BALESTRA Valentina
COSTANTIN Perline
ZAOUN Eya

Rapport du projet de POO Rogue

PeiP 1 - Groupe 1
Année 2022-2023

Table des matières :

1. Gameplay :

• **G.II : (1) Point d'expérience (xp) :** Chaque monstre rapporte un nombre de points d'expérience. Quand le héros change de niveau il gagne en force et regagne tous ses points de vie (hp).

- Creature (Element) : ajout de l'argument d'instance xp initialisé à 1
- Hero(Creature) :
  - Ajout de l'argument d'instance initialisé à 0

- levelup(self) : permet d'augmenter de niveau lorsqu'un certain nombre de points XP est atteint. Augmente également la force et le héros retrouve tous ses hp .

•**G.III : (1) Inventaire limité :** Le héros ne peut stocker que 10 équipements maximum et l'or est compté à part de l'inventaire. Ajout d'une action permettant de détruire un équipement.

- Hero(Creature) :
  - Initialise l'or pour qu'il soit compté hors de l'inventaire
  - take(self, elem) : si la longueur de l'inventaire est inférieure à 10, l'élément est pris, dans le cas contraire, le message « Inventory is full » est envoyé au joueur
  - ajout de la méthode throw(self, elem) permettant de détruire un élément de l' inventaire. Affiche un message qui indique quel élément le joueur a décidé de jeter
- Game( ) :
  - Dictionnaire : association de la touche « t » pour que le joueur puisse effectuer l'action

• **G.VII : (1) Diagonales :** Les déplacements du héros et des monstres peuvent être faits en diagonale.

- Coord( ) :
  - direction(self, other) : Modification de la méthode direction pour permettre un mouvement en diagonale
- Map() : ajout des quatre nouvelles directions avec touches associées dans le dictionnaire
- Game( ) :
  - dictionnaire : association de touches du clavier à chaque mouvement en diagonale

2. Actions :

• **A.II : (1) Repos :** Une fois par carte, le héros peut regagner 5 HP, mais les monstres effectuent 10 déplacements. #problème de déplacement des monstres (les monstres ne se déplacent pas)

- Game( ) :
  - Ajout de la fonction repos(self)
  - Dans init : self.alreadysleep=False. Quand le repos est demandé, elle prend la valeur True et on ne peut plus redemander un repos sur cet étage
  - Dictionnaire : association de la touche « r » pour que le joueur puisse effectuer l'action

3. Objet:

• **O.I : (1) Nourriture :** Le héros à un niveau de satiété (20) qui descend toutes les 3 actions. Si le héros tombe à 0 en satiété, il perd un hp à chaque action, jusqu'à ce qu'il utilise une nourriture, il revient alors au niveau initial de satiété.

- Hero(Creature)

- nourriture(self): méthode qui indique que toutes les 3 actions, on perd 1 point de satiété

- handler( ) :
  - nourish(creature): gère l'utilisation du fruit
- Game( ) : ajout des différentes nourritures dans le dictionnaire de équipements (pineapple, apple, kiwi)

• **O.II : (1) Armes :** Le héros peut équiper une arme (action use) parmi plusieurs existantes et cela ajoute des points de force à sa force initiale.

- Armes(Equipment): nouvelle classe pour les armes qui contient un dictionnaire et une méthode use(self, creature) qui permet d'équiper l'arme.

• **O.IV : (1) Armures :** Certains équipements, une fois portés (action use) permettent au héros de réduire des dégâts qu'il subit.

- Armors(Equipment): nouvelle classe pour les armures qui contient la méthode use qui permet au héros de s'équiper de l'armure afin de limiter les dommages des monstres
- Ajout d'un dictionnaire qui liste les différentes armures possibles et leurs caractéristiques
- Hero(Creature) :
  - ajout de l'argument armure, ayant par défaut la valeur 0
- handler : ajout de la méthode useStuff qui permet d'utiliser l'élément sélectionné

• **O.V : (1) Amulettes :** Le héros peut porter une amulette parmi plusieurs existantes qui lui donne un bonus (bonus de gain d'xp, augmentation de la force, régénération toutes les X actions de hp, ...)

