Travail pratique 2 (20%)

Directives :

1. Ce travail doit être fait en équipe de deux, sauf sur approbation préalable du professeur.
2. La date de remise sera donnée par votre professeur
3. Prenez bien soin de respecter les standards de présentation des algorithmes et du code C#.

Concepts

- Modularité (paramètres)
- Vecteurs

Énoncé

Vous devez concevoir un programme simulant la progression d'un groupe d'aventuriers à travers un donjon. Le parcours du groupe d'aventurier se fera complètement automatiquement, sans intervention de l'utilisateur du programme.

Les éléments mentionnés comme [OPTION] ne sont pas obligatoires, ce sont des suggestions rapportant un point de bonus si elles sont implémentées.

Ce TP est noté sur 20. Soyez autant attentif à l'exécution de votre code (le programme respecte les consignes), qu'à sa qualité (respect des standards, lisibilité, clarté dans le nommage de vos variables et de vos méthodes). Essayez autant que possible de rendre la progression du groupe d'aventuriers facile à suivre à la console.

Groupe d'aventurier

Le groupe d'aventuriers sera constitué d'un nombre de membres entre 1 et 9 inclus (ce nombre sera choisi soit aléatoirement, soit par une entré au clavier). Chaque aventurier aura un nombre de points de vie de 500 au départ de l'aventure. Ils auront un potentiel d'attaque aléatoire entre 1 et 100 (inclus) et un potentiel de défense aléatoire entre 1 et 50 (inclus).

Une fois le groupe d'aventurier créé, affichez un tableau avec une ligne par personnage, une colonne pour les points de vie, une colonne pour le potentiel d'attaque et une colonne pour le potentiel de défense.

[OPTION] Vous pouvez aussi attribuer un nom entré au clavier à chaque aventurier et adapter votre programme en conséquence (par exemple indiquer le nom de l'aventurier attaqué par le monstre).

Donjon

Le donjon sera constitué d'un nombre de salles entre 1 et 9 (inclus). Ce nombre pourra être entré au clavier ou choisi aléatoirement. Dans chaque salle, notre groupe d'aventurier affrontera un monstre, de difficulté croissante. Le niveau du monstre correspond au numéro de la salle (le monstre dans la salle 1 est de niveau 1, celui de la salle 2 est de niveau 2…).

Les règles de création des monstres sont les suivantes :

- Points de vie : le monstre aura un nombre de points de vie aléatoire entre 1 et 500*(niveau du monstre). Par exemple, le monstre dans la salle 1 aura un nombre de points de vie aléatoire entre 1 et 500, celui de la salle 2 aura un nombre de points de vie aléatoire entre 1 et 1000, celui de la salle 3 entre 1 et 1500…
- Potentiel d'attaque : le monstre aura un potentiel d'attaque situé entre 1 et 100*(niveau du monstre). Par exemple, le monstre dans la salle 1 aura un potentiel d'attaque entre 1 et 100, celui de la salle 2 aura un potentiel entre 1 et 200, celui de la salle 3 entre 1 et 300…
- Potentiel de défense : le monstre aura un potentiel de défense situé entre 1 et 50*(niveau du monstre). Par exemple, le monstre dans la salle 1 aura un potentiel de défense entre 1 et 50, celui de la salle 2 aura un potentiel entre 1 et 100, celui de la salle 3 entre 1 et 150…
- [OPTION] Vous pouvez prévoir différents types monstres (dragon, liche, sorcier, gnome, orque…) et en attribuer un au hasard lors de la création du monstre pour pouvoir afficher à l'entrée de la salle une phrase telle que : « Vous affrontez un dragon ».

Système de combat

Lorsque les aventuriers entrent dans une salle, donnez les caractéristiques du monstre (points de vie, potentiel d'attaque et potentiel de défense).

Au cours du combat, vous pouvez considérer que les protagonistes (personnages et monstres) attaquent en même temps en donnant tous un coup sur un ennemi. Les aventuriers attaquent le monstre tous en même temps et le monstre attaque un des aventuriers au hasard. La différence entre le potentiel d'attaque reçu et le potentiel de défense correspond aux points de vie enlevés au personnage ou au monstre attaqué. Vu que les aventuriers attaquent tous le monstre en même temps, vous pouvez considérer cela comme une seule attaque coordonnée.

Votre combat ne résoudra sans doute pas avec une seule attaque. Pensez à mentionner quel aventurier est attaqué par le monstre et à afficher combien de points de vie sont enlevés et combien restent à chaque attaque.

Le combat s'arrête dès que le monstre est mort ou que l'ensemble des aventuriers sont morts. Si le monstre meurt et qu'il reste au moins un aventurier vivant, alors les aventuriers survivants prennent une pause, récupèrent l'ensemble de leurs points de vie (on va considérer qu'ils ont des potions de soins à volonté) et passent à la salle suivante.

[OPTION] Une autre façon de simuler le combat est de considérer que le monstre surprend les aventuriers à leur entrée dans la salle et attaque en premier. Le reste du combat devient alors une alternance d'attaques entre le monstre et les aventuriers.

[OPTION 2] On peut tirer au hasard le fait que les aventuriers ou le monstre attaquent en premier.

Fin de la simulation

À partir du moment où l'ensemble des aventuriers meurent au combat, la simulation s'arrête et indique que le groupe d'aventuriers a péri. Vous pouvez imaginer l'épitaphe que vous voulez.
Si au moins un aventurier parvient à franchir l'ensemble des salles du donjon, alors l'aventure est un succès. Là encore, vous pouvez imaginer l'éloge que vous voulez.

Conseils pour la réalisation

- Ne vous lancez pas dans le code en pensant tout faire d'un seul coup. Prenez le temps de décomposer le problème.
- Si vous n'arrivez pas à utiliser l'approche TOP-DOWN (du plus général au plus spécifique), commencez par un aspect élémentaire. Par exemple, commencez par simuler un combat entre un aventurier et un monstre en fixant les points de vie, potentiel d'attaques et de défenses en dur dans le code. Puis, construisez votre programme petit à petit en ajoutant des couches de complexité au fur et à mesure (choisir les potentiels d'attaques et de défense aléatoirement, comment gérer un combat avec plusieurs aventuriers, comment gérer la mort d'un aventurier…).
- Faites un plan sur Word, PowerPoint ou Excel avec les choses à faire. Et n'hésitez pas à le modifier en cours de route si vous vous rendez compte que quelque-chose ne marche pas. Ce plan n'est pas fait pour être figé, mais pour être un guide auquel vous pouvez vous référer.
- De façon général, allez-y progressivement et pensez à tester le bon fonctionnement de votre programme à chaque fois que vous ajoutez quelque-chose un minimum significatif. Mieux vaut faire trop de tests que pas assez. Si vous attendez trop longtemps, vous risquez d'avoir de multiples erreurs venant de plusieurs endroits et démêler tout ça devient très compliqué.