.
- Appliquer des concepts de typage fort, ownership et borrowing.
- Travailler en équipe sur un code partagé avec Git.
- Développer un moteur de jeu basé sur des fichiers JSON de données.

Le jeu propose une exploration textuelle dans un univers de donjon : le joueur évolue entre des salles, interagit avec des PNJ, collecte des objets, engage des combats, et suit des quêtes scénarisées.

2 Architecture générale et organisation de l'équipe

Structure technique

Le projet est organisé en modules :

- **entities/** : personnages, objets, ennemis, salles
- **traits/** : comportements génériques (combat, inventaire...)
- **io/loader.rs** : chargement dynamique via JSON
- **game.rs** : boucle principale et interface utilisateur

L'interface reste textuelle mais propose un affichage d'image (PNG) dans le terminal pour renforcer l'immersion. L'ensemble du contenu (PNJ, quêtes, objets...) est stocké dans des fichiers JSON, rendant le jeu facilement extensible.

Difficultés rencontrées

- Gestion de l'emprunt mutable dans des structures partagées
- Respect des règles strictes de Rust sur les durées de vie (**lifetimes**)
- Maintien de la cohérence entre les modules et les données JSON
- Collaboration sur Git : conflits, rebase, merge de branches

L'utilisation de traits nous a permis de simuler un comportement polymorphe sans hériter de classes. Chaque entité possède ses propres caractéristiques et comportements, définis via des traits implémentés dans différents modules.

Organisation de l'équipe

Notre équipe de 4 étudiants a adopté une approche collaborative, en s'appuyant sur le partage de connaissances, les sessions de codage en groupe, et des échanges réguliers sur les difficultés rencontrées.

- **Olivier Fabre** – Développeur Rust & Responsable Git : architecture initiale, standardisation du code, merges.
- **Tiago** – Product Owner & Développeur : conception du gameplay, équilibrage des mécaniques de jeu.

- **Ndeye** – Scrum Master & Développeuse : gestion de projet, interactions PNJ, dialogues et quêtes.
- **Amadou Bass** – Développeur : gestion de l'inventaire, affichage des statistiques et quêtes.

Nous avons utilisé GitHub comme plateforme de versionnement avec des branches dédiées aux fonctionnalités, des pull requests et des revues de code régulières.

Dépôt du projet : <https://github.com/olfabre/ProjetRustS2>

3 Introduction

Dans le cadre du projet Rust mené pour l'UCE Algorithme et Modélisation Avancée de notre Master en Génie Logiciel, j'ai pris en charge l'implémentation des **statistiques**, de l'**inventaire** et du système de **quêtes**. Ces fonctions fondamentales garantissent la lisibilité des informations clés pour le joueur, tout en assurant une expérience immersive et cohérente.

Ce rapport décrit mon apport personnel à travers le module `game.rs`, en expliquant les choix techniques et les défis relevés, ainsi que les compétences que j'ai acquises pendant ce projet.

4 Objectifs et Responsabilités

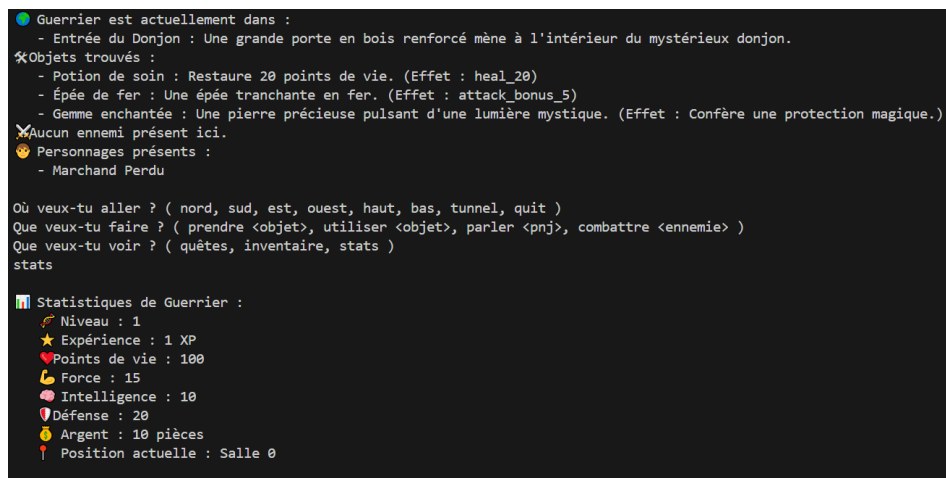
- Gérer l'affichage des **statistiques** du joueur (santé, force, défense, intelligence...)
- Créer un système d'**inventaire** dynamique (prise, utilisation, description)
- Implémenter l'affichage et la gestion des **quêtes actives**
- Garantir une **intégration propre** dans l'architecture modulaire

