Rapport - Dungeon Master

CUVELIER Mathieu **LAMOUR Loïc** **LUCENAY Léonard**

I) Interface graphique

- `main_affichage.findTresor`: Recherche un trésor ouvert sur la carte du jeu en parcourant toutes les positions et renvoie True si un trésor ouvert est trouvé.
- `main_affichage.endWin` : Gère la fin du jeu lorsque le joueur atteint le niveau 25. Elle affiche une image de victoire, arrête le jeu et ferme la fenêtre après un délai de 5 secondes.
- `main_affichage.on_closing` : Gère la fermeture de la fenêtre `root2` dans Tkinter lorsque l'utilisateur tente de fermer la fenêtre.
- `main_affichage.messageFenetre`: Affiche une fenêtre pop-up avec un message donné et un titre optionnel. La fenêtre est centrée à l'écran et contient une étiquette (label) avec le texte du message.
- `main_affichage.is_digit_key` : Retourne True si la touche est une touche de chiffre ou de pavé numérique, et False sinon.
- `main_affichage.play_final`: Vérifie si une touche a été enfoncée et exécute l'action correspondante si elle est valide, puis effectue une série d'opérations liées au héros et aux monstres et à la vision, et enfin appelle `endWin`
- `main_affichage.displayInventory`: Parcourt les éléments de l'inventaire et tente d'afficher une image correspondante à chaque élément
- `main_affichage.textInput` : Permet d'obtenir une entrée utilisateur à partir d'une fenêtre popup
- `main_affichage.play_final` : Implémente la gestion des actions stockées dans `Game._actions`, le mouvement des monstres, et toutes les actions supplémentaires qui étaient gérés par `Game.play`.
- `main_affichage.displayInventory` : Affiche les images de tous les items présents dans l'inventaire du héros
- `main_affichage.update_all` : Permet d'actualiser toutes les informations situées sur le panneau droit (HP, XP, level…).

II) Points d'expérience

- `Hero.getLevel` : permet de renvoyer le dernier niveau d'XP passé par le héros
- `Creature.advanceLevel` : permet d'augmenter l'XP du héros quand le monstre qu'il rencontre meurt.

III) Inventaire limité

- `Hero.take` : méthode modifiée pour différencier le cas où de l'or est pris, et le cas où un autre équipement est pris
- `Equipment.meet` : modifiée pour n'ajouter l'équipement dans l'inventaire que s'il n'est pas déjà plein

IV) Déplacements intelligents

- `Noeud.manhattanDistance` : Calcule une distance de Manhattan entre deux noeuds (distance entre deux points en utilisant un quadrillage)
- `Noeud.shortestPath` : Cherche le plus court chemin entre deux noeuds
- `Noeud.voisins` : Calcule les voisins d'un noeud qui ne sont pas des murs
- `Noeud.reconstructPath` : Permet de reconstruire le chemin après qu'il ait été calculé
- `Coord.direction` : utilise `Noeud.shortestPath` pour calculer la direction que doivent prendre les monstres pour rejoindre le héros

V) Nuage de visibilité et Nuage de visibilité+

- `Map.rangeElement` : Calcule les coordonnées à une certaine distance de la créature passée en paramètre
- `Map.setVisible` : Génère la carte avec le nuage de visibilité

VI) Magie

- `Hero.becomeInvisible` : Rend invisible le héros après avoir utilisé une potion d'invisibilité

VII) Solidité

- `specialActions.equip` : Prend les équipements sélectionnés, leur enlève un point de durabilité s'ils en ont encore, et le supprime sinon

VIII) Boutique

- `Shop.checkItem` : Vérifie si un item se trouve ou non dans la boutique
- `Shop.addQuantity` : Ajoute un ou plusieurs instances d'un équipement déjà présent dans la boutique
- `Shop.isEmpty` : Vérifie si la boutique est vide
- `Shop.description` : Affiche une description des objets vendus
- `Shop.addItem` : Ajoute un item dans la boutique qui n'y était pas
- `Shop.getElementByNumber` : Permet de récupérer l'item dans la boutique en ayant son rang
- `Shop.removeItem` : Enlève un équipement de la boutique
- `Shop.getElem` : Permet de retrouver un équipement
- `Shop.meet` : est appelé quand le joueur rencontre la boutique

IX) Trésor

- `Chest.meet` : Vérifie si le coffre a déjà été ouvert, si le héros a suffisamment d'espace dans son inventaire, puis recherche une clé dans l'inventaire du héros, et si elle est trouvée, est retirée de l'inventaire, et le héros reçoit des récompenses (or et objets) du coffre.

X) Poison

- `Hero.poison` : Marque le héros comme empoisonné en attribuant la valeur `True` à l'attribut `_poisoned` du héros.
- `Hero.checkPoison`: Vérifie si le héros est empoisonné, et si tel est le cas, enlève 1 point de vie du héros.
- `Hero.recover` : Réinitialise l'attribut `_poisoned` à False, indiquant que le héros est maintenant guéri de l'empoisonnement.

XI) Compétences

- `specialActions.fireballThrow` : Lance une boule de feu dans une direction spécifiée, infligeant des dégâts aux créatures situées sur son trajet.
- `specialActions.supervision` : Permet d'acquérir une super-vision pendant 10 tours, augmentant ainsi sa portée de vision.
- `specialActions.tornado` : Affaiblit les créatures voisines du héros en réduisant leurs points de vie de moitié
- `Hero.useSkills` : Permet au personnage d'utiliser ses compétences, en permettant de choisir une compétence parmi celles disponibles.
- `Hero.unlockSkills` : Permet de débloquer les compétences spéciales du héros à certains niveaux.

XII) Rapides - Esquive

- `Hero.meet` : Génère des valeurs aléatoires, et en fonction de leur résultat esquive le coup ou attaque le héros deux fois