Projet Réalisés par :

TATA TATA

ELHADJ ABASSY DIALLO

Documentation du projet UBGarden :

UBGarden est un jeu d'aventure dans le lequel on incarne un Jardinier de sauver un Herisson, tout en évitant les dangers comme les Guêpes et les frelons. Le jardinier doit ramasser toutes les carottes pour ouvrir les portes et de passer dans un Nouveau Monde, ramasser les Bonus comme les pommes pour gagner de l'anergie et utiliser les bombes insecticides pour se protéger.

Fonctionnalités Implémentées :

Partie 1 : Affichage et Déplacement

Affichage Complet de la carte avec tous les éléments de décors (Bonus, Fleurs , portes, Nids de Guêpes et Frelons, Carottes …).

Le déplacement du jardinier à la limite de la carte et l'empêcher de marcher sur certains décors comme les arbres, les fleures, les portes quand elles sont fermées en utilisant les interfaces fournies et le polymorphisme

Le Jardiner se déplace tant qu'il possède de l'énergie et s'il n'a plus la partie est terminée, et il peut récupérer un point d'énergie à chaque seconde s'il est immobile.

La condition de Victoire est de retrouvé le hérisson, quand le jardinier se trouve dans la même case que le hérisson, le jeu est gagné.

Affichage de l'énergie, le niveau de fatigue, les bombes dans une barre d'état en utilisant la classe **StatusBar**

La collecte des Bonus :

Pommes : Restaure l'énergie

Pommes Pourries : augmente le niveau de fatigue a un certain temps défini

Bombes insecticides : utilisés pour se défendre des insectes

Partie 2 : Gestion de Mondes

Le jeu peut être lancé par une carte par défaut ou directement téléchargé à partir d'un fichier.

Chaque niveau contient des décors avec des portes soit fermées ou ouvertes et pour ouvrir les portes le jardinier doit ramasser toutes les carottes pressentes et cela ouvrira directement les portes.

Pour la gestion des portes nous avons fait une seule Classe Porte héritant Décor contenant un booléen si la porte est ouverte ou fermée, ensuite nous avons créé son sprite **SpriteDoor** heritant de Sprite et avons implémenter cette classe et pour son affichage nous l'avons ajouté dans la classe **SpriteFactory ,** ainsi dans la méthode Pickup prenant en paramètre les carottes nous avons implémenter en disant au Jardinier de se souvenir à la porte emprunter pour le niveau suivant et de se positionner juste devant la porte, donc à son retour il sera devant cette porte . Pour passer d'un Niveau a autre nous avons utilisé la méthode checklevel de la classe **GameEngine** qui permet au jardinier de se promener entre les niveaux et pour éviter aux insectes de reparaître à un autre niveau

différent de le leur, nous avons utilisé l'interface **stream** dans la methode **update** de GameEngine pour les filtrer a chaque niveau.

Parties 3 : Génération des insectes (Guepes et Frelons)

Les insectes étant des personnages nous avons créé une classe abstraite Insecte héritant de GameObject et qui implémente les interfaces **Movable, Walkvisitor, Walkable**. Chaque insecte possédant chacun ses propriétés et des propriétés communes, nous avons factorisé le code dans la classe insecte en utilisant l'héritage et dans cette classe nous avons définies les propriétés communes soit le nombre de piqûres disponibles, les dommages de piqûres sur le Jardinier, sa direction, les vitesses de déplacement. Les insectes sont des personnages qui se déplacent alors nous avons définies ces méthodes se déplacer et peut se déplacer en les limitant sur la grille et de les empêcher de marcher sur les décors interdits, nous avons utilisé le **Timer** pour savoir s'il est en cours de mouvement ou s'il ne peut pas se déplacer et la méthode **Update** pour le déplacement aléatoire si le timer a expiré ensuite nous avons défini une méthode Abstraite **Sting** pour chaque sous classe Guêpes et Frelons

Dans les sous classes Guêpes et Frelons ils se déplacent aléatoirement et sont générées par leurs nids respectifs Nids de Guêpes et Frelons, les Guêpes sont générées toutes les 5 secondes et les frelons toutes les 10 secondes. Le coût de piqûres pour les Guêpes réduisent -20 et les frelons -30 de l'énergie du jardinier

Pour les Cas de Guêpes lorsqu'elles sont nées elles génèrent aussi une bombe placée aléatoirement et si elles touchent la bombe elles meurent donc dans la classe **NestWasp** nous avons implémenter ses fonctionnalités en créant une méthode **private spawnWasp** et nous avons appelé cette méthode dans la méthode **Update**, et pour les cas de frelons nous l'avons fait de même dans la classe **NestHorent** donc chaque frelon né cré deux bombes et les place aléatoirement et il faut deux bombes pour tuer un frelon, une seule bombe la blesse et nous avons implémenté si le frelon marche sur une bombe on divise son coût de piqûre par 2 comme il est blessé.

Dans les sous-classes de l'insecte chacune définissant la méthode **Sting()** de ses propres caractéristiques en contact avec le jardinier.

Pour ajouter les insectes nous avons créé une liste d'insectes dans la **World** pour les stocker, chaque insecte créer possède sa classe Sprite pour l'affichage et ensuite on les ajoute dans la classe **SpriteFactory** pour les affichages.

Pour les Cas des Bonus comme l'affichage des Bombes insecticides nous avons créé son sprite et l'avons ajouté dans la classe SpriteFactory ainsi nous avons tous ses sprites dans des listes différentes que nous avons définies dans la classe **GameEngine.**

Ainsi dans cette classe dans la méthode **initialise()** nous avons ajouté des sprites comme les bonus (bombes insecticides).

Détection des collisions entre le Jardiner et les Insectes :

Pour cette partie nous avons d'abord définies des méthodes dans la classe pour définir le nombre de bombe qui peut tuer un insecte et une méthode qui détermine si le jardinier peut tuer l'insecte pour éviter d'être piquer même en possédant des bombes, et dans les sous-classes nous avons implémenter ces méthodes a chacun définissant le nombre de bombe pour le tuer.

Donc la méthode **CheckCollision** nous a permit d'implémenter ses fonctionnalités pour chaque type d'insecte en collision avec le jardinier.

Points à retenir : Dans la classe **MapLevelDefaultStart** qui nous permet de lancer un jeu par défaut ne contient pas de niveau de suivant donc si la porte est ouverte est que le jardnier se trouve sur la porte, il pourra pas aller un niveau superieur donc il sera planté

Ensuite Le but ton Editor qui nous permet de créer une map n’est pas implémenter car ça me donne des erreurs lorsque j’ai voulu le faire