COMPTE-RENDU PROJET ROGUE PeiP1

Membres du Groupe :

GUERRAOUI Ismaïl
HITMI Younes
WILLIAME-NEURANTER Swann
YANG Benjamin

Gameplay :

1) Interface graphique 4pts

- Initialisation : On initie tout ce dont pygame a besoin et chaque image de certains éléments principaux qui ne peuvent pas être créés dans la fonction add
- On crée un dictionnaire liens qui lit chaque abbrv à une image
- Fonction Add: lie chaque élément des monstres / equipement à une image et l'ajoute dans le dictionnaire pour faciliter la tâche
- Fonction resize : change la taille de chaque image pour s'adapter à la taille de l'écran
- Fonction infos : permet d'afficher toutes les infos sur la droite de l'écran à certaines positions
- Fonction use : affiche la liste des choix possibles (pour le select de inventaire, sort ou shop)
- Fonction readMessage : affiche les messages du jeu en haut de l'écran
- Fonction StartGame : affiche l'écran de départ du jeu
- Fonction endGame : affiche l'écran de fin du jeu
- Fonction Affichage : affiche l'image correspondant à chaque élément de la map (plus de détail sur les commentaires dans le code

2) Pts d'XP 1pt

- Hero.XP : attribut d'instance (int) qui permet d'associer de l'expérience au héros
- Creature.XP : attribut d'instance (int) qui permet d'associer de l'expérience à la créature
- Hero.lvl : attribut d'instance (int) qui permet d'associer un niveau au héros
- Creature.meet(other) : Lorsqu'une créature en tue une autre, elle récupère son XP.
- Game.levels(n) : méthode qui permet au héros d'augmenter de niveaux, avec n les paliers pour lesquels il peut augmenter de niveau (en fonction de l'XP).

3) Inventaire limité 1pt

- Hero.reject(item) : permet de supprimer un item dans son inventaire. Utilisée avec la méthode Game.select(liste).
- Equipment.meet(hero) : méthode qui permet au héros de prendre un équipement. Si son inventaire est déjà plein, il ne peut pas prendre l'élément.

5) Nuage de visibilité 3pts

- Sur la carte, # correspond à une case de fumée
- Une liste self.visible est une liste de coordonnées pour lesquelles les cases ne sont pas recouvertes de fumée. Toutes les autres le sont.
- Fonction addVisible : enlève la fumée sur les salles et corridor où le héros est passé (ajoute la coordonnée à self.visible)
- Fonction getVisibleDirection : permet d'enlever la fumée dans les corridors où le héros passe
- Fonction fullVision : enlève toute la fumée de la map (utilisé pour la potion de visibilité)

7) Diagonales 1pt

- Coord.direction(c) : méthode qui a été modifiée pour permettre d'y ajouter les diagonales.
- Game._actions : attribut de classe, avec ajout des mouvements de diagonale.

Actions :

3) Magie 2pts

- Hero.mp : attribut d'instance (int) qui permet d'associer des points de magie au héros
- Creature.invisible : attribut d'instance (int) qui permet d'associer une invisibilité à la créature
- Invisibilite(creature) : fonction qui va mettre l'abbrv du héros à « 9 » pour qu'il ne soit pas reconnaissable par les monstres jusqu'à qu'il les frappe pendant 5 tours
- Teleportation(creature, unique) : fonction qui permet au héros de se téléporter.
- soin(creature, value) : fonction qui permet de soigner le héros en lui retirant des MP.

4) Magie+ 2pts (superforce,vision,invisibilité) :

- Creature.invincible : attribut d'instance (int) qui permet d'associer le nombre d'actions pendant lequel la créature va être invncible
- Creature.meet(self) : si la créature est invincible, elle ne prend pas de dégât.
- Game.sorts : attribut de classe (liste) contenant tous les différents sorts.
- Game.selectSort(liste) : méthode qui permet de sélectionner un des sorts.
- vision(creature) : fonction qui permet au héros de voir la carte
- invisibilite(creature) : fonction qui permet au héros d'être invisible
- superforce(creature) : fonction qui permet d'appliquer une superforce au héros.
- Class(Sort) : Classe qui permet de créer des objets de type sort, utilisés pour la magie.

Objets :

1) Nourriture 1pt

- Hero.satiete : attribut d'instance (int) qui permet d'associer de la satiété au héros

- Hero.Satiété() : méthode qui décrémente l'attribut satiété jusqu'à 0 à chaque mouvement jusqu'à ce qu'il se nourrisse. S'il ne s'est pas nourri, il perd de la vie petit à petit.

