Compte rendu du Projet :

Galli Evan

Baillot Téo

Lécard Maxence

Amrane Neil

Introduction :

Pour réaliser ce projet, nous avons dû apprendre à utiliser Github afin de mettre en communs les avancées de chacun ainsi que de lister la répartition du travail et l’avancement de chaque tâche.

Lien de la repository : [Other-Project/PeiP1-Rogue](https://github.com/Other-Project/PeiP1-Rogue)

Ce projet utilise pygame pour son interface

Liste des éléments qui ont été ajoutés :

Gameplay :

I-Interface graphique (fichier GUI.py) :

**-drawImage (screen, path, x, y, w, h) :** fonction qui permet de dessiner une image de sorte à ce qu’elle « fit » un rectangle x,y,w,h

**-drawText(screen, text, x, y, w, h, size=14, color=(255, 255, 255), fontName= ‘comicsansms’) :** fonction qui permet de dessiner un texte.

-Dans la classe Button (permet de créer des boutons et d’interagir avec sur l’interface) :

**-update(self, events) :** méthode qui permet de détecter les interactions de la souris.

**-drawImage(self, surface : pygame.Surface, imagePath, event=None) :** méthode qui permet de dessiner un bouton en utilisant une image.

**-drawText(self, surface : pygame.Surface, text, event=None) :** méthode qui permet de dessiner un bouton en utilisant un texte.

-Dans la classe GUI (permet de gérer tout le fonctionnement de l’interface) :

-**updateSreenSize(self, w=0, h=0) :** méthode qui permet de gérer le redimensionnement de la fenêtre du jeu.

**-main(self) :** méthode qui permet de gérer la boucle principale.

**-getTileSurface(self, e) :** méthode qui retourne la surface d’un élément de la map.

**-getTilePos(self, x, y, e) :** méthode qui retourne la position sur la map d’un élément.

**-gameMap(self, event) :** méthode qui permet de dessiner la Map.

**-getBarColor(value : float, maxValue : float) :** méthode qui renvoie la couleur de la barre de vie/solidité en fonction de la valeur de celle-ci

**-startScreen(self) :** méthode qui permet de dessiner l’écran de démarrage.

**-drawInfoBox(self, x, y, e, padding=5) :** méthode qui permet de dessiner des boites d’information.

**-drawItem(self, elem, x, y, event, action=lambda elem, hero : elem.deEquip(hero), rightAction=lambda elem, hero : elem.deEquip(hero, True), size=None) :** méthode qui permet de dessiner une box avec un item dedans.

**-drawPotion(self, x, y, i, event) :** méthode qui permet de dessiner le bouton d’une potion.

**-sidePanel(self, event) :** méthode qui permet de dessiner la partie à droite de l’écran (partie avec les informations sur le joueur)

**-drawControl(self, x :int, y : int, w :int, h :int, text : str, image : str, scale :float) :** méthode qui permet de dessiner les informations concernant les contrôles.

**-drawEquipment(self, x, y, w, h, event) :** méthode qui permet de dessiner les équipements

**-drawBarImage(self, x, y, valueMax, image, width, height=None, nbCol=5, padding=5, sizeImage=None) :** méthode qui permet de dessiner des barres d’images horizontales avec le même espacement.

**-drawBar(self, x, y, valueMax, drawFct, width, height=None, nbCol=5, padding=5, sizeImage=None)** : méthode qui permet de calculer une barre horizontale et d’appeler drawFct pour chaque élément.

**-endScreen(self) :** méthode qui permet de dessiner l’écran de fin.

**-getEvents(self, additionalEvents=None)** : méthode qui permet de récupérer les évènements qui ont eu lieu depuis la dernière boucle, les tris, et fait les actions associées.

**-chestPopup(self,** **chest: Chest, sell) :** méthode qui permet de gérer les interactions avec le coffre et avec le marchand

**-takeItemFromChest(self, chest, element) :** méthode qui est appelée quand on clique sur un objet du coffre ou du marchand.

