Université d'Artois

Licence Informatique

Space Invaders MVC

Rapport de Projet

Auteur :
Amine Boucif

www.aminebucif.com
LinkedIn Profile
GitHub du Projet

Encadrant :
M. Jean-Marie Lagniez

Date :
Janvier 2026

Table des matières

1 Introduction

Ce projet consiste en une réimplémentation moderne et robuste du classique *Space Invaders*, développée en langage C. L'objectif principal est de démontrer une architecture logicielle rigoureuse basée sur le patron **Modèle-Vue-Contrôleur (MVC)**.

Le code source complet du projet est disponible sur GitHub :
`https://github.com/Aminbcf/Space-Invader`

Le jeu offre une expérience multi-plateforme avec deux interfaces distinctes (graphique via SDL3 et textuelle via Ncurses), tout en garantissant une gestion mémoire irréprochable et des mécaniques de jeu avancées comme le mode « Bullet Hell ».

2 Décisions Techniques

Plusieurs choix technologiques et architecturaux ont été faits pour garantir la performance et la propreté du code :

- **Architecture MVC Stricte :** Le Modèle est totalement indépendant des entrées-sorties. Cela facilite le portage et le test de la logique métier.
- **Gestion Mémoire :** Utilisation systématique de Valgrind pour garantir l'absence de fuites mémoire (standard *Zero Leaks*).
- **Double Vue Polymorphique :** Utilisation d'une structure de données permettant de basculer dynamiquement entre SDL3 (rendu GPU) et Ncurses (rendu CPU/Terminal).
- **Moteur Audio Procédural :** Intégration de la bibliothèque `miniaudio` pour une gestion fluide des sons sans bloquer la boucle de jeu principale.

3 Conception et Modélisation

3.1 Cas d'Utilisation

Avant d'entamer le développement, les interactions du joueur avec le système ont été définies pour couvrir les besoins fonctionnels (jeu, menus, difficulté).

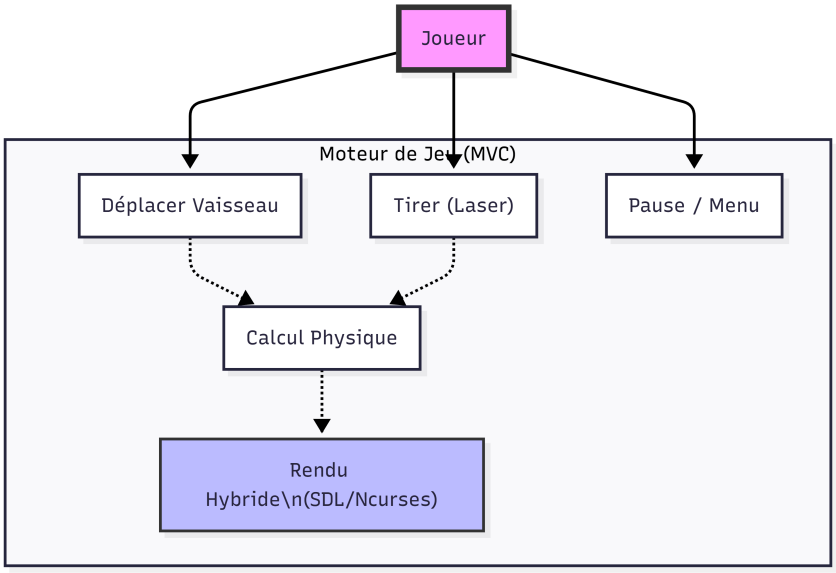

FIGURE 1 – Diagramme des Cas d'Utilisation

3.2 Architecture Logicielle (Diagramme de Classes)

L'architecture repose sur une séparation stricte entre les données (Modèle), le rendu (Vue) et la logique utilisateur (Contrôleur).