2) Armes 1pt

- Hero.arme : attribut d'instance (object) qui permet d'associer une arme au héros
    - Hero.useWeapon(item) : méthode qui permet d'équiper une arme, et qui remplace celle que le héros avait déjà. Augmente la force du héros.
- Creature.meet(self) : Lors d'une attaque avec une arme, elle perd de sa durabilité. Une fois à 0 de durabilité, elle est détruite.
- Class(Arme) : Classe qui permet d'instancier un équipement de type arme (héritée). Les attributs d'instance supplémentaires sont : « strength ».
- Equipment.use(creature) : Si l'objet est une arme, alors lors de l'usage, elle sera associée à l'attribut Hero.arme vu précédemment.

4) Armures 1pt

- Hero.armor : attribut d'instance (object) qui permet d'associer une armure au héros
    - Hero.useArmor(item) : méthode qui permet d'équiper une armure, et qui remplace celle que le héros avait déjà.
- Creature.meet(self) : méthode qui réalise la rencontre entre une créature et une autre. Si la créature attaquée est le héros, et qu'il possède une armure, alors son armure prend les dégâts et pas ses points de vie. Si l'armure n'a plus de durabilité, elle est détruite.
- Class(Armure) : Classe qui permet d'instancier un équipement de type armure (héritée).
- Equipment.use(creature) : Si l'objet est une armure, alors lors de l'usage, elle sera associée à l'attribut Hero.armure vu précédemment.

5) Amulettes 1pt

- Hero.amulette : attribut d'instance (list) qui permet d'associer une ou plusieurs amulettes au héros
    - Hero.take(elem) : reconnaît si l'élément est un élément « Amulette » et ajoute à la liste « hero.amulette » l'élément sans le rajouter dans l'inventaire
- Hero.amuletteAction(item) : reconnaît quel est le type d'amulette et l'applique de manière passive sur le héros.

6) Solidité 1pt

- Equipment.durab : attribut d'instance qui permet de déterminer la durabilité (solidité) de certains équipements.

Salles :

1) Gestion des salles 1pt

- Room.decorate(map, n) : méthode permettant de décorer une salle différemment en fonction du coefficient n. Si n est aléatoire, alors les salles sont générées aléatoirement.
- Stairs.meet(hero) : méthode modifiée, qui permet de générer un étage différent en fonction de l'étage où se trouve le héros. Par exemple, si il est à l'étage 4, il y aura une boutique.

2) Pièges 1pt

- Game.buildFloorTrap() : méthode qui permet de créer un étage avec des pièges.

3) Boutique 2pts

- Hero.gold : attribut d'instance (int) qui permet d'associer de l'or au héros
    - Hero.take(elem) : reconnaît si l'élément est un élément « gold » et implémente de 1 la valeur hero.gold sans le rajouter dans l'inventaire
- Game.buildFloorShop(shop) : méthode qui permet de créer un étage avec une boutique.
- Class(Shop) : Héritée de Element, elle possède les attributs d'instance shop et price. Shop est une liste et price un int.
- Shop.meet(hero) : méthode qui permet au héros, lorsqu'il rencontre une boutique, d'acheter un objet parmi ceux proposés au prix (attribut shop.price).

4) Trésor 2pts

- Game.buildFloorTresor() : méthode qui permet de créer un étage avec un Trésor.
- Game.monstreAleatoire(collection) : méthode qui permet de désigner un monstre aléatoire dans une salle déjà créée.
- Tresor.meet(hero) : Fonction qui laisse le héros prendre le contenu du trésor s'il a la clé
- Class(Tresor) : Classe qui hérite de Element, dont les attributs d'instance supplémentaires sont : « contenu ». Permet de générer un trésor qui possède un élément aléatoire, et accessible uniquement si le héros possède la clé.
- Creature.cle : Booléen associé à une créature disant si elle possède la clé ou non

Monstres :

1) Poison 1pt

- Creature.poison : attribut d'instance (int) qui permet d'associer à une créature si elle est empoisonnée
- Creature.usage : attribut d'instance (function) qui permet d'associer un effet auxattaques de la créature
- Creature.meet(other) : si la créature qui attaque possède un effet à appliquer, alors elle l'applique sur la créature attaquée.
- Creature.use(other) : méthode qui permet d'appliquer les « pouvoirs » de la créature sur l'autre.

2) Rapides 1pt

- Creature.speed : attribut d'instance (int) qui permet d'associer une certaine vitesse à la créature
- Map.moveAllMonsters() : méthode qui a été modifiée pour réaliser x actions d'une créature, x étant son attribut speed.

Total des points : 27

Petite note : La plupart des effets passifs (tels que le poison ou encore la satiété) ont été appliqués dans la méthode Game.play(). Afin de simplifier le compte rendu, il a été préférable de le préciser ici plutôt que dans chaque point de complexité.

Conclusion :

Ce projet a été très intéressant car il nous a permis d'expérimenter le travail d'un code assez conséquent avec un groupe composé de plusieurs personnes. Il a donc fallu sans cesse communiquer, travailler en groupe et efficacement, compétence nécessaire pour la suite.