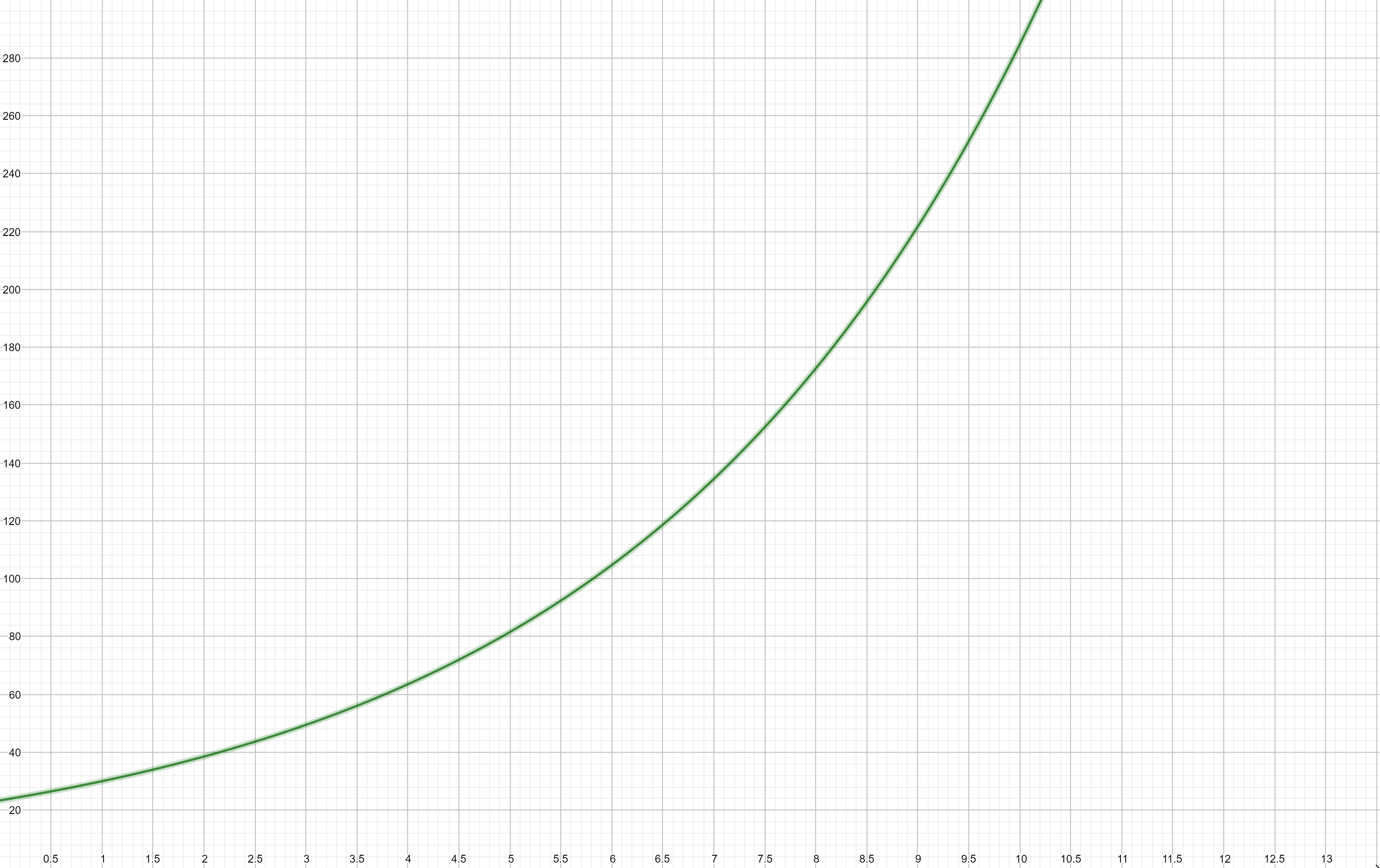
II-Point d’expérience (XP) (fichier Hero.py) :

-Dans la classe Hero :

**-attack(self, attacked, speAttack=None) :** méthode dans laquelle on ajoute l’XP au héro quand il tue un monstre.

-**experience(self) :** méthode qui permet d’appliquer les effets de changement de niveau du héros (augmenter la limite maximum de vie et de mana)

**-lvlSup(self) :** méthode qui retourne l’XP qu’il faut avoir pour passer au niveau supérieur. (la courbe d’xp est exponentielle)



III-Inventaire limité (fichier Hero.py) :

-Dans la classe Hero :

-**addInventory(self, item) :** méthode qui permet d’ajouter l’élément à l’inventaire si celui-ci n’est pas plein

IV-Déplacements intelligents (fichier Astar.py) :

- Dans la classe State :

Contient les différentes constantes d’état utilisées par la classe Node

-Dans la classe Node :

**-h(self, dest)** : méthode qui retourne la distance en ligne droite entre ce nœud et le nœud final.

**-g(self) :** méthode qui retourne la longueur du chemin depuis le nœud de départ jusqu'à ce nœud.

**-f(self, dest) :** méthode qui retourne la distance totale estimée.

-**getPath(self) :** méthode qui retourne une liste de coordonnées menant à ce nœud.