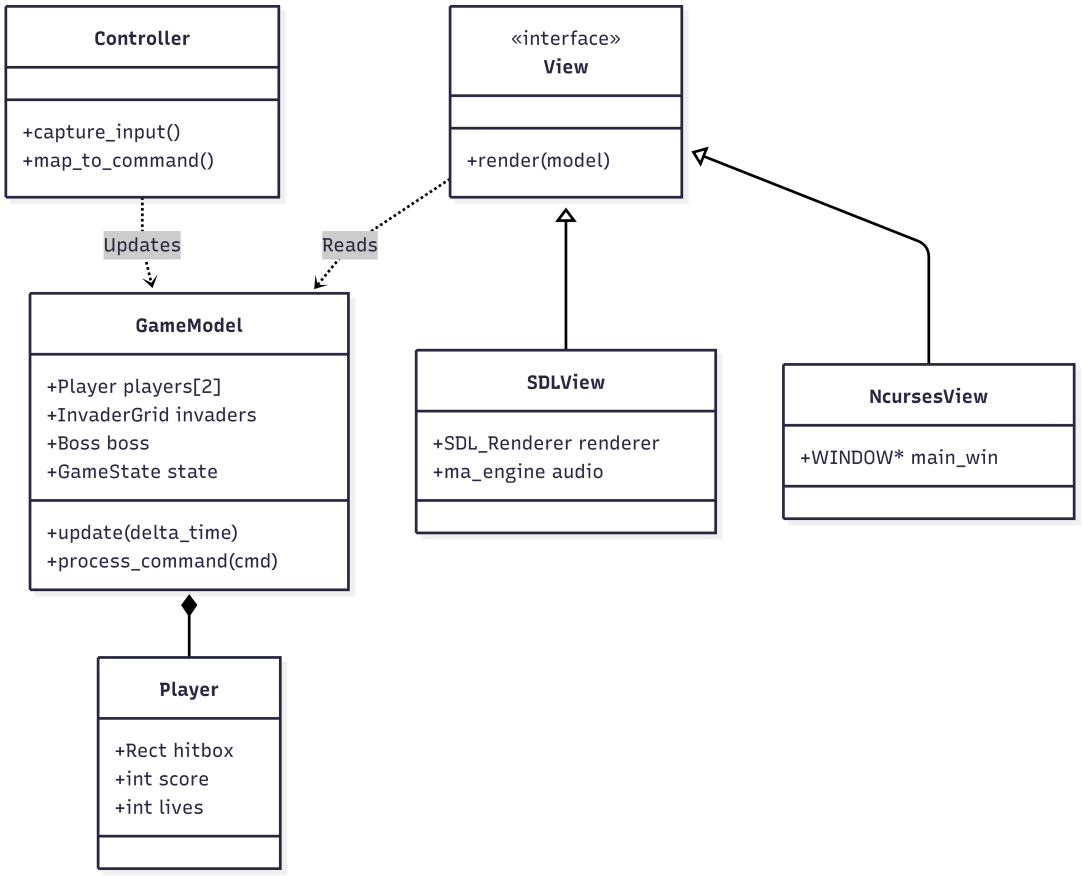

FIGURE 2 – Architecture MVC et Polymorphisme des Vues

3.3 Dynamique du Système (Diagramme de Séquence)

Ce diagramme détaille le flux d'exécution temporel d'une seule frame (1/60ème de seconde), montrant comment le contrôleur met à jour le modèle avant que la vue ne soit notifiée.

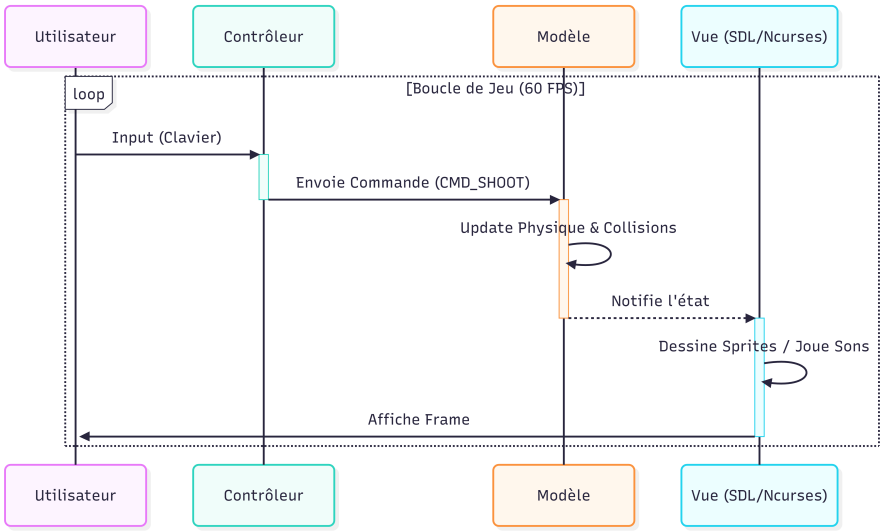

FIGURE 3 – Flux de données dans la boucle principale

4 Difficultés Rencontrées

1. **Développement en Solitaire (Contrainte Majeure) :** Suite au changement de groupe inattendu de mon binôme à la dernière minute, j'ai dû assumer l'intégralité de la charge de travail (conception, code, assets, rapport) seul. Cela a drastiquement réduit le temps disponible pour le polissage final.
2. **Compatibilité Environnement (Arch Linux & SDL3) :** L'environnement de développement (Arch Linux) a posé des problèmes de compatibilité avec les paquets SDL3 officiels, notamment pour l'audio.
   — J'ai dû remplacer les solutions standards par la bibliothèque `miniaudio` pour assurer le son.
   — J'ai intégré un dossier `3rdparty` contenant des versions spécifiques de SDL3, SDL3_Image et SDL3_TTF. En conséquence, le projet nécessite impérativement le compilateur `gcc` pour fonctionner correctement avec ces bibliothèques incluses.
3. **Synchronisation Audio/Modèle :** Le Modèle ne devant pas connaître l'audio, la Vue doit détecter les changements d'état (ex : incrément de `shots_fired`) pour déclencher les sons au bon moment.

5 Conclusion

Ce projet a permis de fusionner des concepts de bas niveau (gestion de la mémoire en C) avec des abstractions architecturales modernes. Le résultat est un jeu complet, fluide et extensible, respectant les standards de l'industrie en matière de séparation des responsabilités.

Sur un plan personnel, ce projet a été un véritable défi. **Ayant dû le réaliser entièrement seul suite à la défection de mon binôme**, j'ai dû faire preuve d'une grande autonomie et d'une gestion du temps rigoureuse. Les contraintes techniques liées à mon OS (Arch Linux) m'ont également forcé à comprendre en profondeur la compilation et l'édition de liens (`gcc`, bibliothèques statiques). Bien que j'aie pris beaucoup de plaisir à développer ce moteur et que j'aurais souhaité implémenter davantage de fonctionnalités, je suis fier du résultat final : un code robuste, sans fuites mémoire, et fonctionnel.