TP 1 : Nourrir les animaux (/70)

Figure 1: not representative of final product

Finalité

- 2 périodes de laboratoire sont prévues pour ce travail.
- Ce travail compte pour 10% de la session.
- Vous familiariser avec le développement de fonctionnalité à partir de requis et d'instructions

Prérequis

- Avoir réalisé avec succès le formatif 1 et 2.

À remettre

- La remise devra être faite en classe à la semaine 3 en présentant le résultat à l'enseignant

Notes importantes

- Se baser sur le package « **TP1 – Nourrir les animaux** » fourni avec le TP1
- Se placer en équipe de 1 ou 2 pour le TP

Description du jeu final attendu

- Le jeu doit être en vue de haut, de type « space invader », une légère perspective est permise
- Le personnage avance continuellement et peux se déplacer de droite à gauche
- Alors que le personnage avance, il rencontre des animaux affamés de sa ferme
- Le personnage peut lancer de la nourriture en appuyant sur Espace pour nourrir les animaux
- Si un animal qui a toujours faim est dépassé (ils descendent plus loin que le joueur) la partie est perdue et le gameplay s'arrête
- Des animations si vivantes que vous vous y croiriez réellement
- Des effets visuels et sonores à couper le souffle
- Un GameOver attristant

Figure 2 Apparence plus près du résultat attendu

Remarques Générales

- Tous les déplacements doivent être normalisés dans le temps (utiliser Time.deltatime)
- Les constantes utilisées doivent être sous forme de variables (pas « hard-coded »)
- Le code doit être commenté de façon adéquate

Comportement des différents archétypes :

Positionnement et Comportement de la caméra :

- La caméra doit être fixe et ne jamais bouger sauf pour des effets visuels
- C'est le reste du monde qui bouge pour donner l'illusion de déplacement à travers un script MoveDown.cs partagé pour le background et les animaux

Positionnement et Déplacement du joueur :

- Le personnage doit être un humain
- Un seul script PlayerController doit gérer le comportement du personnage
- Le joueur a une position verticale fixe en bas de l'écran
- Le joueur peu se déplacer horizontalement selon l'Input avec Translate()
- Son déplacement doit être indépendant du temps
- Toutes les valeurs doivent utiliser des variables au lieu de constantes
- La position doit être limitée pour qu'il demeure à l'écran

Positionnement et Comportement des animaux :

- Les animaux doivent être une prefab instanciés par un script AnimalSpawner.cs, qui s'occupe de spawn des animaux à interval aléatoire
- Lorsqu'instancié, l'apparence de l'animal doit être sélectionné aléatoirement parmi plusieurs (minimum 2) apparences différentes
- Le système d'animation doit correspondre à l'apparence de l'animal
- Les animaux sont instantiés en dehors de l'écran avec une position horizontale aléatoire
- Les animaux affamés défilent continuellement vers le bas à une vitesse fixe, sans impact sur leurs animations et se déplace lentement de droite à gauche
- Lorsque nourri, un animal se met à manger pendant un court moment et s'exprime de joie, puis s'enfui vers un des cotés de l'écran
- Si un animal défile et dépasse le joueur, la partie s'arrête car elle est perdue et les animaux arrêtent de défiler

Caractéristiques du projectile (nourriture) :

- Le projectile doit être une Prefab, dont on instancie des copies lorsque le joueur appuie sur Espace
- Le projectile vole vers l'avant de façon continue
- Si le projectile collisionne avec un animal, il doit appeler une fonction sur le script de l'animal qui l'averti qu'il est maintenant nourri et disparaitre
- Le projectile doit être détruit s'il sort de l'écran il qu'il n'as pas été mangé par un animal

Scripts Nécessaires et fonctions (/50):

PlayerController.cs (/10)

- Détecte les inputs et déplace le personnage de droite à gauche avec tranform.Translate()
- Donne l'illusion que le personnage avance tout le temps
- Limite le déplacement pour que le personnage demeure à l'écran
- Détecte la touche espace et spawn une pointe de pizza à sa position
- Gère les animations du personnage

MoveDown.cs (/6)

- Fait défiler l'objet vers le bas si on est en jeu (pas gameOver)
- Si l'objet est le background, reset la position du background pour créer l'illusion d'infinité
- Détruit l'objet s'il sort de l'écran par le bas

FoodController.cs (/10)

- Déplace la nourriture vers l'avant
- Détruit la nourriture si elle sort de l'écran
- Détecte la collision avec un animal, appelle la fonction Manger() sur l'animal, et détruit la nourriture.

AnimalController.cs (/10)

- Si affamé et en jeu (pas gameOver), l'animal se déplace de droite à gauche lentement mais demeure à l'écran
- Si nourri et en jeu, l'animal se déplace vers en dehors de l'écran par la gauche ou par la droite
- Fonction Manger() qui nourri l'animal et change son état
- Gère les animations de l'animal
- Si gameOver, l'animal doit agir d'une autre façon (au choix de l'étudiant)

AnimalSpawner.cs (/7)

- Attend un delay aléatoire (+/- 50% du délai de base)
- Spawn un animal aléatoire à une position aléatoire
- Bonus : à chaque 15s, le délai de base réduit de 5%, augmentant lentement la difficultée

GameOverTrigger.cs (/7)

- Si entre en collision avec un animal, active le gameOver

Animations (/10) :

Personnage

- Cours continuellement en jeu
- S'arrête (idle) si la partie est gameOver et joue une animation autre (au choix)

Animaux

- Animation de déplacement lorsque se déplace de droite à gauche
- Animation «Eat » lorsque nourrit pour 1s puis se remet à se déplacer
- Arrête de se déplacer lorsque gameOver et joue une autre animation (au choix)

Effets sonores (/5):

- En jeux : Musique de fond (au choix)
- Lorsqu'un animal se nourrit
- Lorsque la partie échoue (gameover)

Effets de particule (/5) :

- Lorsque de la nourriture est lancée