- handler(creature) :
  - amuletGain(creature,xp,strength): fonction pour augmenter les xp et la force du héros lorsqu'il a une amulette
- Game( ) :
  - Dictionnaire : ajout de deux sortes d'amulettes (topaze, rubis)

4. Salles :

• **S.II : (1) Pièges :** La map contient des pièges placés de façon aléatoire, leur apparence ressemble aux cases de sol '.' . Quand le héros marche sur une case piégée, il perd 2 points de vie.

- Trap(Creature) : nouvelle classe pour les pièges avec une méthode meet pour enlever au héros 2 points de vie et pour afficher un message qui indique au joueur qu'il est passé sur un piège qui lui a infligé des dommages
- Room( ) :
  - decorate(self, map) : création d'une variable r qui prend un nom aléatoire entre 0 et 4 puis on place r pièges de façon aléatoire sur la map

• **S.III : (2) Marchand** : Une salle qui contient en son centre un marchand auquel le héros peut acheter des équipements en échange de l'or.

- Merchant(Element) : nouvelle classe pour les pièges avec une méthode merchantPrices(self) qui fixe les prix du marchand et une méthode meet qui affiche ce que le marchand propose de vendre avec les prix pour que le joueur puisse décider ce qu'il veut acheter
- Equipement(Element) : ajout du prix pour les équipements

• **S.IV : (2) Trésor :** Une salle qui contient un coffre au trésor. La clé a été donnée à un monstre aléatoire sur la carte et le héros peut la récupérer s'il tue le monstre.

- Treasure(Element) : nouvelle classe pour le trésor avec une meet qui permet d'ouvrir le coffre lorsque le joueur est à coté est possède la clé. Renvoie un message pour informer le joueur de l'objet qu'il a obtenu.
- Map( ) :
  - putTreasure(self) : méthode qui permet de placer le trésor sur une case vide de la map
  - giveMonsterTkey(self): permet de choisir aléatoirement un monstre qui détient la clé et d'envoyer un message pour dire au joueur quel est ce monstre.
- Creature(Element) :
  - meet(self, other) : si la créature a la clé, un message est affiché

5. Monstres :

• **M.I : (1) Poison :** Des monstres peuvent empoissonner le joueur, il perd un hp chaque X action, jusqu'à ce qu'il se soigne.

- Hero(Creature) :
  - Initialisation de poisoned à False
- Creature(Element) :
  - Ajout de l'argument d'instance poisonning initialisé à False
  - meet(self,hero) : si le monstre rencontré est empoisonné, il empoisonne le héros et l'argument poisoned de héros prend la valeur True
- Game( ) :
  - Dictionnaire ajout de deux sortes de monstres empoisonnés : des araignées et des scorpions : Creature("Scorpio","D", 4, fast=True,poisoning=True), Creature("Spider", "X", 2, fast=True,poisoning=True)
  - play(self) : Si le héros est empoisonné, il perd 1 point de vie à chaque tour de jeu
- heal(creature) : permet d'utiliser une potion pour soigner le poison

• **M.II : (1) Rapides :** Des monstres font 2 actions à chaque tour au lieu d'une.

- Creature(Element) :
  - Ajout d'un attribut d'instance qui vaut par défaut False
- Hero(Creature) :
  - Ajout d'un paramètre pour sélectionner les monstres les plus rapides
- Map( ) :

- moveAllMonsters(self): ajout d'une condition qui indique que si le monstre fait partie des monstres rapides, il effectue 2 déplacements par tour

• **M.IV : (1) Invisibles :** On ajoute un type de monstre, les fantômes, qui sont invisibles ('.') jusqu'à ce qu'ils frappent le héros ou que le héros les frappe.

- Game( ) :
  - Dictionnaire ajout d'un fantôme : Creature("Spirit",8,'.', xp=2)
- Creature(Element) :
  - meet(self) : si le nom de la creature rencontrée est Spirit, le '.' Est remplacé par 'S' et le message « An invisible creature is here ! » est affiché au joueur

TOTAL POINTS : 16