-**getAdj(self) :** méthode qui retourne les coordonnées des cellules adjacentes atteignables.

-Dans la classe Astar :

*Comme son nom l’indique, cette classe implémente* [*l’algorithme A\**](https://fr.wikipedia.org/wiki/Algorithme_A*)

*Des tests de fonctionnement peuvent être retrouvé dans test/astar.py*

**-getMatRepr(self, path=None) :** méthode qui renvoie une représentation de la matrice actuelle avec (éventuellement) un ensemble de nœuds mis en évidence.

**-getAdjacentWalkableNodes(self, fromNode : Node) :** méthode qui renvoie les nœuds accessibles à partir du nœud actuel.

**-search(self, currentNode : Node, endNode) :** méthode qui permet de trouver un chemin entre le nœud actuel et le point de destination.

**-findPath(self, destination) :** méthode qui renvoie une liste de coordonnées menant au point de destination.

V-Nuage de visibilité (fichier GUI.py) :

-Dans la classe GUI :

**-gameMap(self, event) :** méthode qui permet au joueur de ne pas voire toute la carte, mais seulement les tuiles déjà visitée si la difficulté sélectionnée est medium.

VI-Nuage de visibilité+ (fichier GUI.py) :

-Dans la classe GUI :

**-gameMap(self, event) :** méthode qui permet au joueur de ne voir les monstres et équipements qu’à proximité de l’endroit où il se trouve si la difficulté sélectionnée est hard.

Actions :

I-Jet (fichiers Hero.py et Projectile.py) :

-Dans la classe Hero :

**-shootProjectile(self, gui, monster, onCollide=None) :** méthode qui permet de déclencher l’animation du tir et d’effectuer des dégâts au monstre visé.

-Dans la classe Projectile :

**-draw(self) :** méthode qui permet de dessiner l’animation du jet et inflige les dégâts au monstre.

II-Repos (fichier Map.py) :

-Dans la classe Map :

**-rest(self, hero) :** méthode qui permet de récupérer 5 points de vie en passant 10 tours.

III-Magie (fichiers GUI.py et config.py) :

-Dans la classe GUI :

**-drawPotion(self, x, y, i, event) :** méthode qui permet d’activer l’effet de la potion si on clique dessus.

-Dans le fichier config.py :

**-heal(hero : Hero, hpGain=3) :** fonction qui augmente les points de vie du héro de 3.

**-manaPotion(hero : Hero, manaGain=1) :** fonction qui permet de donner du mana au héro (le héro doit dépenser plus ou moins de mana en fonction du sort pour pouvoir l’utiliser)

**-teleport(hero : Hero) :** fonction qui permet de téléporter le héro a un endroit aléatoire de la map.

**-invisible(hero : Hero) :** fonction qui permet de rendre le héro invisible pendant 10 tours tant qu’il n’attaque pas de monstre.

IV-Magie+ (fichiers GUI.py et config.py) :

-Dans la classe GUI :

-**drawPotion(self, x, y, i, event)** : méthode qui permet d’activer l’effet de la potion si on clique dessus.

-Dans le fichier config.py :

**-zap(hero : Hero) :** fonction qui inflige 3 de dégâts à tous les monstres dans sa portée.

**-fireball(hero : Hero) :** fonction qui tue un monstre aléatoire dans la portée du héro.

**-superStrength(hero : Hero) :** fonction qui donne un boost d’attaque au héro.

Objets :

I-Nourriture (fichiers config.py, GUI.py et Game.py) :

-Dans la classe GUI :

**-sidePanel(self, event) :** méthode dans laquelle on dessine la barre de vie en utilisant la méthode drawBarImage.

Dans la classe Game :

**-newTurn(self) :** méthode qui permet de faire descendre la satiété du héro à chaque tour et lui enlève des points de vie si ça arrive à 0.

-Dans le fichier config.py :

**-eat(hero : Hero, satietyGain=2) :** fonction qui permet d’augmenter la barre de nourriture du héro quand il mange.

II-Armes (fichier Weapon.py) :

-Dans la classe Weapon :

**-equip(self, hero : Hero) :** méthode qui gère l’équipement d’une arme qui rajoute de la force au héro.

**-deEquipe(self, hero, remove=False) :** méthode qui permet de déséquiper une arme (enlevant donc les dégâts que l’arme rajoutait).

III-Armes de jet (fichier Weapon.py) :

-Dans la classe Weapon :

**-rangedAttack(self) :** méthode qui gère les attaques à distance.

IV-Armures (fichier Armor.py) :

-Dans la classe Armor :

**-equip(self, hero :Hero) :** méthode qui permet d’équiper une armure lui ajoutant de la résistance et lui permettant de subir moins de dégâts de la part des monstres.

