PPC

Ryan Vigoureux, Taiga Matsumoto

Janvier 2026

1 Introduction

2 Architecture

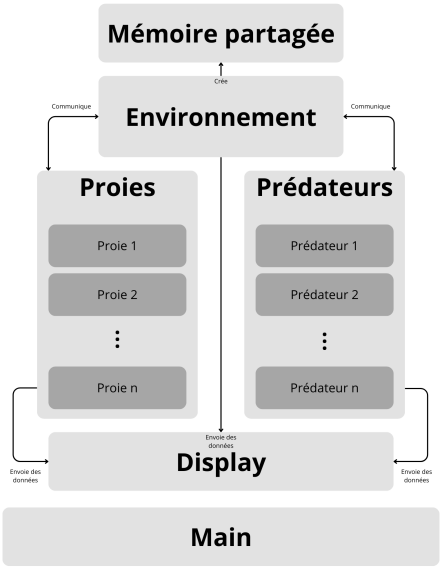

Figure 1: Architecture du projet

3 Connectivité

Les proies et les prédateurs se connectent à l'environnement via une session TCP et peuvent ainsi accéder à ses ressources par l'intermédiaire d'une API. Les processus communiquent également à l'aide de queue.

4 Algorithmes

Lors de la connexion à l'environnement, si celle-ci échoue l'agent essaye de se connecter plusieurs fois jusqu'à ce que la connexion s'établisse.

5 Implémentation

5.1 Environnement

L'environnement est un processus qui gère toutes les ressources dont auront besoin les proies et les prédateurs. Par exemple, les proies ont besoin de manger de l'herbe ; l'environnement s'occupe de faire pousser l'herbe et contrôle la quantité restante d'herbe. Il fonctionne en arrière-plan, indépendamment des interactions des agents, ce qui permet de simuler une croissance biologique continue.
De plus, c'est le processus qui assure l'intégrité des données via des mécanismes de verrouillage pour prévenir les conflits de concurrence.

5.2 Proies & Prédateurs

Les agents (proies et prédateurs) possèdent chacun leur propre processus. Ils peuvent interagir avec leur environnement pour survivre pendant toute la durée de la simulation. Il possèdent un niveau d'énergie et possèdent deux états : actif et passif. Les agents peuvent mourir si leur énergie atteint 0 ou si l'environnement les déclare comme mort. La dynamique de population est assurée par une boucle de surveillance active dans le processus principal, qui remplace l'attente bloquante classique par une écoute des naissances via une file de messages.
Un agent peut

- S'alimenter: si c'est une proie, elle interroge l'Environnement pour savoir s'il reste de l'herbe disponible et si c'est un prédateur il cherche une proie disponible dans le registre de l'Environnement pour augmenter son niveau d'énergie.
- Se reproduire lorsque l'énergie dépasse un certain seuil.
- La mort peut survenir de deux manières: la mort naturelle équivaut à une énergie tombée à 0, et la mort par prédation si une proie a été chassé et qu'elle a été mangée.

5.3 Main

La fonction principale main gère les différents processus de manière active pour détecter les fins de vie et écoute la file pour les naissances. Cela a pour but de

garder un contrôle total sur la population globale et d'assurer un arrêt propre de tous les services une fois la simulation terminée.

Les différents processus discute avec le display a l'aide d'une queue qui affiche les messages, plutôt que chaque processus imprime son message dans le terminal dans son coin. Ce passage a une fonction tiers permet une lecture chronologique et sans interférences, mais également de permettre une future implémentation d'une interface graphique.

6 Tests

Nous avons mené plusieurs tests pour éviter des scénarios comme l'extinction immédiate ou l'explosion démographique. Nous avons essayé plusieurs valeurs de seuils de reproduction (ex: 50 vs 80 unités d'énergie), nous avons observé l'impact sur la pérennité de l'espèce. Un seuil trop bas sature le processeur de nouveaux processus, tandis qu'un seuil trop haut mène à une simulation qui n'évolue pas ou à une extinction avant toute naissance.
Nous avons également changé le réglage du time.sleep(), un temps trop court rendant les agents "hyperactifs", épuisant les ressources de l'environnement trop vite et saturant le terminal. Nous avons stabilisé ce temps pour simuler un rythme métabolique lisible et adaptée au PC.
Dans la nature, le ratio entre deux rangées d'un chaîne alimentaire est de 10 proies pour un prédateur.

7 Exécution

Lors de l'exécution, on peut observer les différents agents interagir avec leur environnement, tandis que celui-ci émet des notifications à chaque changement d'état (par exemple l'enregistrement d'une proie ou d'un prédateur, ou la repousse de l'herbe).

8 Difficultés rencontrées

Les principales difficultés rencontrées concernent d'une part la communication avec l'environnement, notamment l'établissement d'une connexion fiable sur un port libre afin de permettre l'enregistrement correct des agents, et d'autre part la gestion de la reproduction.

La création dynamique de nouveaux processus lors d'une naissance s'est révélée complexe : bien que l'utilisation d'une file de messages permette de signaler l'apparition d'un nouvel agent, celui-ci ne s'enregistrait pas toujours correctement auprès de l'environnement, ce qui posait des problèmes de cohérence dans le suivi global de la population.

9 Points d'amélioration

9.1 D'un point de vue de la réalité la simulation

Pour améliorer notre projet de simulation et la rendre plus réaliste, nous pourrions nous pencher sur les points suivants:

- Déplacement
  Ajouter des coordonnées X et Y aux agents et à l'herbe, simuler les déplacements et faire baisser l'énergie en fonction de la distance parcourue.

- Age et capacité
  Ajouter un paramètre d'âge où un jeune ou vieux prédateur à une chance de réussite de la chasse plus faible qu'un adulte, ou respectivement pour la proie une chance de survie plus élevée. Réussir à attraper une proie (ou à s'échapper pour la proie) diminuerait (ou augmenterait) cette chance pour les prochaines fois.

- Reproduction
  Une rencontre serait nécessaire en plus du seuil d'énergie, et la descendance hérite des chances de prédation (ou évitement)

- Météo
  Une fonction simulant les conditions météorologiques et qui peut impacter les actions des agents.

10 Annexe IA

L'IA a joué un rôle de support technique tout au long du développement du projet. Elle a d'abord été utilisée pour accélérer l'écriture de code, en proposant des implémentations de fonctions courantes et des corrections de code. Elle a également servi d'outil d'aide à la réflexion lors de la conception des algorithmes, en permettant de comparer différentes approches possibles pour la gestion de la concurrence, la synchronisation entre processus ou encore la dynamique de reproduction et de mortalité des agents.

Par ailleurs, l'IA a aussi été utilisée comme outil de recherche documentaire rapide, afin d'obtenir des rappels sur des concepts précis (IPC, verrous, sockets TCP, patterns de retry, etc.) et d'identifier des pistes de résolution lors de bugs ou de comportements inattendus.

Enfin, elle a permis de réduire le temps consacré aux tâches répétitives ou à faible valeur ajoutée, comme la reformulation de messages, l'amélioration de la documentation, la génération de texte pour le rapport ou la factorisation de portions de code similaires.