ATI L3 2021-22

Support de TD Unity

TD 1 (Cours 2) – Balade dans l'espace

14/10/19

Présentation

Dans ce TD, nous allons créer un environnement spatial comprenant :

- Un vaisseau spatial que le joueur contrôle.
- Un Soleil au centre et des planètes qui tournent autour.
- Des satellites qui tournent autour des planètes.
- Des vaisseaux qui vont d'une planète à l'autre.
- Des astéroïdes qui tournent aléatoirement.
- Une Camera qui suit lentement le joueur dans ses mouvements.

Des renvois au support de cours seront fréquents.

Démarche

Mise en Scène (Lumière, Caméra)

Objectifs	Tâche	Résultat
Configurer la Caméra [FACILE]	Mettez le Clear Flags de la caméra en Solid Color, et changez ce bleu ignoble pour du noir. Placez-la en (0,30,0), et mettez sa rotation à (90,0,0).	Le fond du jeu sera noir. La caméra propose à présent une vue du dessus sur votre scène vide.
Faire le Soleil [FACILE]	Créez une sphère. Renommez la et placez la en (0, 0, 0). Mettez son échelle à (8,8,8) ; Supprimez la Directionnal Light de la scène et créez une Point Light, que vous placerez aussi en (0, 0, 0). Augmentez le Range de la lumière pour 1000. Créez un nouveau material, et changez son Shader Type (tout en haut) pour un Unlit Color, et changez sa couleur en jaune. Appliquez ce material sur la sphère.	Vous aurez un magnifique Soleil.
Créer d'autres planètes [FACILE]	Créez d'autres sphères, de tailles différentes, que vous alignerez le long de l'axe X.	Vous aurez une succession de planètes. Vous pouvez leur créer des matériaux de toutes les couleurs.
Créer des lunes [FACILE]	Créez encore des sphères ! Plus petites cette fois, elles doivent être parentées à leurs planètes respectives.	Vos planètes auront des lunes.

Comportement du Joueur (Inputs GetKey, Translate, Rotate)

Objectifs	Tâche	Résultat
Préparer le vaisseau [FACILE]	Importez le « ship.fbx » fourni, et placez-le à côté du soleil. Sa rotation doit être à (0,0,0) et son échelle à (30,30,30).	Vous aurez un petit triangle qui fera office de vaisseau spatial. Vous pouvez lui aussi lui créer un material.
Créer le script [FACILE]	Créez un script C# « Player ».	Vous aurez un script.
Déplacement [FACILE]	Dans le script Player, créez une variable float publique « vitesseDeplacement ». Créez une condition d'input : Input.GetKey(KeyCode.UpArrow). Si la condition est vraie, faire une translation en forward, en fonction du Time.deltaTime et de vitesseDeplacement. Répétez l'opération pour reculer.	Si vous placez le script sur le vaisseau , il pourra avancer et reculer quand vous appuyez sur les flèches.
Rotation [FACILE]	Créez une variable float publique « vitesseRotation ». Créez une condition d'input avec la flèche de droite. Si vérifiée, alors faire une rotation en Vector3.up en deltaTime * vitesseRotation. Répétez l'opération pour tourner vers la gauche.	Vous pourrez mieux contrôler votre vaisseau intergalactique.

Comportement des Planètes (Rotate & RotateAround)

Objectifs	Tâche	Résultat
Script Planète [FASTOCHE]	Créez un script C# « Planete » (sans accents).	Vous aurez un deuxième script.
Faire tourner sur elle-même [FACILE]	Créez une variable float publique « vitesseRotation ». Faire une rotation en Vector3.up multiplié par le deltaTime et le vitesseRotation. N'appliquez pas le script Planete aux satellites, juste à la planète.	Votre sphère va tourner sur elle-même, et emportera les satellites avec elle, qui vont tourner autour. A ce stade on pourrait aussi parenter les planètes au soleil et faire pareil, mais nous allons procéder différement.
Faire tourner sur un point [INTERMEDIAIRE]	Créez une variable publique de type float, « vitesseTournerAutour », ainsi qu'une variable publique de type Transform nommée « centre ». Faites un RotateAround avec comme premier paramètre « centre.position », comme deuxième paramètres « Vector3.up », et comme troisème paramètre la variable de vitesse multipliée par le deltaTime.	En appliquant le soleil comme variable centre dans l'Inspector, puis en lançant le jeu, vous observerez votre planète tourner autour du soleil, avec les satellites qui tournent bien comme il faut. Pensez à mettre des speeds différents pour les planètes.

Comportement des Asteroïdes (Random)

Objectifs	Tâche	Résultat
Script Asteroïdes [EASY]	Créez un nouveau script C# « Asteroides ». (Toujours pas d'accents)	Un troisième script ! Vous pouvez commencer une collection !
Aléatoire du mouvement [INTERMEDIAIRE]	Créez une variable privée de type Vector3 : « axe » égal à Vector3.zero. Dans le Start(), assignez à axe.x une valeur aléatoire entre -90 et 90 (cf. support de cours). Faites de même pour axe.y et axe.z. Dans le Update(), faites une rotation en fonction de axe et du deltaTime. Créez des petits cubes et assignez-leur le script.	Les asteroïdes vont tourner sur eux-mêmes, mais chacun de façon unique.