5 Implémentation technique dans `game.rs`

5.1 Affichage des statistiques du joueur

Les statistiques sont extraites de la structure `Vivant`. Voici un extrait du code :

```
pub fn afficher_stats(joueur: &Character) {
    println!("  Statistiques :");
    println!("- PV : {}", joueur.vivant.health);
    println!("- Force : {}", joueur.vivant.strength);
    println!("- Intelligence : {}", joueur.vivant.intelligence);
    println!("- Défense : {}", joueur.vivant.defense);
    println!("- Niveau : {}", joueur.level);
    println!("- Or : {}", joueur.money);
}
```



```

● Guerrier est actuellement dans :
  - Entrée du Donjon : Une grande porte en bois renforcé mène à l'intérieur du mystérieux donjon.
✖ Objets trouvés :
  - Potion de soin : Restaure 20 points de vie. (Effet : heal_20)
  - Épée de fer : Une épée tranchante en fer. (Effet : attack_bonus_5)
  - Gemme enchantée : Une pierre précieuse pulsant d'une lumière mystique. (Effet : Confère une protection magique.)
✖ Aucun ennemi présent ici.
😊 Personnages présents :
  - Marchand Perdu

Où veux-tu aller ? ( nord, sud, est, ouest, haut, bas, tunnel, quit )
Que veux-tu faire ? ( prendre <objet>, utiliser <objet>, parler <pnj>, combattre <ennemie> )
Que veux-tu voir ? ( quêtes, inventaire, stats )
stats

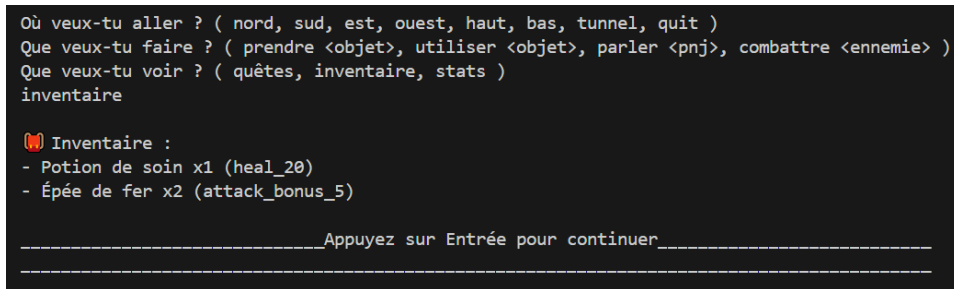
📊 Statistiques de Guerrier :
👤 Niveau : 1
★ Expérience : 1 XP
❤️ Points de vie : 100
💪 Force : 15
🧠 Intelligence : 10
🛡️ Défense : 20
💰 Argent : 10 pièces
📍 Position actuelle : Salle 0
  
```

Figure 1. Affichage des statistiques du joueur

5.2 Affichage de l'inventaire

La structure `Inventaire` contient les objets et leurs quantités. J'ai conçu une fonction d'affichage filtrée :

```
pub fn afficher_inventaire(inventaire: &Inventaire, items: &Vec<Item>) {
    println!("  Inventaire :");
    for obj in &inventaire.items {
        if let Some(item) = items.iter().find(|i| i.id == obj.item_id) {
            println!("- {} x{} : {} (Effet : {})",
                    item.name, obj.quantity, item.description, item.effect);
        }
    }
}
```



```
Où veux-tu aller ? ( nord, sud, est, ouest, haut, bas, tunnel, quit )
Que veux-tu faire ? ( prendre <objet>, utiliser <objet>, parler <pnj>, combattre <ennemie> )
Que veux-tu voir ? ( quêtes, inventaire, stats )
inventaire

📦 Inventaire :
- Potion de soin x1 (heal_20)
- Épée de fer x2 (attack_bonus_5)

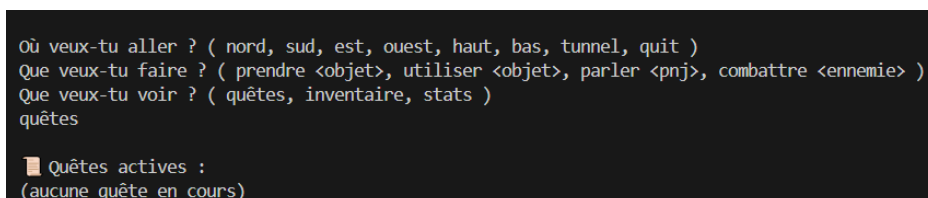
-----Appuyez sur Entrée pour continuer-----
```

Figure 2. Affichage de l'inventaire

5.3 Affichage des quêtes

Les quêtes actives sont stockées sous forme d'identifiants. Le code ci-dessous parcourt ces ID et affiche le contenu correspondant :

```
pub fn afficher_quetes(joueur: &Character, quetes: &Vec<Quete>) {
    println!("  Quêtes actives :");
    for qid in &joueur.quests {
        if let Some(quete) = quetes.iter().find(|q| q.entity.id == *qid) {
            println!("- {} : {}",
                    quete.entity.name, quete.entity.description);
        }
    }
}
```



```
Où veux-tu aller ? ( nord, sud, est, ouest, haut, bas, tunnel, quit )
Que veux-tu faire ? ( prendre <objet>, utiliser <objet>, parler <pnj>, combattre <ennemie> )
Que veux-tu voir ? ( quêtes, inventaire, stats )
quêtes

📋 Quêtes actives :
(aucune quête en cours)

-----Appuyez sur Entrée pour continuer-----
```

Figure 3. Affichage des quêtes actives

6 Défis techniques rencontrés

- Gestion de l'ownership et du borrowing
- Interactions mutables entre JSON, entités et affichage
- Gestion sécurisée des références avec `Option` et `if let`

7 Réflexion personnelle et acquis

Ce projet a été un levier concret pour apprendre Rust. J'ai particulièrement progressé sur :

- La manipulation de structures fortement typées
- L'usage de traits pour la modularité
- La gestion du code en équipe avec Git
- L'adoption de bonnes pratiques de conception

8 Conclusion

Ce projet m'a permis d'appliquer les concepts enseignés dans un cadre exigeant et réaliste. La partie que j'ai développée est au cœur de l'interaction joueur-jeu, et j'ai appris à concevoir du code à la fois robuste, clair et évolutif.

Projet disponible sur GitHub :
<https://github.com/olfabre/ProjetRustS2>