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° 2 du 18 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 du PR n°2 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

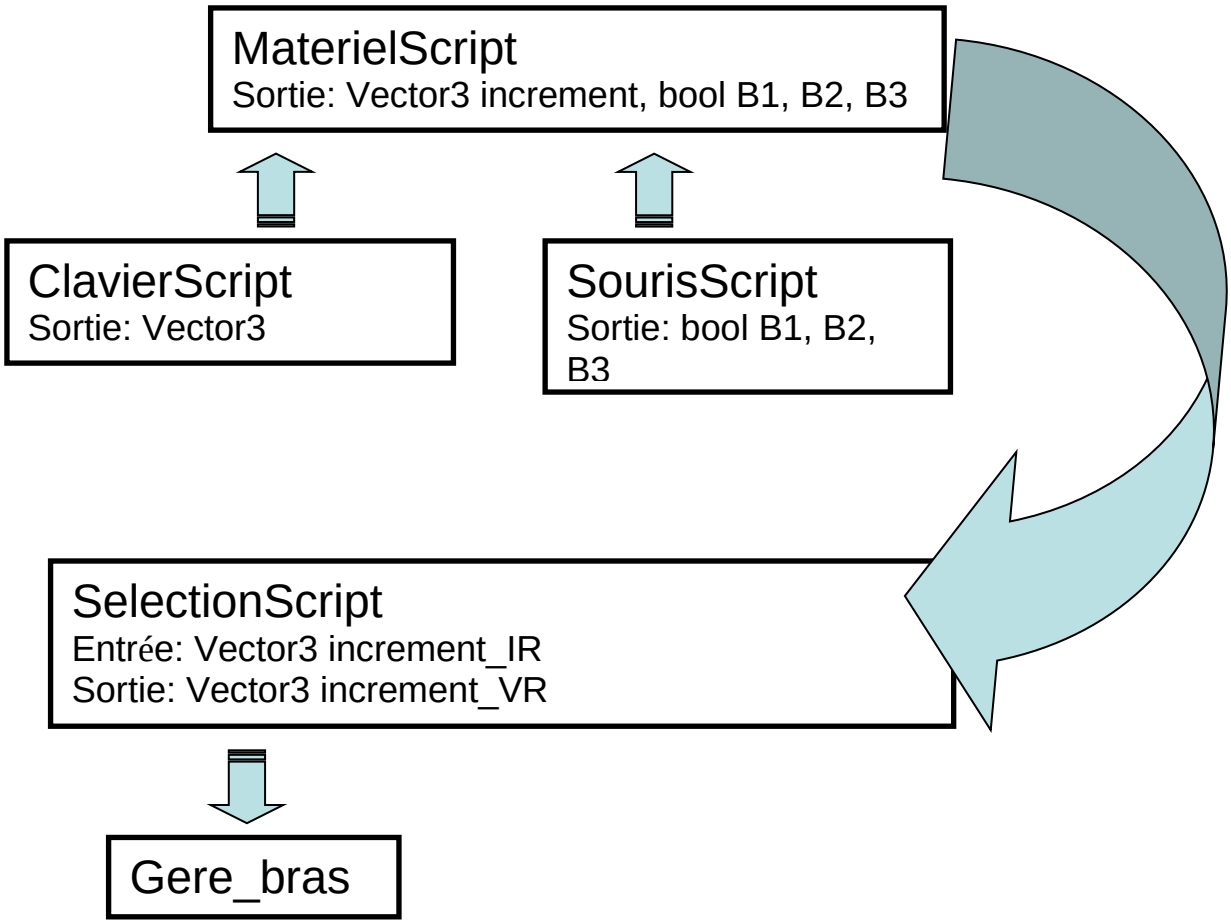
Travail à rendre:

- Le package de votre avancement au cours du PR n°2 par courriel.

