Bases de programmation 2 : Projet

IUT SD Aurillac

Consignes :

- Le code rendu doit être un dossier ou une archive NomPrenom contenant tous vos fichiers.
- Vous avez le droit d'utiliser Chatgpt **uniquement** pour vous aider à concevoir votre application. Si vous rendez un code généré via IA sans ajustements de votre part, vous aurez un malus. Sur ce projet, j'attends que vous alliez au delà de ce que l'IA peut proposer.
- Pensez à bien nommer vos variables et fonctions en respectant les conventions de nommage de Python.
- Pensez à ajouter la docstring
- Si vous lisez cette ligne, demandez moi de venir vous voir en cours et montrez la moi, vous aurez un bonus sur votre note de devoir maison. Il y a un nombre de points bonus limité par groupe qui sera réparti entre tous les étudiants m'ayant montré cette ligne.
- Date limite de rendu (via l'ENT) : **Lundi 28 avril (23h59)**
- Ce projet se fait **seul**

Ce qui compte dans la notation :

- Respect des consignes.
- Respect des conventions de nommage de Python
- Documentation
- Présence des tests unitaires (tests pertinents).
- Exécution possible du code de mon côté sans avoir à corriger d'éventuelles erreurs de votre part (indentation, noms de fichiers, code non fonctionnel).

Rendu attendu :

- Un programme fonctionnel et utilisable, dont le code est bien documenté.
- Un dossier ou une archive NomPrenom contenant tous vos fichiers (**code ET tests unitaires**).

Choisissez au choix un des trois sujets suivants :

1. Jeu de combat.
2. Jeu de gestion d'un camp de survie.
3. Gestionnaire de planning pour une entreprise.

Les sujets sont volontairement larges, de manière à vous permettre de les ajuster selon vos envies.

1 Jeu de combat

1.1 Objectifs

Vous devez développer un jeu de combat où le joueur choisit un personnage et affronte plusieurs enemis d'affilée.

1.2 Fonctionnalités attendues

Le jeu pourra se faire en **tour par tour**. A chaque tour, l'utilisateur pourra choisir : d'attaquer, de se soigner, de se défendre ou d'utiliser un sort parmis ses sorts disponibles.

1.2.1 Personnages

Un personnage doit avoir les informations suivantes :
- nom du joueur (à demander à l'utilisateur avant de lancer le combat)
- classe (Guerrier, Mage, Archer, etc.)
- des statistiques : force, soin, points de vie, précision, défense.
- une liste de sorts qui ont chacun : un nom, un comportement (soin, attaque, boost, etc.), une valeur d'attaque / soin, un temps de recharge.

1.2.2 Ennemis

Le joueur va affronter une suite d'ennemis (les uns après les autres). Après certains ennemis, le joueur peut débloquer des nouveaux sorts. La puissance des ennemis doit être croissante.

1.2.3 Mode de jeu

Il s'agit d'un jeu au **tour par tour**. A chaque tour, le joueur peut : consulter ses statistiques et effectuer une action (attaque simple, sort, soin, etc.).

A la fin d'une partie, les informations et statistiques de la partie sont affichés : nombre d'ennemis vaincus, statistiques du personnage, liste des sorts débloqués. Le joueur peut choisir de re-jouer ou d'arrêter le jeu.

1.3 Bonus

1. Ajoutez la possibilité de gagner des objets après avoir battu un ennemi et de les utiliser en combat.
2. Ajoutez une interface graphique simple (texte et boutons).
3. Ajoutez une musique de fond sur les combats.
4. Ajoutez des images pour les combattants (images simples sans mouvent)

1.4 Suggestions concernant le jeu

Commencez par concevoir l'architecture de votre jeu sur papier :
- nombre de modules et leurs contenus,
- format des dictionnaires pour les données,
- formules de calculs des dégats / soins / sorts.
- etc.

Commencez par développer les fonctionnalités simples, puis ajoutez les fonctionnalités plus complexes.

2 Jeu de gestion d'un camp de survie

2.1 Objectifs

Vous devez développer un jeu de survie dans un mode post-apocalyptique envahi par les zombies. L'objectif du jeu est de maintenir un groupe de survivants en vie.

2.2 Fonctionnalités attendues

Le jeu se fera en **tour par tour**. A chaque tour, l'utilisateur pourra choisir : de consulter l'état des survivants et les stocks de ressources et choisir de faire une action (partir en expéditions pour chercher des ressources ou chercher de nouveaux survivants). A la fin de chaque tour, le camp peut être attaqué par une horde de zombies (taille variable).

2.2.1 Survivants

Chaque survivant a les caractéristiques suivantes :
- nom du survivant,
- des statistiques (force, précision, santé)

2.2.2 Ressources

Pour survivre, les survivants ont besoin de :
- eau
- nourriture
- munitions

Chaque survivant consomme 1 quantité d'eau et 1 quantité de nourriture par jour.

Si les survivants manquent de nourriture et d'eau, ils perdent de la vie.

2.2.3 Défense contre les hordes de zombies

Lorsque qu'une horde de zombies attaque le camp, les survivants doivent se défendre. Pour cela, ils utilisent (s'il en reste) des munitions. Les munitions permettent de tuer en un coup un zombie.

S'il n'y a plus de munitions, les personnages se defendent à main nue : ils peuvent être blessés (il vous faudra choisir un pourcentage de blessures / perte de santé) et les dégats qu'ils font correspondent à leur force. La précision des joueurs doit aussi entrer en jeu.

Le jeu est terminé lorsqu'il n'y a plus de suvivants.

2.2.4 Bonus

1. Ajoutez une interface graphique simple (texte et boutons).
2. Ajoutez une musique de fond.
3. Ajoutez des images pour le camp et les hordes de zombies (images simples sans mouvement)

2.3 Suggestions concernant le jeu

Commencez par concevoir l'architecture de votre jeu sur papier :
- nombre de modules et leurs contenus,
- format des dictionnaires pour les données,
- formules de calculs des dégats faits par les survivants à main nue (précision + force ?). Réfléchissez à un calcul global des dégâts (et non pas au cas par cas). Vous pouvez voir une horde de zombies comme un groupe avec une totalité de points de vie sur laquelle faire des dégâts.
- formule de calcul du nombre de survivants trouvés par recherche
- formule de calcul des ressources trouvées en expédition (en fonction du nombre de survivants ?)

Commencez par développer les fonctionnalités simples, puis ajoutez les fonctionnalités plus complexes.

3 Gestionnaire de planning pour une entreprise

3.1 Objectifs

Vous devez développer un système permettant de gérer les plannings des employés d'une entreprise, avec gestion des horaires et des rendez-vous.

3.2 Fonctionnalités attendues

3.2.1 Employés

Chaque employé a les caractéristiques fixes suivantes :

- nom
- prenom
- intitulé du poste

Chaque employé a des disponibilités (plages horaires pendant lesquelles il est disponible pour des rendez-vous et/ou tâches).

Il est possible d'ajouter ou de supprimer des employés. Il est aussi possible de modifier l'intitulé de poste d'un employé.

3.2.2 Planification des rendez-vous

Vous devez pouvoir attribuer des tâches ou des rendez-vous aux employés en fonction de leur emplois du temps.

3.2.3 Détection des conflits

L'application ne doit pas permettre d'attribuer deux tâches à la même heure pour un même employé.

3.2.4 Affichage du planning

L'application doit permettre d'afficher le planning d'un employé ou de toute l'entreprise.

3.2.5 Bonus

1. Ajoutez une interface graphique simple (texte et boutons)
2. Ajoutez la prise en compte des congès (fonctionnalités CRUD)