Compte rendu projet

Programmation Orientée Objet

Vahan KOMARYAN – Julien PANNIER – Poe-Iti PEROLINI

Elément Gameplay.I. *Interface Graphique :*

Class Interface :

- *left_interface* : permet de dessiner l'inventaire et les caractéristiques du joueur.

- *update* : met à jour l'interface.

Classe Icon :

- __init__(self, image, pos) **:** créé une icône avec une position et une image.

Partie « Start Menu » :

On affiche l'image de l'écran de démarrage et le bouton « start ».

Elément Gameplay.II. *Point d'expérience :*

Player.__init__ (self,…, xp, xpmax, …) : on ajoute au joueur les points d'expérience et l'expérience maximale qui correspond à l'expérience à atteindre pour passer un niveau.

Creature.__init__ (self,…, xp_gived, …) : on ajoute la variable xp_gived aux créatures qui correspond à l'expérience donnée lorsque la créature est éliminée.

Creature.damage(self, amount) : on augmente l'expérience du joueur lorsque la créature est éliminée. Si celle-ci dépasse xpmax, le joueur monte de niveau.

Elément Gameplay.III. *Inventaire limité :*

Player.__init__ (self, …): on créé les listes inventory, weapon, armour, amulet.

Player.take_equipment(self, equipment) : on permet au joueur de prendre les équipements qu'il trouve sur la carte. Il ne peut pas ramasser plus d'une arme (mêlée ou distance) pas plus de trois potions et seulement une amulette et une armure.

Elément Gameplay.III. *Déplacements intelligents :*

Creature.get_coord(self) : récupère les coordonnées du centre des monstres dans le système de coordonnées de la carte.

Creature.path_to_player(self) : renvoie le chemin le plus court pour aller sur la position du joueur.

Creature.distance_to_player(self) : renvoie la distance entre le monstre et le joueur.

Creature.direction_to_creature(self, creature) : renvoie la direction du vecteur entre la créature et une autre créature.

Creature.direction_to_wall(self, wall): renvoie la direction du vecteur entre la créature et un mur.

Creature.path_direction(self) : renvoie la direction à suivre pour atteindre le prochain point du chemin vers le joueur.

Creature.seth_path(self) : créé le chemin vers le joueur et les rectangles qui permettront de suivre l'avancée du monstre.

Creature.check_collide(self) : test si le monstre a rencontré un rectangle du chemin.

Elément Gameplay.VII. *Diagonales :*

Player.get_input(self) : le joueur peut appuyer sur deux touches de déplacement en même temps pour se déplacer en diagonale.

Creature : les créatures se déplacent en diagonale le long de la trajectoire de path finding.

Elément Actions.I. *Jet :*

Class Projectiles :

- Constructeur : on définit le type du projectile, son lanceur, sa vitesse, les dégâts qu'il fait, on initialise les textures en fonction de son type.
- Méthodes :
  - *rotate* : permet de faire tourner les projectiles sur eux même pour créer un effet de rotation.
  - *remove* : supprime le projectile de la carte.
  - *move* : on fait déplacer et tourner le projectile. De plus on va regarder si pendant le mouvement du projectile le projectile rentre en contact avec quelque chose. S'il rencontre quelque chose il est supprimé et si c'est une entité celle-ci se prend les dégâts du projectile.
  - *update* : met à jour le projectile.

Player.launch_projectile(self, lanceur, type, rotation) : permet au joueur de lancer un projectile (magie ou équipement).

Player.get_input(self) : on définit les touches permettant de lancer des projectiles :

- Espace : lance une boule de feu et utilise de la magie.
- V : lance une arme si possible.

Elément Actions.II. *Repos :*

Player.update(self) : le joueur regagne un point de vie et un point de magie toutes les dix secondes.

Elément Actions.III. *Magie ou Magie+ :*

Player.__init__ (self,…, pa_magie, pa_magie_max, …) : on ajoute au joueur les points de magie et les points de magie maximum. Ceux-ci sont utilisés pour lancer des sorts.

