Simulation Flappy Bird : Moteur de Jeu Hybride Unity / Godot

Ivo TALVET, Timothée M'BASSIDJE

Projet de Simulation et d'Intégration moteur

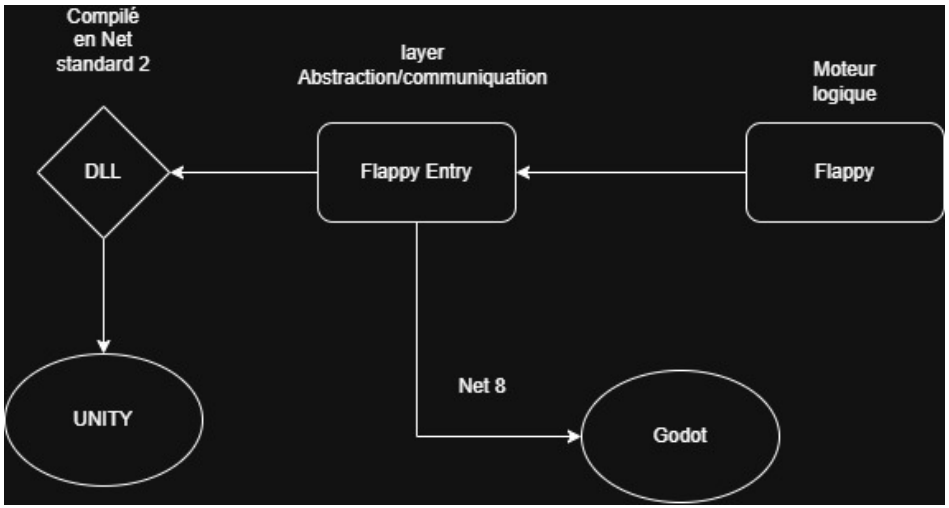
October 30, 2025

Contexte du projet

- Développement d'un environnement de simulation physique de type **Flappy Bird**.
- Implémentation de la logique centrale du jeu en **C# natif** (bibliothèque .dll).
- Intégration de cette logique dans deux moteurs :
  - **Unity** : affichage 3D.
  - **Godot** : affichage 2D.
- Objectif : assurer la **cohérence** entre la simulation logique et la visualisation.

Architecture générale du système

- Séparation stricte entre la **logique de jeu** (Flappy.dll) et la **vue Unity/Godot**.
- Communication via des structures `InputData` et `OutputData`.
- Gestion du cycle :
  1. Mise à jour physique (C# natif)
  2. Synchronisation des obstacles et du joueur
  3. Rendu et interactions dans le moteur hôte



Principe du jeu Flappy Bird

- Le joueur contrôle un oiseau qui doit voler entre des tuyaux sans les toucher.
- La gravité tire l'oiseau vers le bas.
- Le joueur peut **battre des ailes** (Flap) pour remonter brièvement.
- Les tuyaux défilent de droite à gauche à vitesse constante.
- Le jeu se termine lorsque :
  - L'oiseau touche un tuyau.
  - L'oiseau touche le sol ou dépasse le plafond.

Classe `Flappy`

- Représente l'état interne du joueur (l'oiseau).
- Gère les variables physiques :
  - Position verticale (`FlappyHeight`)
  - Vitesse verticale (`VerticalSpeed`)
- Applique la physique :
  - **Gravité** : augmente la vitesse vers le bas.
  - **Saut** : impulsion positive lors du "Flap".
- Met à jour la position selon `deltaTime`.