**-deEquip(self, hero : Hero, remove=False) :** méthode qui permet de déséquiper l’armure (lui enlevant la résistance associée à l’armure).

V-Amulettes (fichier Amulet.py) :

-Dans la classe Amulet :

**-equip(self, hero : Hero) :** méthode qui permet d’équiper une amulette et d’activer l’effet associé.

**-deEquip(self, hero, remove=False) :** méthode qui permet de déséquiper une amulette et désactive l’effet associé.

VI-Solidité (fichiers Monster.py et Hero.py) :

-Dans la classe Monster :

**-attack(self, attacked : Hero, damage=None) :** méthode qui fait baisser la solidité de l’armure quand un monstre attaque le héro et détruit l’armure quand la solidité est nulle.

-Dans la classe Hero :

**-attack(self, attacked, speAttack=None) :** méthode qui fait baisser la solidité de l’arme quand le héro attaque et la détruit quand la solidité est nulle.

Salles :

I-Gestion des salles (fichier Map.py) :

Les différents types de salle héritent de Room. Les salles sont choisies aléatoirement dans la fonction *generateRooms* en respectant une pondération.

II-Pièges (fichiers RoomTrap.py, Map.py et GUI.py) :

-Dans la classe RoomTrap :

**-decorate(self, floor) :** méthode qui permet de générer la position des pièges

-Dans la classe Map :

**-onTrap(self) :** méthode qui permet de vérifier si le joueur est dans un piège et si oui, lui inflige des dégâts

-Dans la classe GUI :

**-heroTrapped(self, coord: Coord, image="assets/hero/heroTrapped.png") :** remplace momentanément l’image du héro

III-Boutique (fichier Merchant.py) :

-Dans la classe Merchant :

**-meet(self, hero) :** méthode qui permet de gérer la rencontre entre le héros et le marchand. (affiche le popup)

**-takeItem(self, hero, element) :** méthode qui permet de prendre l’élément si le héro a suffisamment de pièces et que son inventaire n’est pas plein.

-Dans la classe RoomShop :

**-decorate(self, floor) :** méthode qui permet de générer la salle du marchand.

IV-Trésor (fichier Chest.py) :

-Dans la classe Chest :

**-meet(self, hero) :** méthode qui gère la rencontre entre le héro et le coffre. (affiche le popup)

**-takeItem(self, hero, element) :** méthode qui permet de prendre l’élément si le héro a suffisamment de place dans son inventaire.

-Dans la classe RoomChest :

**-decorate(self, floor) :** méthode qui permet de générer la salle du marchand.

Monstres :

I-Poison (fichier Spider.py et Game.py) :

-Dans la classe Spider :

**-meet(self, attacker) :** méthode qui enlève de la vie au monstre quand il est attaqué.

**-attack(self, attacked, damage=None) :** méthode qui gère l’attaque du monstre et l’application du poison.

-Dans la classe Game(object) :

**-newTurn(self) :** méthode qui permet d’enlever de la vie à chaque tour tant que le héro est empoisonné.

II-Rapides (fichier Monster.py) :

-Dans la classe Monster :

**-doAction(self, floor : Map) :** méthode qui permet de déplacer les monstres en fonction de leur vitesse de déplacement

III-Archers (fichier Monster.py)

-Dans la classe Monster :

**-doAction(self, floor : Map) :** méthode qui permet de gérer les déplacements et attaques à distance des monstres.

IV-Invisibles (fichier Ghost.py):

-Dans la classe Ghost :

**-meet(self, attacker) :** méthode qui permet de rendre le fantôme invisible tant qu’il n’a pas rencontré le héro.

**-attack(self, attacked, damage=None) :** méthode qui permet de rendre le fantôme visible à partir du moment où il attaque le héro.

Bonus :

I-Choix de la difficulté au démarrage (fichier GUI.py) :

-Dans la classe GUI :

**-gameMap(self, event) :** méthode qui demande au démarrage du jeu au joueur de choisir la difficulté : easy= pas de nuage de visibilité, medium= nuage de visibilité activé, hard= nuage de visibilité+ activé.