2. Mode opératoire

2.1. Transformation du script ClavierScript.cs du PR 1

Objectif: Il s'agit d'obtenir une structure logicielle compatible avec les 3 couches du modèle d'I3D. Pour cela, on va s'attacher à créer la structure d'échange de données ci-dessous.



Les flèche signifient "fournit de l'information à".
Ainsi, les données des différents périphériques sont fournies à la classe *MaterielScript*: *ClavierScript* produit le *Vector3 increment* qui correspond à un incrément de déplacement au cours d'un tracking. La classe *SourisScript* produit les valeurs des 3 booléens correspondant aux 3 contrôles d'application autorisés dans le projet.
Les données en sortie de *MaterielScript* sont utilisées par la technique d'I3D Main Virtuelle Simple programmée dans *SelectionScript*. La mise à jour graphique effectuée par *gere_bras* utilise la sortie de *SelectionScript* comme entrée. C'est donc dans *SelectionScript* qu'il faut introduire une variable publique de type gere_bras: `public gere_bras gere_bras=null;`

Etape 1: Modifier le script *ClavierScript.cs* du PR n°1 et créer les scripts *MaterielScript.cs* et *SelectionScript.cs* sous l'*Empty Object O0* afin de respecter l'organisation logique proposée ci-dessus et que *SelectionScript.cs* implémente la technique *Main Virtuelle Simple*.

Etape 2: Créer le script *SourisScript.cs* qui récupère dans les booléens B1, B2 et B3 les évènements liés à l'appuie sur les trois boutons d'une souris. Pour cela, utiliser *Input.GetButton("Fire1" ou "Fire2" ou "Fire3")* ou ses dérivés *Input.GetButtonUp/Down()*.

Sauvegardez votre travail sous la scène « PR 2 - Scène 1 – Etape 2 »

2.2. Déplacement du robot mobile

Objectif: Mettre en œuvre une technique de Navigation de votre choix (Allez au plus simple!), qui va utiliser les données produites par *MaterielScript.cs* afin de tester la navigation du robot

mobile. Il n'est pas nécessaire pour le moment qu'elle soit *Grabbing the Air (Tirer sur la Corde)*.

Etape 3: Créer le script *NavigationScript.cs* associé à *O0* qui utilise en entrée le *Vector3 increment_IR* produit par *MaterielScript.cs* et qui possède en sortie le *Vector3 increment_VR*.

Etape 4: Créer le script *BougeBaseMobile*.cs associé à O0 qui utilise le vecteur *increment_VR* pour translater/tourner *O0* de *increment_VR* et tester la navigation du robot mobile. Le script *BougeBaseMobile.cs* joue le même rôle que le script *gere_bras.cs* . On pourra utiliser les fonctions *transform.Translate()* et *transform.Rotate()* appliquée sur *O0*.

Sauver la scène sous la scène « PR 2 - Scène 2 – Etape 4 »

2.3. Identification d'un objet sélectionnable par le robot, méthode 1

Objectif: Lorsque la pince du robot *Otool* s'approche "suffisamment prêt" d'un objet *cube* et que le robot se trouve dans l'état de sélection, la couleur de l'objet est modifiée et un booléen *EstSelectionnable* est mis à 1, ce qui permettra au robot de sélectionner l'objet et de passer dans l'état de manipulation.

Méthode 1: "suffisamment prêt" s'interprète par la distance entre l'objet et Otool inférieure à un réel positif *threshold*.

Etape 5: Importer le fichier *cube.fbx* dans votre projet dans le répertoire Graphique. Remarquer le lieu du point pivot

Etape 6: Créer un *Empty Object Ocube*, placé en position *(0.0, 0.0, 0.0)* et placer *cube* sous *Ocube* dans ***Hierarchy***. Faire en sorte que la position de *Ocube* corresponde à la position du point pivot de *cube*.

