TP1 : Nourrir les animaux (/70)

Une image contenant mammifère, Animation, Silhouette d’animal, ciel

Description générée automatiquement

Attention: non représentatif du produit final.

**Finalité**

* 4 périodes de laboratoire sont prévues pour ce travail (deux semaines).
* Ce travail compte pour 20% de la session.
* Vous familiariser avec le développement de fonctionnalité à partir de requis et d’instructions

**Prérequis**

* Avoir complété les laboratoires 1 et 2.

**À remettre**

* La remise, **incluant une démo en classe** du résultat à l’enseignant, devra être faite **via GitHub avant le 24 septembre 2025, 23h59**
* Vous devrez me donner un accès à votre *repo* GIT du TP1 à [mathieu.piette@cegeplimoilou.ca](mailto:mathieu.piette@cegeplimoilou.ca) (Voir annexe 2 pour les instructions complète sur la création et le partage d’un repo GIT à votre coéquipière ou coéquipier, et l’enseignant du cours.)

**Notes importantes**

* Utiliser le *package* « **TP1 – Nourrir les animaux** », qui est fourni avec le TP1
* Le TP est **à réaliser seul** ou **en équipe de 2**

(ATTENTION : pour celles et ceux qui réaliseront le travail en équipe, il est obligatoire de compléter et remettre la grille de répartition des tâches, présentée en annexe 1).

# Description du jeu final attendu

* Le jeu doit être en vue de haut, de type « *space invader* », une légère perspective est permise
* Le personnage doit avancer continuellement au lancement du jeu, et peut 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 bas que le joueur), la partie est perdue et le jeu s’arrête
* Des animations vivantes qui nous permettent d’y croire
* Des effets visuels et sonores qui viennent bonifier l’expérience de jeu
* Un *GameOver* attristant

Une image contenant dessin humoristique, mammifère, capture d’écran

Description générée automatiquement

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

# Éléments à intégrer à votre projet

* Importer et utiliser ses propres sons (*.mp3*) avec les composants *AudioSource* et *AudioClip*
* Ajouter un *ParticleSystem* de *0* et le configurer adéquatement.
* Utiliser des *prefabs* pour créer un *array* de *GameObject[]*, et les faire apparaître de manière aléatoire.
* Différence d’utilisation pour un *Collider* avec *isTrigger = true;*
* Utiliser des variables *static* pour référencer les scripts uniques et importants de notre jeu.

# 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 peut 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 faire apparaître des animaux à intervalle aléatoire
* Lorsqu’instanciée, l’apparence de l’animal doit être sélectionnée aléatoirement parmi plusieurs (minimum 2) apparences différentes
* Le système d’animation doit correspondre à l’apparence de l’animal
* Les animaux sont instanciés 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’enfuit vers un des côtés 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 fait apparaître 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*, réinitialise 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 nourrit 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 délai aléatoire (+/- 50% du délai de base)
* Fait apparaître un animal aléatoire, à une position aléatoire
* Bonus : à chaque 15 secondes, le délai de base réduit de 5%, augmentant lentement la difficulté

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 jeu : 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

**Annexe 1 – Répartition des tâches**

Chaque membre de l’équipe doit remplir cette grille pour détailler sa contribution au travail.

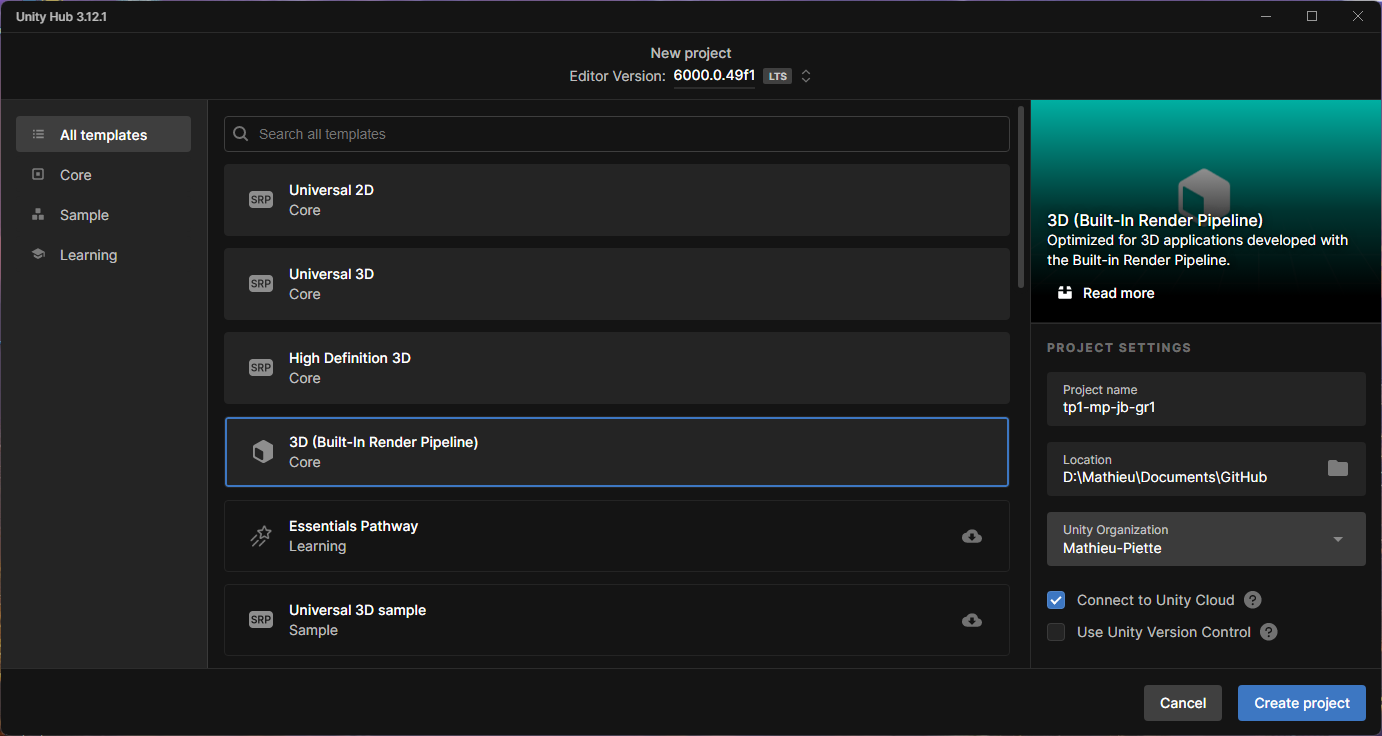
ATTENTION : Utilisez des verbes à l’infinitif (ex. : Programmer le script *PlayerController*, ajouter des sons, créer les animations, etc.).

