Création d'une base mobile avec bras articulé pour la saisie et la manipulation d'objets en RV et implémentation de techniques d'Interaction 3D adaptées.

Projet RV, MASTER 1, EFREI, PR n° 3 du 25 octobre 2023.

Frédéric Davesne

0. Introduction

- Il s'agit de reprendre le cahier des charges construit au TD n°2 (voir exemple de corrigé sur https://ibisc.univ-evry.fr/~fdavesne/ens/) ainsi que le robot et les premiers codes construits au cours du PR n°1. L'objectif de ce PR n°3 est de finaliser la structure du programme de gestion du robot en respectant les 3 couches définies dans le cours, partie II.

1. Pré-requis et assistance

Pré-requis

- Le cours de Réalité Virtuelle et, en particulier, les techniques d'interaction 3D ;
- La correction du TD n°2 présente sur le WEB (https://ibisc.univ-evry.fr/~fdavesne/ens/) ;
- La trame de l'articulation du robot et les premiers scripts du PR n°1 dont une correction est donnée sur le WEB (https://ibisc.univ-evry.fr/~fdavesne/ens/).

Assistance

- Concernant le cours de Réalité Virtuelle, voir sur https://ibisc.univ-evry.fr/~fdavesne/ens/
- Concernant le projet en lui-même, les éléments concernant le robot (objets graphiques, script de gestion du bras articulé *gere_bras* (modèle inverse), documentation) se récupèrent ici: https://ibisc.univ-evry.fr/~fdavesne/ens/ens_efrei_m1/proj/data/elements_robot.zip

Travail à rendre:

- Le package unity montrant votre avancement au cours du PR n°3 à m'envoyer par courriel.

2. Mode opératoire

2.0. Introduction

A l'issue du PR n°2, vous avez créé un premier projet complet dont l'objectif est, pour le robot, d'aller saisir un cube et de le mettre dans la zone de dépôt, les opérations étant tracées dans un fichier, en utilisant l'architecture en 3 couches décrite dans le cours, partie II. La méthode pour savoir si un objet est sélectionnable ou si un objet est dans la zone de dépôt utilise la distance, soit entre le haut du cube et *Otool* ou la position de *Ocube* sur le plan OXZ.

2.1. Construction du terrain, des lumières, des objets cibles sur le terrain et des caméras.

Objectif: Il s'agit de créer les objets graphiques qui vont être utilisés dans le jeu.

Etape 1: Eléments de base du décor.
- Création du Terrain, que l'on nommera ***Sol*** et dont on redéfinira les dimensions à 15 mètres X 15 mètres. On placera son altitude Y à -0.50 mètres afin d'être juste en dessous des deux roues du robot mobile.
- Création de la zone de dépôt des objets nommée *Depot* (GameObject de type Cube d'une dimension de 3m x 3m x 0.1m, de couleur bleue).
- Création de deux lumières directionnelles pour éclairer la scène, à 3 et à 5 mètres du sol ;

Etape 2: Création des trois caméras, une exocentrique et 2 égocentriques (sur le robot) et d'un script *GereCameras* permettant d'activer une caméra et de désactiver les deux autres. Le passage d'un affichage de caméra à un autre se fera grâce à un des B1, B2 et B3. Pour cela, prendre exemple sur le script *camera_controller*.cs donné dans le fichier zippé contenant les éléments du robot. Mais utiliser un des contrôles d'applications associés à *B1*, *B2* ou *B3*.

Etape 3: Modifier le script *MachineAEtatsScript.cs* (étape 14 du PR n°2) afin d'intégrer la gestion des caméras.

Sauver la scène sous la scène « PR 3 - Scène 1 – Etape 3 »

2.2. Physicalisation

Objectif: Il s'agit de tenir compte de la gravité et des effets de collisions entre objets. Pour cela, on ajoute une propriété *Rigid Body* aux *Game Objects*.

Etape 4: Reprendre le Ocube initialAjouter une propriété *Rigid Body* à *Ocube*, mettre une masse de 0.1 kg et cocher la case *Use Gravity*. Lancer le programme et observer l'évolution de la position de *Ocube* : le *cube* passe au travers du sol ...

Etape 5: Ajouter une propriété *Box Collider* à *Ocube* et ajuster la taille et la position du *Box Collider* de manière à légèrement envelopper le cube. Lancer le programme et observer l'évolution de la position de *Ocube* : le *cube* ne passe plus au travers du sol. Si se n'est pas le cas, vérifier que le booléen *Is Trigger* associé au *Box Collider* du cube et le booléen *Is Trigger* associé au *Terrain Collider* du Sol ne sont pas cochés. Si oui, il faut les décocher.

Etape 6: Compréhension de *Is Trigger*. Créer deux cubes *Cube1* et *Cube2*. On constate que, par défaut, chacun de ces deux cubes est associé à un *Box Collider* de la taille du cube. Créer pour chacun un *Rigid Body* n'utilisant pas la gravité et ayant une masse nulle. Cocher la case *Is Trigger* associée au *Box Collider* d'un des deux cubes. Puis créer un script *SelectionnableTriggerScript.cs* qui utilise la fonction *void OnTriggerEnter(Collider objet_collision)* qui affiche dans la console le nom de l'objet qui a pénétré dans la zone de collision du cube. Cette fonction est appelée à chaque fois qu'un *Rigid Body* pénetre dans la zone de trigger, qui est le *Box Collider* associé au cube. Essayer cela en déplaçant "à la main" un des deux cubes. Que constatez-vous si le déplacement est très rapide?

Etape 7: Utiliser l'étape 6 afin de modifier le script *SelectionnableDistScript.cs* afin qu'un cube soit sélectionnable lorsque *Otool* rentre en collision avec la zone de saisie du cube.

Etape 8: Utiliser le procédé de l'étape 6 afin de détecter si un cube est bien dans la zone de Dépôt. Modifier pour cela le script *EnZoneDepot.cs* du PR n° 2.

Etape 9: Création des objets cibles sur le terrain.
On va les générer à partir d'un script *GenereObjetsScript.cs* à des positions aléatoires dans l'aire de jeu au début du jeu, à partir du cube *cube* tel que défini dans le PR n°2 et modifié à l'étape 7 de ce PR n°3. Les *GameObjects* générés auront seront nommés *Ocube<k>*, *Tcube* dépendant de *Ocube<k>* et *cube* dépendant de *Ocube<k>*, tels que définis dans le PR n°2. On utilisera en particulier la fonction *Instantiate() as GameObject* qui, à partir d'un GameObject donné en entrée, le copie et l'instancie à une position aléatoire (fonction *Random.Range()*). Regarder sur le Web l'utilisation respective de ces fonctions.

Sauver la scène sous la scène « PR 3 - Scène 2 – Etape 9 »

2.3. Mise en œuvre de la technique d'I3D "Tirer sur la corde" (*Grabbing the Air*)

Etape 10: Réécrire *NavigationScript.cs* afin d'implémenter la technique d'I3D "Tirer sur la corde" (voir le cours, partie II). Cette technique prend en entrée *increment_IR* et un contrôle d'application parmi *B1*, *B2* ou *B3* et produit en sortie une rotation *increment_rot_VR* et/ou une translation *increment_translation_VR* et utilise le script *BougeRobot.cs* pour modifier la position/l'orientation de la base mobile.

Sauver la scène sous la scène « PR 3 - Scène 3 – Etape 10 »

FIN