Devoir NSI terminale

Jeu de Zelda : quête de sanctuaires

Vous allez créer une version simplifiée du célèbre jeu Zelda. Un chevalier nommé Link doit retrouver la princesse Zelda. Pour cela, il va se déplacer dans un monde enchanté où il va devoir réussir des épreuves qui se trouvent dans des sanctuaires. Chaque sanctuaire abrite une épreuve. Les sanctuaires sont cachés et rattachés à une tour. On compte 4 sanctuaires par tour. Pour révéler le premier sanctuaire, Link doit monter au sommet d'une tour et l'activer. Il lui faut ensuite se rendre au sanctuaire et gagner l'épreuve imposée. En cas de réussite, le sanctuaire suivant devient visible, sinon, Link doit recommencer l'épreuve.
Quand tous les sanctuaires à proximité d'une tour ont été révélé et réussit, il faut changer de tour et recommencer la quête des sanctuaires.
Une fois tous les sanctuaires réussis, la princesse est libérée.

Règles de réussite au sanctuaire :

Les sanctuaires possèdent un niveau de difficulté de 1 à 3, correspondant à une probabilité décroissante de gagner. Le succès à une épreuve d sanctuaire suit les règles suivantes :

- Sanctuaire de niveau 1 : on tire au sort un nombre x entre 1 et 100 compris. Si 0<x<=50 alors défaite sinon succès.
- Sanctuaire de niveau 2 : on tire au sort un nombre x entre 1 et 150 compris. Si 0<x<=100 alors défaite sinon succès.
- Sanctuaire de niveau 3 : on tire au sort un nombre x entre 1 et 200 compris. Si 0<x<=150 alors défaite sinon succès.

Rappels :

- Une fonction ou une méthode peut retourner la valeur *None* pour signifier qu'aucun traitement n'a été effectué ou qu'aucune erreur n'est survenue.
- La bibliothèque *random* permet d'obtenir des aléas. Elle contient les méthodes *random()* et *randint(a,b).*
- Une saisie au clavier est interprétée comme une chaîne de caractère. Pour la convertir en entiers, on utilise la fonction *int(),* exemple : *int(input(« Entrez un chiffre »))*

On modélise le monde enchanté par un plateau de jeu.

Déroulement de l'implémentation du jeu :

- On commence par créer les classes et leurs constructeurs
- Dans chaque classe, on identifie des méthodes dont on aura besoin et on les code
- On crée une première version du jeu qui permet le dérouleent d'une séquence de jeu, à savoir :

- D'afficher les tours
- De choisir la tour à jouer
- D'afficher le premier sanctuaire de la tour choisie
- De passer l'épreuve de ce premier sanctuaire

- On enrichie ensuite cette première version pour permettre la répétition jusqu'à la fin du jeu

1. Citez deux classes que vous pourriez créer et expliquez pourquoi ces classes.
2. Pour chaque classe citée à la question 1. Donnez deux méthodes statiques et deux méthodes dynamiques que vous pourriez coder. Expliquez pourquoi ces méthodes.
3. Création du plateau de jeu

Le monde dans lequel Link se déplace est modélisé par une matrice. Les sanctuaires et les tours sont considérés comme des éléments de jeu. On aura ainsi une classe « Plateau » et une classe « Element ». Chaque case de la matrice contient un élément de jeu. Un élément de jeu peut être vide, c'est-à-dire qu'il n'est ni un sanctuaire, ni une tour.

a. Créez la classe « Plateau » comportant les attributs suivants :
   - Hauteur du plateau
   - Largeur du plateau
   - Matrice représentant le plateau sous la forme d'une grille

Les attributs hauteur et largeur seront donnés en paramètre d'entré lors de l'instanciation de l'objet plateau. La matrice est automatiquement créée via une méthode dédiée nommée « init_plateau » que nous créerons un peu plus tard.

b. Dans la classe plateau, créer une méthode « dans_plateau » qui prend en paramètre d'entrée des coordonnées x et y et qui vérifie que la case (x,y) se trouve bien dans le plateau.

c. Créer une méthode « setXY » qui prend en paramètre d'entrée les coordonnées x et y et un élément de jeu. Cette méthode remplit la case correspondante de la matrice par l'élément de jeu fourni en paramètre.

d. Créer une méthode « getXY » qui retourne l'élément contenu aux coordonnées (x,y)