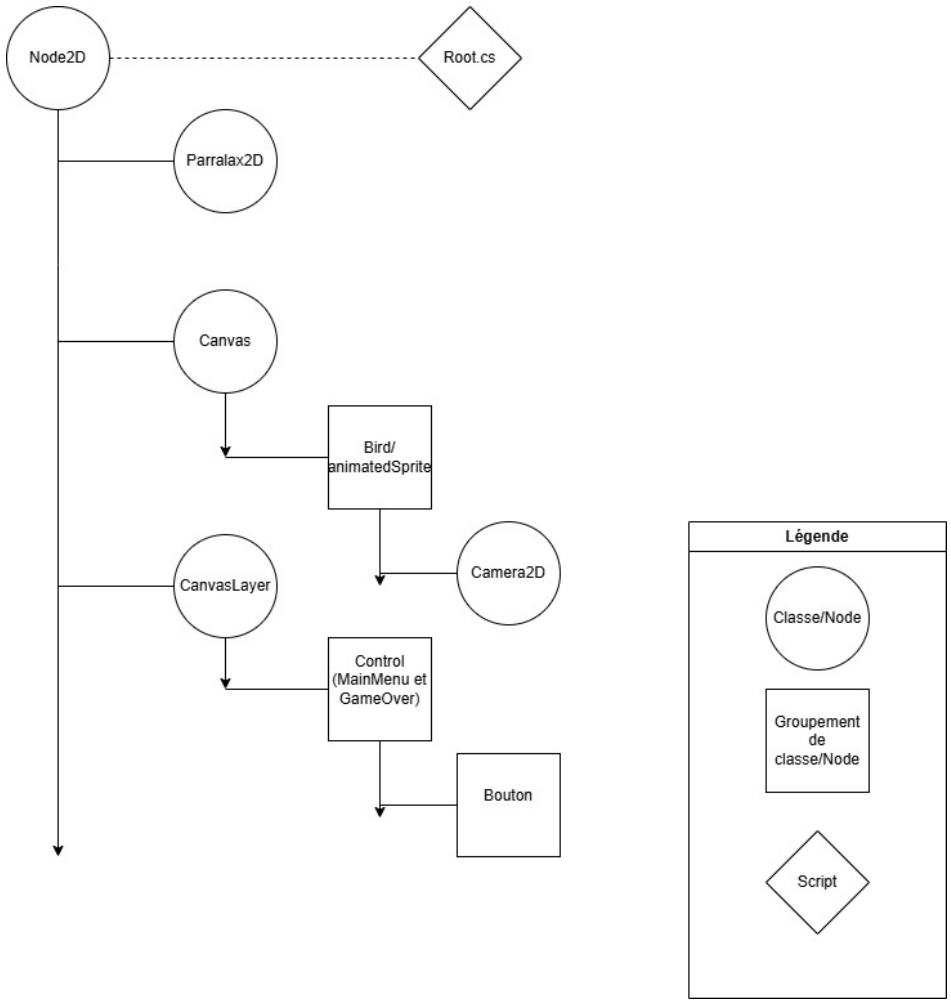
Classe `FlappyEntry`

- Point d'entrée principal de la DLL.
- Initialise tous les éléments du jeu :
  - Oiseau, obstacles, paramètres (gravité, vitesse...).
- Fournit les méthodes publiques :
  - `Init()` : création et configuration initiale.
  - `Update(InputData, ref OutputData)` : mise à jour logique.
  - `Reset()` : réinitialisation du jeu.
- Sert d'interface entre le moteur (Unity ou Godot) et la logique physique.

Vue Godot et hiérarchie de scène

- La logique du jeu est intégrée via la DLL.
- La scène principale contient :
  - **Root (Node2D)** : gère la logique et les dessins.
  - **Bird (Node2D + AnimatedSprite2D)** : oiseau animé.
  - **CanvasLayer** : interface utilisateur (menus, game over).
  - **Parallax2D** : fond défilant.
- Utilisation de la méthode Draw() pour afficher dynamiquement les tuyaux.

