



# Rabbit Invaders

Une architecture distribuée pour le jeu  
vidéo

---

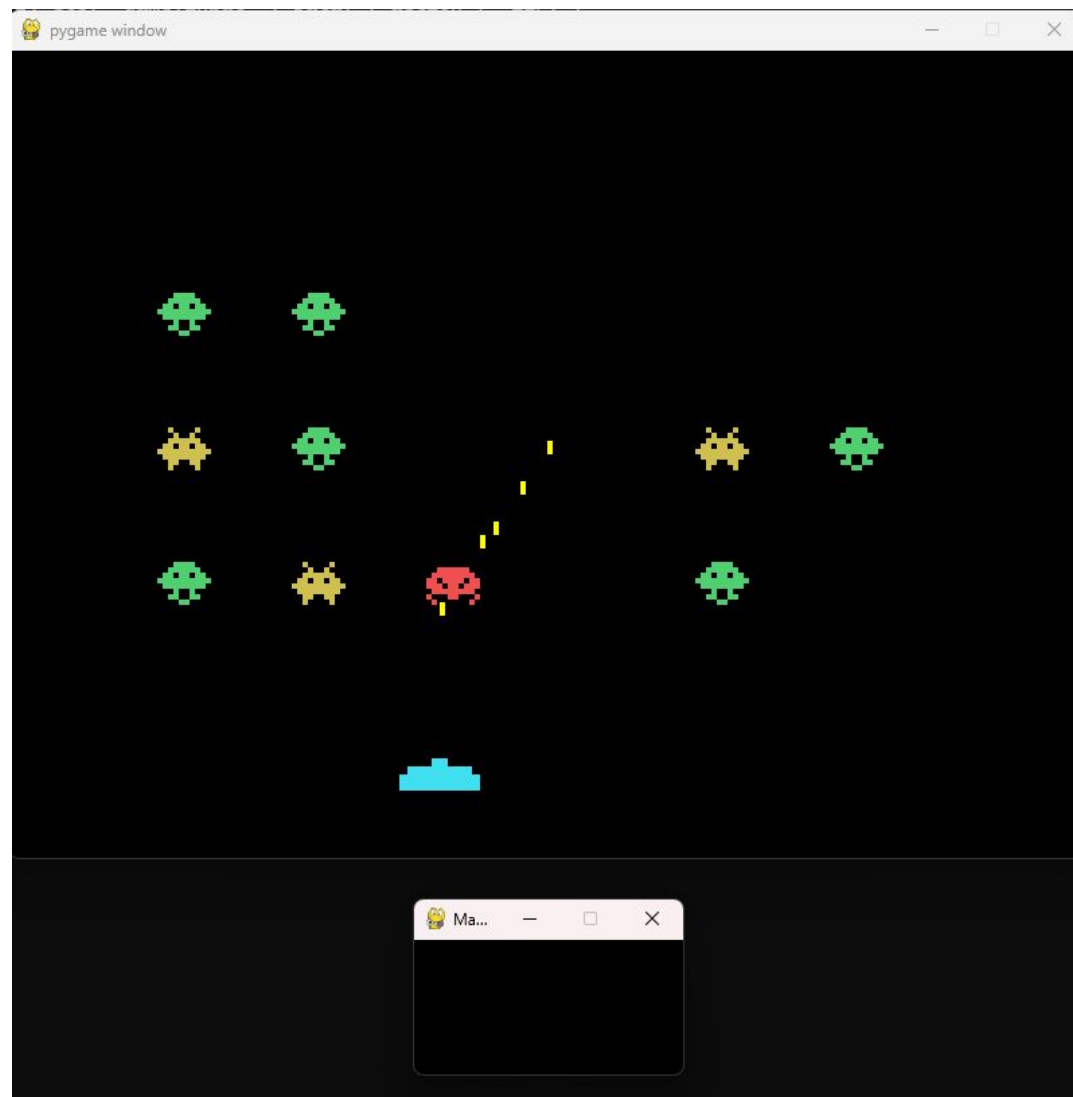
UE CALC • Janvier 2026

# Du Monolithe au Distribué

**L'Objectif :** Mettre en évidence les capacités des MOM (Message Oriented Middleware) à travers un cas ludique.

- ✓ Découplage des composants
- ✓ Communication asynchrone
- ✓ Tolérance aux pannes

**Le Cas d'Étude :** Une implémentation de *Space Invaders* où chaque élément (Manette, Moteur, Écran) est un micro-service autonome.