Comportement des vaisseaux (MoveTowards)

Objectifs	Tâche	Résultat
Créer le script [SI VOUS NE SAVEZ PAS LE FAIRE A CE STADE, VOUS AVEZ UN PROBLEME]	Créez un script C# «Vaisseau».	Vous aurez 80 % de vos scripts finaux.
Faire aller d'un point A à un point B [INTERMEDIAIRE]	Créez une variable publique de type Transform « planete1 », et une variable publique de type float « vitesse ». Dans le Update(), créez un MoveTowards (cf. support de cours) entre la position actuelle et planete1. Placez un nouveau vaisseau en 3D dans la scène, et assignez-lui ce script et un joli matérial qui le distingue de votre vaisseau jouable.	En assignant une de vos planètes dans l'Inspector, l'objet ira de façon continue jusqu'à la position du Transform planete1.
Faire un aller et un retour entre A et B [DIFFICILE]	Créez une variable publique « planete2 » et une variable publique de type float « distanceMin » égale à 2f. Créez deux variables private bool « surPlanete1 » = false et « surPlanete2 » = true. Créez également une variable privée de type float « distance ». Dans le Update(), Créez une condition pour vérifier si surPlanete1 est vrai (cf support de cours). Si vrai, faire un MoveTowards jusqu'à planete2. Toujours dans la condition, récupérez la	L'objet va se diriger d'abord vers la planète 1, puis la planète 2, puis la planète 1, et ainsi de suite jusqu'à la fin des temps. Augmentez sa vitesse s'il n'arrive pas à rattraper la planète.

	distance entre le vaisseau et la planete2 (cf support). Si la distance est inférieure à distanceMin (cf support), alors surPlanete1 passe à faux et surPlanete2 passe à true. Répétez l'opération pour la seconde planète.	
Faire regarder l'objet en permanence vers sa direction [INTERMEDIAIRE]	A chaque MoveTowards, appliquez juste après un LookAt (cf support) vers la panète cible.	Le vaisseau regardera dans la bonne direction.

Comportement de la Camera du joueur (Lerp)

Objectifs	Tâche	Résultat
Script Caméra [PLUS SIMPLE QUE LA SUITE]	Créez un script C# « CameraPlayer ». Assignez le à votre Main Camera.	C'est le dernier script à créer pour ce TD.
Déplacement de la caméra [INTERMEDIAIRE]	Créez une variable publique de type Transform « joueur ». Créez une variable vitesse. Dans le Update, déclarez un Vector3 « ciblePos » qui sera égal à joueur.position. Puis rendez ciblePos.y égal au transform.position.y. Appliquez ensuite un Lerp (cf support) depuis la transform.position jusqu'à ciblePos, avec une allure égale à vitesse multipliée par le deltaTime.	Après avoir assigné « joueur » dans l'interface, vous pourrez constater que la Camera suit lentement le vaisseau, avec un léger retard mais en restant à sa hauteur.

Réaliser un Build du jeu

Objectifs	Tâche	Résultat
Enregistrez votre scène [FACILE]	Ctrl+S	Votre scène sauvegardée s'ajoute à vos Assets.
Modifier les Build Settings [FACILE]	File > Build Settings Rajoutez votre scène (par un glissé / déposé) dans la grosse boite grise en haut de cette nouvelle fenêtre.	Le nom de votre scène devrait y apparaître.
Lancez le Build [FACILE]	Cliquez sur Build et séléctionnez un emplacement **en dehors** des dossiers d'Assets de votre projet.	Vous aurez un exécutable de votre jeu à côté d'un dossier Data, d'un .dll, et quelques autres trucs. Le dossier Data et le reste sont essentiels au

		fonctionnement de l'exe.
Me rendre le Build [FACILE]	Rendez-moi l'éxecutable et son fichier Data dans un .zip à votre nom sur le FTP, dans Transfert > L3 > Unity > TD1	Vous éviterez le zéro et n'aurez pas travaillé si dur pour rien.

Objectifs Bonus

Accélération du vaisseau (condition complexe)

Objectifs	Tâche	Résultat
Accélération v1 [DIFFICILE]	Créez un float privé « acceleration » égal à zéro. Multipliez vos déplacement d'avancer et reculer par cette variable à la place de moveSpeed. Dans le Update, toujours, posez une condition pour savoir si on appuie sur accélérer **ou** reculer (cf. support de cours). Si vérifiée, incrémenter la valeur d'acceleration avec le temps (cf. support de cours). Sinon, la diminuer avec le temps (cf support de cours). Dans la condition principale, poser une nouvelle condition : si acceleration est supérieur à moveSpeed, la rendre égale à moveSpeed. Dans le sinon, poser une nouvelle condition : si acceleration est inférieure à zéro, alors la rendre égale à zéro.	Vous aurez un vaisseau qui accélère progressivement avant d'atteindre sa vitesse maximale.
Accélération v2 [DIFFICILE]	Créez un nouveau float public « accelerationSpeed », et un Vector3 privé « mouvement » égal à Vector3.zero. Lors de l'incrémentation d'acceleration (dans la double condition), multipliez le deltaTime par accelerationSpeed. De même pour la décrémentation. Mettez en commentaire vos deux lignes de Translate qui font aller en avant et en arrière votre vaisseau. Écrire juste en desous de chacun ceci : « mouvement = Vector3.forward; » pour avancer, et « mouvement = -1f * Vector3.forward; » pour reculer. Juste après les deux conditions d'input pour reculer et avancer, dans la bloc pricipal de l'Update(), créez une translation par le Vector3 « mouvement », multiplié par « acceleration » et le deltaTime.	Votre vaisseau accélérera selon votre accelerationSpeed, et le code sera maintenant plus propre. Il décélérera progressivement quand vous relâcherez le bouton, aussi.