4. On considère la classe « Element » qui va représenter les éléments de jeu (sanctuaire ou tour). Un élément de jeu possède les attributs suivants :
   - Niveau : niveau de difficulté dans le cas d'un sanctuaire, allant de 1 à 3. Passé en paramètre d'entrée lors de l'instanciation
   - Type : « S » pour un sanctuaire et « T » pour une tour. Passé en paramètre d'entrée lors de l'instanciation
   - Visible : indique si l'élément est visible sur le plateau, par défaut non visible. On pourra utiliser un booléen.
   - Debloque : indique si l'élément a été débloqué dans le cas d'un sanctuaire, par défaut non débloqué. On pourra utiliser un booléen.
   - Tour : dans le cas d'un sanctuaire, indique la tour de rattachement (tuple de coordonnées (x,y)), par défaut à None

   a. Ecrivez la classe « Element »

b. Dans la classe « Element » écrivez la méthode « rendre_visible » qui met à jour l'attribut « visible » à True
c. Dans la classe élément, écrivez la méthode « __str__ » afin qu'elle affiche :
    - des « - » dans toutes les cases du plateau qui sont vides ou non visible
    - des « S » pour les sanctuaires visibles
    - des « T » pour les tours

5. Dans la classe « Plateau », écrire la méthode « init_plateau » qui prend en paramètre d'entrée un taux de remplissage, qui parcours la matrice et, pour chaque case :
    a. Crée un élément de jeu. Pour le paramètre d'entré « niveau », on génère un nombre aléatoire entre 1 et 3
    b. Génère un nombre aléatoire permettant, en fonction du taux de remplissage demandé, de savoir si l'élément crée doit rester vide ou bien s'il faut lui affecter un type « T » et le rendre visible (toutes les tours sont visibles par défaut)
    c. Ajoute l'élément créé à la matrice

    d. Ecrivez, la méthode __str__ qui permet l'affichage du plateau de jeu complet. Pour chaque case on affichera « T » pour une tour, « S » pour un sanctuaire et « - » sinon.

6. Ecrivez, dans la classe « Plateau », une méthode init_sanctuaires » qui parcourt la matrice et qui crée les 4 sanctuaires rattachés à la tour si l'élément courant est une tour. Un sanctuaire est un élément de type « S » et son attribut « tour » doit contenir la tour actuellement traitée. Par défaut un sanctuaire n'est ni visible, ni débloqué.

7. Le programme principal se trouvera dans un nouveau script nommé « zelda.py ». Créer ce script et ajouter y une fonction « zelda » prenant en paramètre d'entrée la hauteur et la largeur du plateau. Cette fonction doit :
    - Instancier un Plateau
    - Appeler la méthode « init_sanctuaires »
    - Afficher le plateau

8. Cœur du jeu :

Nous allons modifier la fonction « Zelda » pour afficher les tours et demander au joueur d'entrer les coordonnées de la tour qu'il veut visiter.
Pour cela :

a. Ajouter dans la classe « Plateau » une méthode « explorer_tour » qui prend en entrée des coorodonnées x et y d'une tour. Cette méthode doit aller chercher l'élément aux coordonnées (x,y) et si c'est une tour, la débloquer.
b. Dans la classe « Element », ajouter une méthode « même_tour » qui prend en paramètre d'entré les coordonnées x et y d'une tour et qui les compare avec les coordonnées de l'attribut tour.
c. Dans la classe « Plateau », ajouter une méthode « rendre_visible_sanctuaire » qui prend en paramètre d'entrée les coordonnées x et y d'une tour. Cette méthode devra parcourir toute la matrice pour rechercher le premier élément de type sanctuaire dont la tour est la même que celle aux coordonnées x et y. On utilisera

la méthode « même_tour » de la classe « Element ». Si la tour correspond alors on rend le sanctuaire visible.

d. Dans le programme principal, demander au joueur les coordonnées de la tour à joueur puis explorer cette tour et rendre visible le premier sanctuaire. Terminer en affichant à nouveau le plateau. Au moins un sanctuaire doit apparaitre.
e. Dans la classe « Sanctuaire », créer une méthode « epreuve » qui rend visible le sanctuaire suivant rattaché à la tour en cours si le joueur a gagné l'épreuve. Pour gagner l'épreuve, il faut coder les règles de réussite aux sanctuaires en fonction du niveau.
f. Compléter le programme principal pour indiquer au joueur le message « Bienvenue au sanctuaire de niveau {niveau du sanctuaire}, une épreuve vous attend, appuyer sur la touche entrée pour la passer » avec {niveau du sanctuaire} remplacé par la valeur adéquate. Lorsque le joueur appuie sur la touche « entrée » la méthode « epreuve » est appelée. Le résultat « Succès ! » ou « Défaite, retentez votre chance » est affiché.
g. En cas de succès au sanctuaire, s'il reste des sanctuaires non débloqués rattachés à la tour, alors le prochain est débloqué. Sinon, on demande au joueur de saisir les coordonnées de la prochaine tour qu'il veut jouer.
h. En cas de défaite au sanctuaire, le joueur doit recommencer l'épreuve jusqu'à ce qu'il réussisse.

9. Le jeu s'arrête quand tous les sanctuaires du plateau ont été explorés. On affiche alors « Gagné ! ». Modifiez le programme principal afin d'obtenir ce comportement.