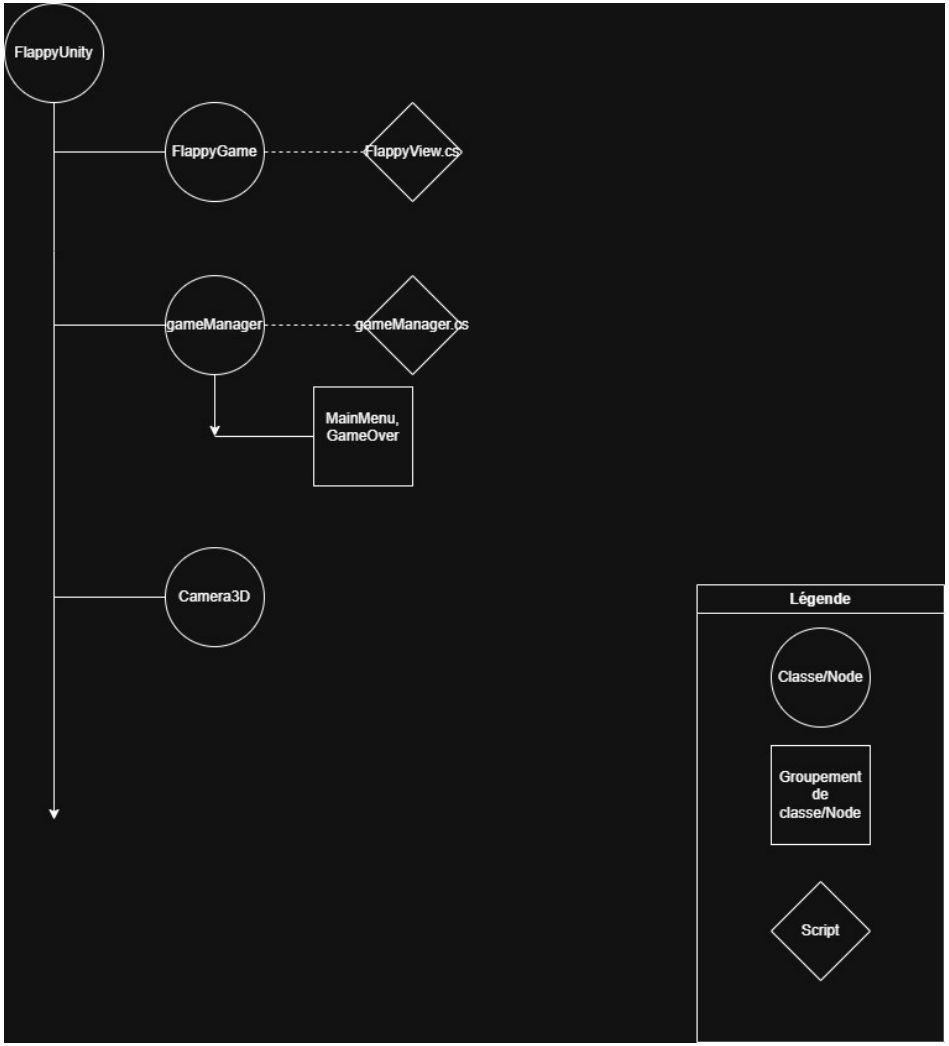
Hiérachie Godot



Vue Unity et hiérarchie

- Intégration du moteur logique via une DLL C#.
- **FlappyView.cs** : script de synchronisation entre Unity et la simulation.
- Hiérarchie Unity :
  - **Main Camera**
  - **FlappyGame** (gère la visualisation)
  - **GameManager** : gère les états du jeu (menu, game over...).
  - **UI Canvas** : interface utilisateur.
- Conversion entre coordonnées logiques et Unity (`UnitsPerMeter`, etc.).

Hiérachie Unity



Démonstration du jeu en temps réel !

Ce que nous avons appris

- Création et utilisation d'une **DLL C#** réutilisable entre moteurs.
- Gestion de la communication entre logique et moteur graphique.
- Comprendre la **hiérarchie des nœuds Godot**.
- Synchronisation d'un moteur 2D (Godot) et 3D (Unity) avec la même logique.
- Compréhension du cycle de mise à jour `Update()` et du rendu.

Perspectives d'amélioration

- Ajout d'un **système de score** et d'un affichage dynamique.
- Intégration de **sons** et d'effets visuels.
- Ajout du **scrolling du background** dans Godot.
- Animation des tuyaux ou éléments de décor.
- Portage du moteur logique vers d'autres frameworks (ex: MonoGame).

Conclusion

- Nous réussie à utiliser une **même logique de jeu** dans deux moteurs distincts.
- Cette approche permet une meilleure **modularité** et **réutilisabilité du code**.
- La séparation entre logique et rendu facilite la maintenance et l'expérimentation.
- Ce projet a permis de relier les notions de **simulation physique**, **architecture logicielle** et **moteurs de jeu**.

Remerciements et questions

Merci pour votre attention !

Questions ?

Ivo TALVET & Timothée M'BASSIDJE