|  |  |
| --- | --- |
| **Nom et prénom** | **Tâches réalisées** |
| [Étudiant 1] | * […] * […] * […] |
| [Étudiant 2] | * […] * […] * […] |

**Annexe 2 – Création d’un projet Unity, d’un repo GitHub, et partage du repo.**

# Création d’un nouveau projet

1. Ouvrez **Unity Hub**, puis sélectionnez ***New Project***
2. Sélectionnez la bonne version de Unity (**6000.0...**)
3. Choissez le modèle **3D (Built-In Render Pipeline)**, puis nommez votre projet en inscrivant tp1-[initiales étudiant 1]-[initiales étudiant 2]-[groupe]. Par exemple : « **tp1-xy-yz-gr1** »[[1]](#footnote-1).
4. Choisissez l’emplacement local de sauvegarde de votre projet, puis cliquez sur **Create project**
5. **Attendez que le projet soit créé et s’ouvre (cela peut prendre plusieurs minutes…)**

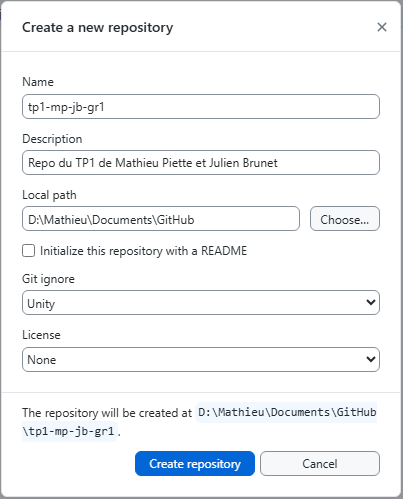


# Création d’un Repository GitHub et démarrage

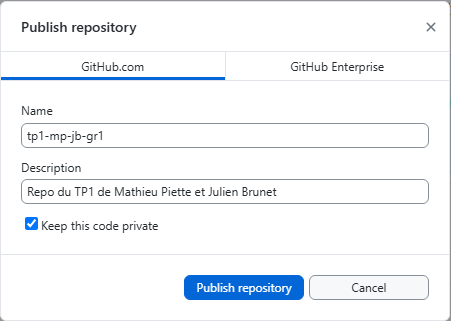
1. Une fois la création du projet terminée, fermez Unity et Unity Hub
2. Ouvrez GitHub Desktop, et créez un nouveau repository du même nom que votre projet Unity. Par exemple : « **tp1-xy-yz-gr1** ».
3. Ajoutez une description, puis spécifiez le chemin d’accès.

**ATTENTION** : Le chemin d’accès doit être la racine où vous stockez l’ensemble de vos projets Unity, pas le dossier du projet.

1. Sous *Git ignore*, sélectionnez Unity.
2. Cliquez sur *Create repository*



1. De retour dans l’interface de GitHub Desktop, cliquez sur *Publish repository*
2. Assurez-vous que *Keep this code private* est bien coché, puis cliquez sur *Publish repository.*



1. **Partage de votre repo à une coéquipière ou coéquipier, ainsi que l’enseignant du cours.**
2. Rendez-vous sur l’interface Web de GitHub (<https://github.com/login>)
3. Une fois connecté, dans le menu de droite, accédez à vos *Repositories*Une image contenant texte, capture d’écran, nombre, Police

   Le contenu généré par l’IA peut être incorrect.
4. Lorsque vous êtes dans votre Repo, cliquez sur *Settings*, *Collaborators*, puis *Add people.*

Une image contenant texte, Police, nombre, logiciel

Le contenu généré par l’IA peut être incorrect.

1. Ajoutez ensuite le courriel de votre coéquipière ou coéquipier (si le travail est réalisé en équipe), ainsi que mon courriel (mathieu.piette@cegeplimoilou.ca).

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Les personnes invitées recevront une invitation pour collaborer sur le repo, et pourront écrire sur ce dernier.

Bon travail!

1. Comme nos projets sont hébergés sur GitHub et que le nom de projet doit pouvoir former une URL fonctionnelle, nommez vos projets selon la convention kebab-case. [↑](#footnote-ref-1)