Python
Simulateur de créature virtuelle

Contexte

Le projet consiste à créer une application en Python qui simule la vie d'une créature virtuelle. Le joueur doit s'occuper de cette créature en gérant ses besoins quotidiens pour qu'elle reste heureuse et en bonne santé. Ce projet mettra à l'épreuve vos compétences en programmation orientée objet (POO), en manipulation de données et en gestion d'événements dynamiques.

Objectifs pédagogiques

- Maîtriser les concepts de la programmation orientée objet en Python (classes, objets, méthodes, encapsulation).
- Utiliser et intégrer des modules Python pour gérer les entrées/sorties et les événements.
- Manipuler des données de manière dynamique pour suivre l'état de la créature.
- Travailler la gestion des états via des interactions utilisateur.

Fonctionnalités de base

1. **Création de la Créature :**
    - Le joueur peut créer une créature en choisissant un nom et un type (par exemple : chaton, chiot, dragon, etc.). Chaque type peut avoir des caractéristiques spécifiques.
    - Attributs de base : faim, énergie, bonheur, santé, âge.
2. **Gestion des Besoins :**
    - **Nourrir** : Augmente la jauge de faim, mais peut entraîner une baisse de la santé si l'alimentation n'est pas équilibrée.
    - **Jouer** : Augmente la jauge de bonheur mais diminue l'énergie.
    - **Dormir** : Restaure l'énergie mais laisse la jauge de faim diminuer.
    - **Soigner** : Si la créature tombe malade (événement aléatoire), elle peut être soignée via des actions spécifiques.
3. **Événements Aléatoires :**
    - La créature peut tomber malade, rencontrer d'autres créatures (affectant son bonheur), ou connaître des événements climatiques qui influencent son état.

4. **Évolution et Vieillissement :**
    - La créature grandit en fonction du temps et des soins apportés par le joueur. Elle peut évoluer (par exemple, de bébé à adulte) selon certains critères.
5. **Suivi de l'État :**
    - Affichage régulier de l'état de la créature (faim, énergie, bonheur, santé, âge).
    - Le joueur reçoit des alertes si un état devient critique (exemple : « La créature est très fatiguée ! »).

Fonctionnalités avancées (optionnelles)

1. **Personnalisation de la Créature :**
    - Le joueur peut personnaliser l'apparence ou les attributs de la créature à travers des choix (couleurs, traits de caractère, etc.).
2. **Interactions Sociales :**
    - Permettre à la créature de rencontrer d'autres créatures virtuelles, générant des événements aléatoires (amitié, conflit, etc.).
3. **Mini-Jeux :**
    - Intégrer des mini-jeux simples pour augmenter le bonheur ou gagner des objets spéciaux.
4. **Sauvegarde et Chargement :**
    - Les données de la créature (état, âge, attributs) doivent pouvoir être sauvegardées et chargées.

Exigences techniques

1. **Programmation Orientée Objet (POO) :**
    - Utilisation de classes pour modéliser la créature et ses interactions.
    - Méthodes pour gérer les actions du joueur (nourrir, jouer, etc.).
2. **Modules Python :**
    - Utilisation de modules pour les entrées/sorties (ex. sauvegarde) et la gestion d'événements aléatoires.
3. **Manipulation de Données :**
    - Gestion des états de la créature via des structures de données (dictionnaires, listes, etc.).

Critères d'évaluation

- **Qualité de la Programmation** : Respect des concepts POO, clarté et propreté du code.
- **Fonctionnalité** : Les fonctionnalités de base doivent être opérationnelles.
- **Gestion des États** : Cohérence et mise à jour dynamique des états de la créature.
- **Expérience Utilisateur** : Facilité d'utilisation, messages clairs, et immersion dans le jeu.
- **Originalité (Bonus)** : Ajout de fonctionnalités originales et créatives.