Creature.damage(self, amount) : on augmente les points de magie maximum du joueur lorsque la créature est éliminée.

Player.get_input(self) : on définit les touches permettant de lancer des sorts :

- Espace : lance une boule de feu et utilise de la magie.

- B : téléporte le joueur dans une autre pièce (magie)

Elément Actions.II. *Armes :*

Equipment.__init__(self, …, durability, strenght, …) : on peut créer une arme qui apporte de la force et qui possède une durabilité.

Player.take_equipment : permet au joueur de ramasser des équipements au sol. Le joueur ne peut porter qu'une seule arme en même temps. S'il passe sur une arme et que son inventaire est vide, elle est automatiquement ajoutée à l'inventaire et ses bienfaits sont attribués au joueur.

Player.get_input(self) : on définit la touche permettant de supprimer l'arme que l'on ne veut pas :

- W : supprime l'arme du joueur s'il en a une.

Elément Actions.III. *Armes de jet :*

Class Projectiles : (cf. Elément Actions.I. Jet)

Player.get_input(self) : on définit les touches permettant de lancer des armes :

- V : lance une arme si possible, si l'arme est un shuriken ou une épée, elle disparait de l'inventaire, si c'est un arc le joueur peut tirer trois flèches avant qu'elle disparaisse.

Elément Actions.IV. *Armures :*

Equipment.__init__(self, …, life_provided, …) : on peut créer une armure qui apporte de la vie supplémentaire au joueur, augmentant ses chances de survie.

Player.get_input(self) : on définit la touche permettant de supprimer l'armure que l'on ne veut pas :

- X : supprime l'arme du joueur s'il en a une.

Elément Actions.V. *Amulettes :*

Equipment.__init__(self, …) : on peut créer une amulette qui augmente le nombre de point de magie du joueur, lui permettant de faire plus de sorts.

Player.get_input(self) : on définit la touche permettant de supprimer l'armure que l'on ne veut pas :

- C : supprime l'amulette du joueur s'il en a une.

Elément Actions.VI. *Solidité :*

Equipment.__init__(self, …, durability, …) : on peut créer des équipements possédant une durabilité.

Player.update(self) : diminue la durabilité des équipements quand le joueur attaque une créature ou se fait attaquer. Si la durabilité de l'équipement est 0, il est supprimé de l'inventaire.

Player.get_input(self) : si le joueur tire une flèche avec un arc la durabilité de l'arc diminue et après trois flèches il disparait.

Elément Salles.II. *Pièges :*

Map.spawn_random_creature(self) : permet de faire apparaitre n'importe où sur la carte une créature « piège » qui ne peut pas bouger. Celle-ci à la texture d'un sol mais avec une tête de mort transparente donc difficile à voir si on ne se concentre pas.

Player.move(self) : gère la collision entre le joueur et un piège. Si c'est le cas, le joueur ne peut plus se déplacer pendant trois secondes et prend des dégâts.

Elément Monstres.II. *Rapides :*

Creature.__init__(self, …, speed) : On ajoute la variable vitesse aux créatures.

Map.spawn_random_creature(self) : permet de créer des créatures avec des vitesses différentes. Dans notre cas les chevaliers noirs sont rapides et ont une vitesse deux fois supérieure aux diables.

Elément Monstres.III. Archers *:*

Map.spawn_random_creature(self) : permet de créer des créatures qui peuvent attaquer à distance. Dans notre cas ce sont les diables qui tirent des boules de feu.

Creature.launch_projectile(self) : vérifie si le joueur est dans la ligne de tir de la créature, s'il se trouve à moins de cinq cases et s'il n'y a pas de mur entre elle et lui, dans ce cas, elle tire une boule de feu dans sa direction.

Element Animation.?. ?:

Player.animate(self) : permet d'animer le joueur s'il est en cours de déplacement selon des textures que l'on a chargé dans le constructeur. L'animation s'adapte à la direction de joueur.

Creature.texture_update(self) : permet de mettre à jour les textures des créatures selon leurs directions.

Projectiles.move(self) : permet d'appliquer une texture différente aux projectiles suivant leurs directions.