**Project Pacman : Rapport technique**

**Prefabs :**

* Player : Représente le joueur
* Enemy : Représente un ennemi

**Scripts :**

**-GameManager** : Ce projet est basé sur une fonction ClockUpdate()  qui est appelée dans le GameManager un certain nombre de fois par seconde. La fonction ClockUpdate() du GameManager appelle les ClockUpdates() des autres scripts, ce qui me permet d’avoir une fonction qui se lance en boucle et en même temps dans plusieurs scripts (GameManager, PlayerController, EnemyController).

Dans le Start() : le GameManager sauvergarde la scène actuelle dans le PlayerPrefs.

Il instancie les nœuds (Node) d’un graph (Graph) représentant les cases de la carte où les personnages peuvent marcher.

Il instancie 4 ennemis.

Dans la ClockUpdate() : Si un personnage se trouve sur un téléporteur, il le téléporte.

Si un ennemi a touché le joueur, il lance le game over.

Si le joueur est sur un Pacgum ou un Super Pacgum, il change l’apparence de cette case pour la faire passé d’un sol avec un Pacgum ou un Super Pacgum à un sol vide.

Si le pourcentage de Pacgum et Super Pacgum mangés est supérieur ou égal à 0, 0.2, 0.4, 0.6, il fait passer en état 2 le premier ennemi, puis le deuxième, puis le troisième, puis le quatrième.

**-PlayerController**:

Dans l’ Update() : Si le joueur appuie sur une touche de direction et que cette direction pointe vers un endroit où le joueur peut aller, le PlayerController l’enregistre dans un Vector2Int

Il fait avancer le joueur de sa position actuelle jusqu’à sa position voulue

Dans la ClockUpdate() : Si la direction enregistrée dans l’Update() pointe vers un endroit où le joueur peut aller, il enregistre cet endroit comme étant la position voulue.

Il tourne le joueur vers là où il va.

**-EnemyController** :

Dans l’ Update() : L’Enemy Controller fait avancer l’ennemi vers une position cible.

Dans le ClockUpdate() : Si l’ennemi est en état 1 : il fixe la position cible à celle de l’ennemi pour qu’il ne bouge pas .

Si l’ennemi est en état 2 : il fixe la position cible à la sortie du spawn des ennemis jusqu’à ce que l’ennemi y arrive. Puis fait passer l’ennemi en état 3.

Si l’ennemi est en état 3 : il fixe la position cible sur une des cases voisines de celle où se trouve l’ennemi, sauf celle d’où il vient.

**-MenuManager :** Lance un des 2 niveau ou montre les crédits en fonction du bouton sur lequel on appuie.

**-CreditsManager :** Retourne au menu si on appuie sur échap.

**- GameOverSreenManager :** Relance le niveau ou retourne au menu en fonction du bouton sur lequel on appuie.

**-Node :** C’est un objet représentant les cases de la map, qui contient une liste de ses Node voisin, des coordonnées et une variable qui nous informe sur ce qui est sur la case qu’il représente.

**-Graph :** C’est un objet qui à partir d’une tilemap, créé un graph de Node où chaque Node représente une tile de cette tilemap.

Il possède aussi les fonction HasNode(), qui permet de savoir s’il y a un Node avec certaines coordonnées dans un graph et GetNode, qui permet de récupérer un Node d’un graph à partir de ses coordonnées.

**Difficultés techniques :**

**Ne pas avoir de binôme**-> Aide généreuse de PH.

**Ne pas utiliser de collider** -> Pour ce TP j’ai voulu essayer de me passer de collider, pour se faire, j’ai d’abord vérifié le contenu des Node sur lesquelles venait d’arriver mon Player à chaque déplacement. Cela fonctionnait pour savoir si mon Player était au même endroit qu’un objet fixe comme un Pacgum ou un téléporteur. Mais pour les ennemis cela ne fonctionnait plus. Si le joueur et un ennemi étaient sur des Node voisines puis se croisaient, ils ne partageaient à aucun moment la même Node, l’ennemi ne tuait alors pas le joueur.

J’ai donc dû utiliser un système me permettant de synchroniser plusieurs évènements de plusieurs scripts différents.

Avec la ClockUpdate(), les ennemis et les joueurs commencent et terminent leurs mouvement d’une case à l’autre en même temps. Et je peux vérifier à ce moment précis, si la dernière Node d’un ennemi est égale à la Node actuelle du joueur et que la Node actuelle de l’ennemi est égale à la dernière Node du joueur, ce qui témoignerai d’un « croisement ennemi – joueur » où le joueur ne mourrait pas avec l’ancienne méthode. Au quel cas je pourrais lancer le game over.

**Fonctionnalités supplémentaires :**

-L’implémentation de la fonction ClockUpdate()

-Le mouvement du joueur lissé grâce à un Lerp()

-Deux niveaux qui fonctionnent avec le même code

-Tentative d’implémenter des Super Pacgum, permettant de manger les ennemi (abandon pour cause de manque de temps)

**Fonctionnalités obligatoires manquées :**

Absence de récapitulatif de fin de partie et IA des ennemis très faible. (manque de temps)