II-Salle de boss (fichier RoomBoss.py):

-Dans la classe RoomBoss:

**-decorate(self, floor) :** méthode qui permet de créer une salle spéciale contenant un boss

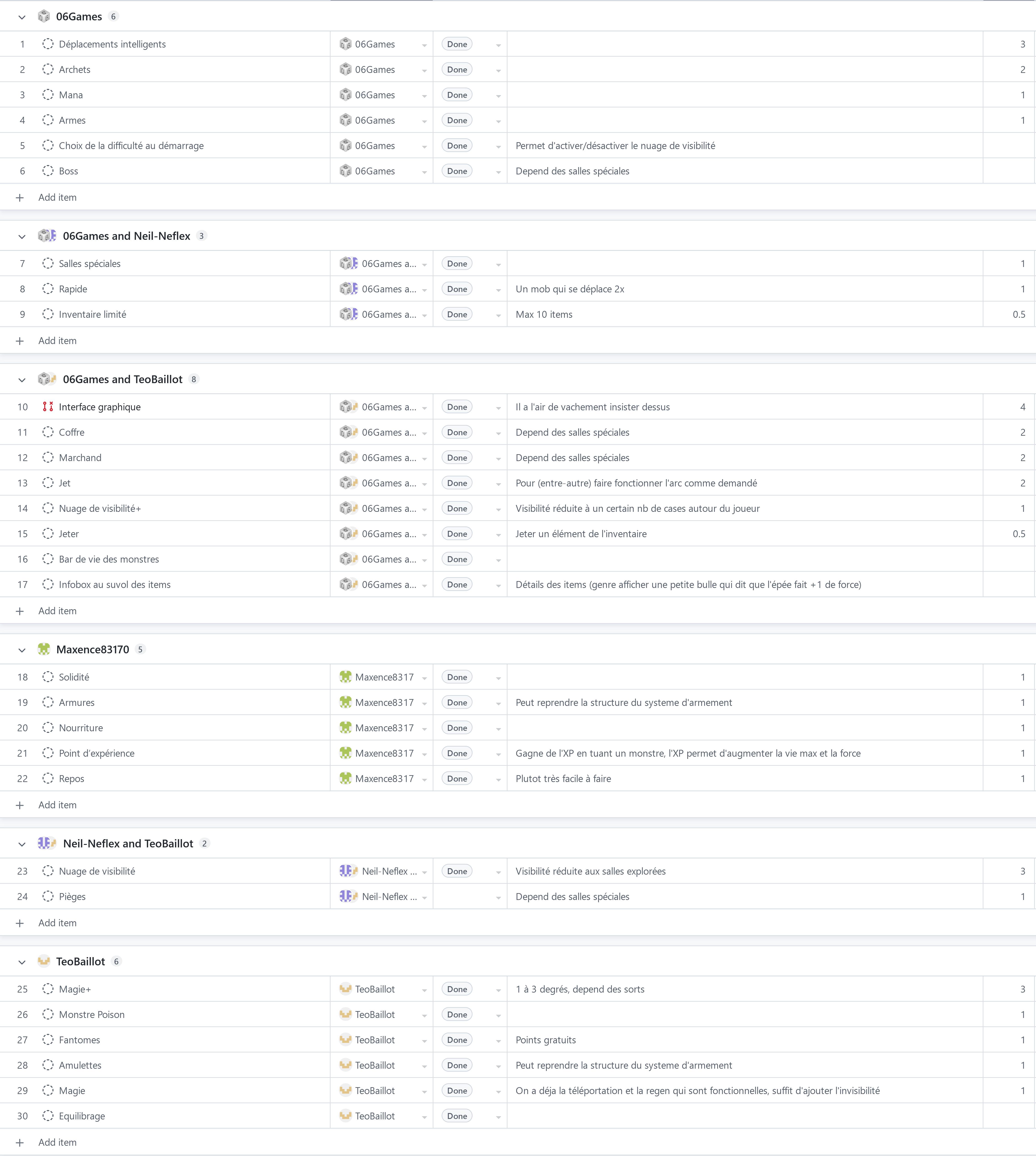
Répartition du travail :

06Games : Galli Evan

TeoBaillot : Baillot Téo

Maxence83170 : Lécard Maxence

Neil-Neflex : Amrane Neil



Ce que le projet nous a apporté :

Pour conclure, ce projet nous a permis d’énormément progressé et nous a poussé à apprendre par nous-même ce qui est vraiment une bonne chose car dans notre métier futur nous devrons constamment le faire par nous-même. On s’est vu progressé tout au long de l’avancement du projet ce qui est vraiment satisfaisant. On a aussi appris à réaliser un projet en groupe, à se répartir le travail, à surmonter des difficultés, à organiser des réunions afin de mettre en commun les avancements et les idées, à choisir ce qu’on veut faire car on peut faire absolument tout ce qu’on veut, à se redistribuer les tâches pour atteindre les objectifs quand quelqu’un a pris du retard.