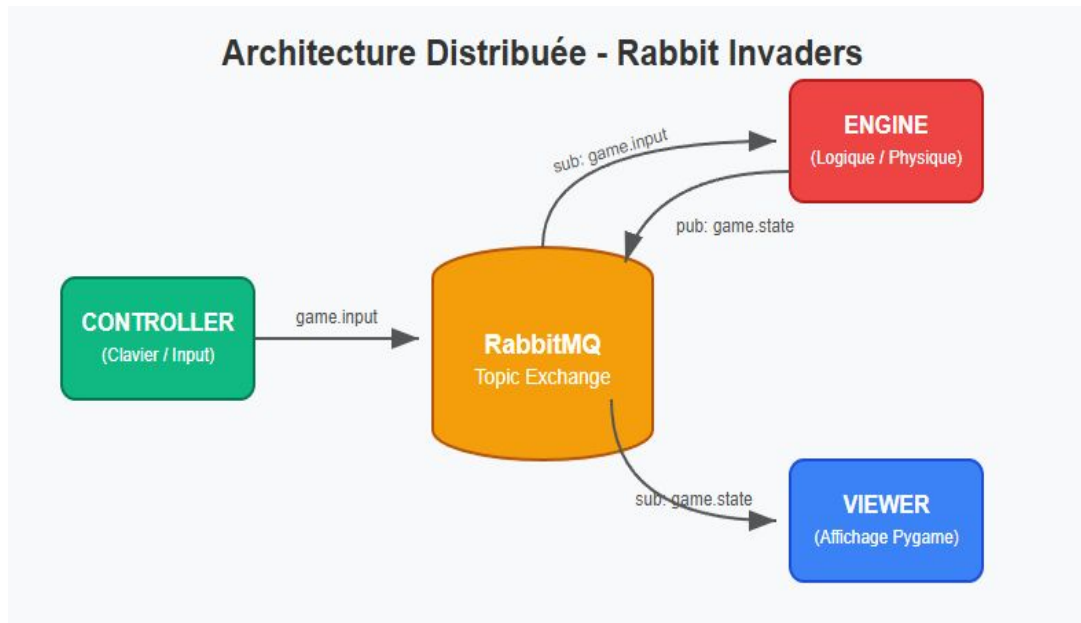


# Architecture Logicielle

- **Controller** : Producteur.  
Envoie les touches sur  
game.input.
- **Engine** : Consommateur &  
Producteur.  
Gère la physique, publie sur  
game.state.
- **Viewer** : Consommateur.  
Affiche l'état du monde.

**Middleware** : RabbitMQ Exchange (Topic)

# Démo 1 : Flux d'Entrées



## Indépendance Totale

La manette n'a aucune connexion directe avec le jeu. Elle ne fait qu'envoyer des messages JSON standardisés.

```
[ENVOI] Clé: 'game.input' | Data: {'action': 'SHOOT'}  
[ENVOI] Clé: 'game.input' | Data: {'action': 'LEFT'}  
[ENVOI] Clé: 'game.input' | Data: {'action': 'SHOOT'}
```

