TP3 : Polymorphisme en C# pour Unity

Objectif du TP

Comprendre et appliquer le principe de polymorphisme pour créer un système d'armes flexible et facilement extensible dans un jeu Unity.

Contexte

Vous travaillez sur un jeu de rôle où le joueur peut utiliser différents types d'armes (épée, arc, baguette magique, etc.). Actuellement, la gestion des armes est implémentée avec beaucoup de conditions et de code répétitif, ce qui rend difficile l'ajout de nouvelles armes.

Problématiques identifiées

Dans le script `WeaponManager.cs` fourni, plusieurs problèmes sont présents :

1. Une longue série de conditions `if/else` pour gérer les différents types d'armes
2. Code difficile à maintenir et à étendre (pour ajouter une nouvelle arme, il faut modifier plusieurs parties du code)
3. Logique d'attaque spécifique à chaque arme mélangée avec la logique de gestion des armes
4. Pas de moyen simple d'ajouter des comportements spécifiques à chaque type d'arme

Consignes

1. Analysez le script `WeaponManager.cs` et identifiez les problèmes liés à l'absence de polymorphisme.
2. Créez une classe abstraite ou une interface `Weapon` qui définit les méthodes et propriétés communes à toutes les armes.
3. Implémentez des classes dérivées pour chaque type d'arme (Sword, Bow, Wand) avec leurs comportements spécifiques.
4. Refactorisez le `WeaponManager` pour qu'il utilise le polymorphisme, simplifiant ainsi le code et le rendant plus extensible.
5. Ajoutez une nouvelle arme (par exemple, une hache) pour démontrer la facilité d'extension de votre système.

Code à améliorer

Le fichier `WeaponManager.cs` est fourni dans le répertoire du TP. Analysez-le et proposez votre implémentation basée sur le polymorphisme.

Critères d'évaluation

- Application correcte du principe de polymorphisme
- Conception d'une hiérarchie de classes cohérente
- Simplification du code de gestion des armes
- Facilité d'extension du système (ajout de nouvelles armes)
- Séparation claire des responsabilités entre les différentes classes

Rendu attendu

- Une classe abstraite ou interface `Weapon`
- Des classes dérivées pour chaque type d'arme
- Une version refactorisée du `WeaponManager`
- Un court document expliquant vos choix d'implémentation et les avantages du polymorphisme dans ce contexte

Bonus

- Ajoutez des effets visuels et sonores spécifiques à chaque arme
- Implémentez un système d'amélioration des armes qui fonctionne pour tous les types d'armes
- Créez un système d'armes à deux mains qui nécessite des comportements spéciaux