TP2 : Héritage en C# pour Unity

Objectif du TP

Comprendre et appliquer le principe d'héritage pour organiser le code et éviter la duplication dans un jeu vidéo Unity.

Contexte

Vous développez un jeu de type Action-RPG et vous avez remarqué que le code de vos différents types d'ennemis contient beaucoup de duplications. Chaque ennemi a été codé indépendamment, ce qui rend les modifications difficiles et augmente les risques de bugs.

Problématiques identifiées

Dans les scripts `Zombie.cs` et `Skeleton.cs` fournis, vous observerez des duplications importantes :

1. Les mécanismes de base (détection du joueur, mouvement, santé) sont répliqués à l'identique
2. La logique de prise de dégâts est dupliquée
3. L'interaction avec le joueur est répétée
4. Les ajustements de comportement nécessitent de modifier chaque classe séparément

Consignes

1. Analysez les scripts `Zombie.cs` et `Skeleton.cs` et identifiez les parties communes qui pourraient être factorisées.
2. Créez une classe de base abstraite `Enemy` qui contient tout le code commun.
3. Modifiez les classes `Zombie` et `Skeleton` pour qu'elles héritent de la classe `Enemy`.
4. Utilisez les modificateurs d'accès appropriés (private, protected, public) pour les attributs et méthodes.
5. Implémentez des méthodes virtuelles dans la classe de base pour les comportements qui peuvent être surchargés.
6. Ajoutez un troisième type d'ennemi (`Boss.cs`) qui hérite également de `Enemy` mais avec des comportements spécifiques.

Code à améliorer

Les fichiers `Zombie.cs` et `Skeleton.cs` sont fournis dans le répertoire du TP. Analysez-les et créez votre hiérarchie de classes.

Critères d'évaluation

- Application correcte du principe d'héritage
- Factorisation efficace du code commun
- Utilisation appropriée des modificateurs d'accès
- Implémentation pertinente des méthodes virtuelles et des surcharges
- Extensibilité de la solution proposée (facilité à ajouter d'autres types d'ennemis)

Rendu attendu

- Un fichier `Enemy.cs` contenant la classe de base abstraite
- Des fichiers `Zombie.cs`, `Skeleton.cs` et `Boss.cs` implémentant les classes dérivées
- Un court document expliquant vos choix d'implémentation et les avantages de l'héritage dans ce contexte

Bonus

- Implémentez un système d'IA plus avancé avec des comportements spécifiques pour chaque type d'ennemi
- Créez une interface d'affichage des statistiques d'ennemi qui fonctionne pour tous les types d'ennemis
- Intégrez un système de spawn d'ennemis qui utilise la hiérarchie de classes pour créer différents types d'ennemis