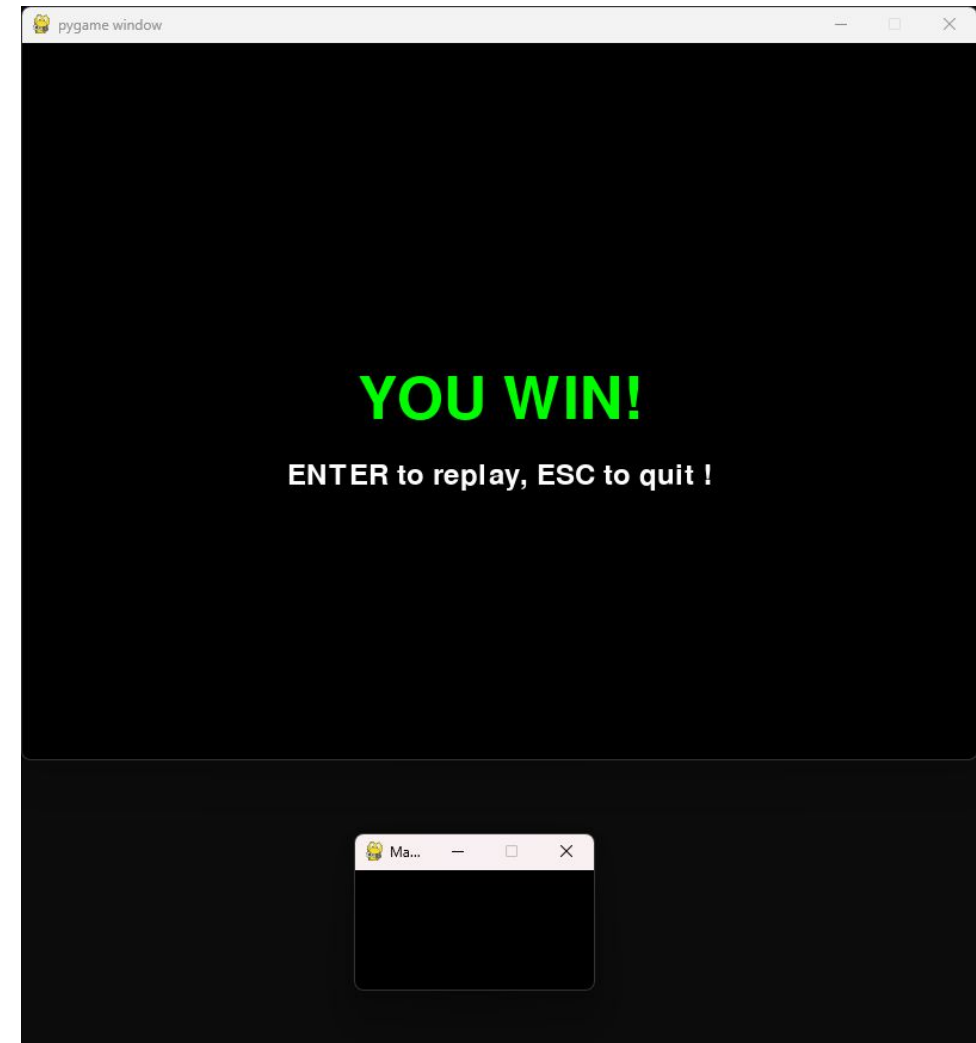
Cela permet de remplacer le clavier par une IA ou une interface Web à chaud, sans redémarrer le moteur.

# Démo 2 : Résilience & Tolérance

## Le "Crash Test"

- 1 **Action** : On tue le processus d'affichage  
· (Viewer) en pleine partie.
- 2 **Résultat** : Le Moteur (Engine) continue de  
· tourner imperturbable. Les aliens continuent de descendre.
3. **Reprise** : On relance le Viewer. Il se reconnect à RabbitMQ, reçoit le dernier game.state et affiche la partie exactement là où elle est rendue.

✓ **Aucune perte de session de jeu.**



# Bilan du Projet

---

## Ce qui est fait

- Jeu complet fonctionnel en architecture distribuée.
- Mise en œuvre propre du pattern **Publish/Subscribe**.
- Code source modulaire et commenté.
- Preuve de concept de la tolérance aux pannes.

## Perspectives

- **Service Sonore** : Ajouter un micro-service qui joue des sons en écoutant les events.
- **Multijoueur** : Gérer plusieurs manettes sur le même canal d'input.
- **Déploiement** : Lancer les modules sur des machines physiques différentes via le réseau.