Etape 7: Créer un script *ToColor.cs*, que vous placerez sous Ocube, qui mettra *cube* à la couleur *color*. Créer pour cela une fonction *ToColor(Color color)*. La couleur d'un objet *object* se change grâce à un `object.GetComponent<Renderer>().material.color = Color.<couleur>`. Vous utiliserez la fonction *Find()* pour accéder à *cube* depuis *Ocube*. Faite en sorte d'initialiser la couleur du cube (fonction *Start()* de la classe *ToColor*) à *gray*.

Etape 8: Créer un *Empty Object Tcube*, qui dépend de *Ocube*, puis placer *Tcube* au sommet de *cube*. Nous allons ensuite considérer la distance entre *Otool* et *Tcube* pour savoir si le cube est sélectionnable ou non.

Etape 9: Créer un script *SelectionnableDistScript.cs* attaché à *Ocube*, qui permet de mettre à jour le booléen *EstSelectionnable* selon la distance de *Otool* à *Tcube*. Comment faire pour accéder à cette distance? Changer la couleur du cube en bleu s'il est sélectionnable et le faire reprendre sa couleur d'origine gray s'il n'est plus sélectionnable.

Etape 10: Vérification du bon fonctionnement des étapes 6,7, 8 et 9 en déplaçant le cube "à la main" dans la fenêtre scène.

Sauver la scène sous la scène « PR 2 - Scène 3 – Etape 10 »

2.4. Mise en œuvre de la prise et de la dépose d'un objet.

Objectif: En considérant les deux scripts *SelectionScript.cs* et *ManipulationScript.cs* [à créer], il s'agit de pouvoir saisir un cube et le déposer, tout en sachant s'il a été déposé dans la zone de dépôt.

.

Etape 11: Créer un script *PrendObjet.cs* qui ferme la pince du robot (graphique) et attache le GameObject *Ocube* à *Otool*.

Etape 12: Créer un script *LacheObjet.cs* qui ouvre la pince du robot (graphique) et qui détache possède comme entrée un des booléens B1, B2 ou B3 et qui ouvre la pince du robot (graphique) et qui détache le GameObject *Ocube* de *Otool*.

Etape 13: Créer un script *EnZoneDepot.cs* lié à *Ocube* qui met à jour un booléen *EstEnZoneDepot* si le cube est dans une zone de dépôt définie par 4 points.

Sauver la scène sous la scène « PR 2 - Scène 4 – Etape 13 »

2.5. Mise en place de la machine à états pilotant le basculement navigation/sélection/manipulation

Objectif: Mettre en œuvre la séquence complète d'enchainements entre navigation, sélection et manipulation en s'aidant du graphe d'état du TD n°2;

Etape 14: Créer un script *MachineAEtatsScript.cs* qui fait appel aux scripts *SelectionScript.cs, NavigationScript.cs, MaterielScript.cs* et *SelectionnableDistScript.cs* afin de permettre l'exécution de la technique d'I3D courante et la transition d'une technique d'I3D à une autre via un ou des contrôles d'application (un ou des booléens parmi *B1, B2* et *B3* fournis par *MaterielScript.cs*).

Sauver la scène sous la scène « PR 2 - Scène 5 – Etape 14 »

2.6. Création de la couche Application.

Etape 15: Placer la on les fonctions permettant de tracer les données telles que voulues dans le cahier des charges, dans un fichier nommé *log.dat*. Regarder sur le Web les fonctions qui permettent d'écrire dans un fichier sous C# Unity. Pour cela, créer un script *TraceData.cs* .

Etape 16: Aide graphique.

- Indiquer à tout moment l'état du système (quelle tâche d'I3D parmi Sélection, Manipulation, Navigation) et la durée depuis le début de la partie (voir *Time.time*). Pour cela, regarder en particulier sur le Web l'utilisation des ***UI Canvas*** et des ***UI Text***.

Etape 17: Finaliser le jeu pour la prise et la dépose d'un cube.

Sauver la scène sous la scène « PR 2 - Scène 6 – Etape 17 »

FIN