Projet RV, MASTER 1, EFREI, TD n° 2 du 27 septembre 2023.

Exemple de correction

Frédéric Davesne

2. Travail à effectuer

2.0. Réflexion préliminaire.

2.0.1. Dîtes comment vous comptez gérer les points suivants:

- Savoir si la partie est terminée ou non et dans laquelle des 3 couches vous trouvez l'information ;

La partie est terminée si les 10 objets sont dans la zone de dépôt. Nous définissons cette zone comme un rectangle compris entre 4 points *P1*, *P2*, *P3* et *P4*, 4 vecteurs 2D, définis dans la couche d'I3D. Nous définirons un booléen *finished* , dans la partie Application, qui vaudra vrai ssi les 10 objets sont dans la zone de dépôt. Ce booléen sera calculé par la fonction *is_finished()* selon un et logique appliqué aux 10 booléens associés à la présence ou non de chacun des 10 objets dans la zone de dépôt (voir ci-dessous). Ces 10 booléens sont pris dans la couche d'I3D. La zone de dépose pourra être définie de deux manières:

- une zone plane définie par 4 points 3D. Dans ce cas, on pourra créer une fonction permettant de savoir, pour chaque point (x, y , z) s'il appartient ou non à cette zone plane;
- un objet graphique rectangulaire dont on pourra tester la collision avec chacun des cubes (solution proposée dans le graphe de scène ci-après).

- Dans le premier cas, on devra savoir si un objet a été déposé dans la zone de dépôt ou non et dans laquelle des 3 couches vous trouvez l'information;

Nous associons alors à chacun des 10 objets graphique k un points Objk situé au sommet de l'objet et nous définissons une fonction *is_inside()* appliquée à chacun des 10 *Objk* qui retourne *vrai* si *Objk* est à l'intérieur du rectangle formé par P1, P2, P3 et P4. Un booléen est donc associé à chacun des 10 objets et se situe dans la couche d'I3D.

- Dans le deuxième cas, la fonction *is_inside()* regardera la collision, pour chacun des cubes, avec la zone de dépose.

- Savoir si un objet est sélectionnable par le robot ou non et dans laquelle des 3 couches vous trouvez l'information ;

Pour chacun des 10 objets k, nous calculons la distance de Objk par rapport à Opince. Si cette distance est inférieure au seuil *Thresh*, nous considérons que l'objet est *sélectionnable*, propriété attaché à l'objet Objk. Sinon, l'objet est non sélectionnable. Cette information et cette fonction se situent dans la couche d'I3D.

- Traçage des éléments suivants
  - Position du robot (caractérisé par le point O0) à un instant t ;
  - Position de la pince du robot (caractérisée par le point *Opince*) à un instant t ;
  - Etat du robot à un instant t ;
  - Etat des objets à un instant t (libre hors zone de dépôt, pris par le robot , libre dans la zone de dépôt) .

- etc. (voir le **cahier des charges**, section 3.3.)

Nous choisissons de tracer l'ensemble des données dans la couche Application, grâce à une fonction *trace_on()* qui spécifiera à chaque pas de temps l'ensemble des valeurs souhaitées. Cela permet d'obtenir un fichier possédant un nombre de colonnes fixe et gérable facilement pour une exploitation des données ex post.

2.0.2. Dîtes comment vous comptez réaliser la communication entre les trois couches de l'application.

La communication se fera par lecture de variables produites dans une couche par une autre couche. L'idée est que la lecture se fasse d'une couche supérieure ou égale à la couche où la donnée est produite.

2.1. Conception de la partie Interaction 3D

2.1.1. Donner les objets ayant part à l'interactivité de l'application, suivant le cahier des charges et l'anatomie du robot mobile

Les objets ayant part à l'interactivité sont les objets étant directement impliqués dans les tâches de navigation, sélection, manipulation ou contrôle d'application. Nous utiliserons la fonction donnant le modèle inverse du bras articulé, permettant de contrôler la position des différents éléments du bras connaissant la position de la pince *Opince*. La conséquence à cela est que **Opince est l'unique objet du bras articulé impliqué dans l'interactivité du bras du robot**.

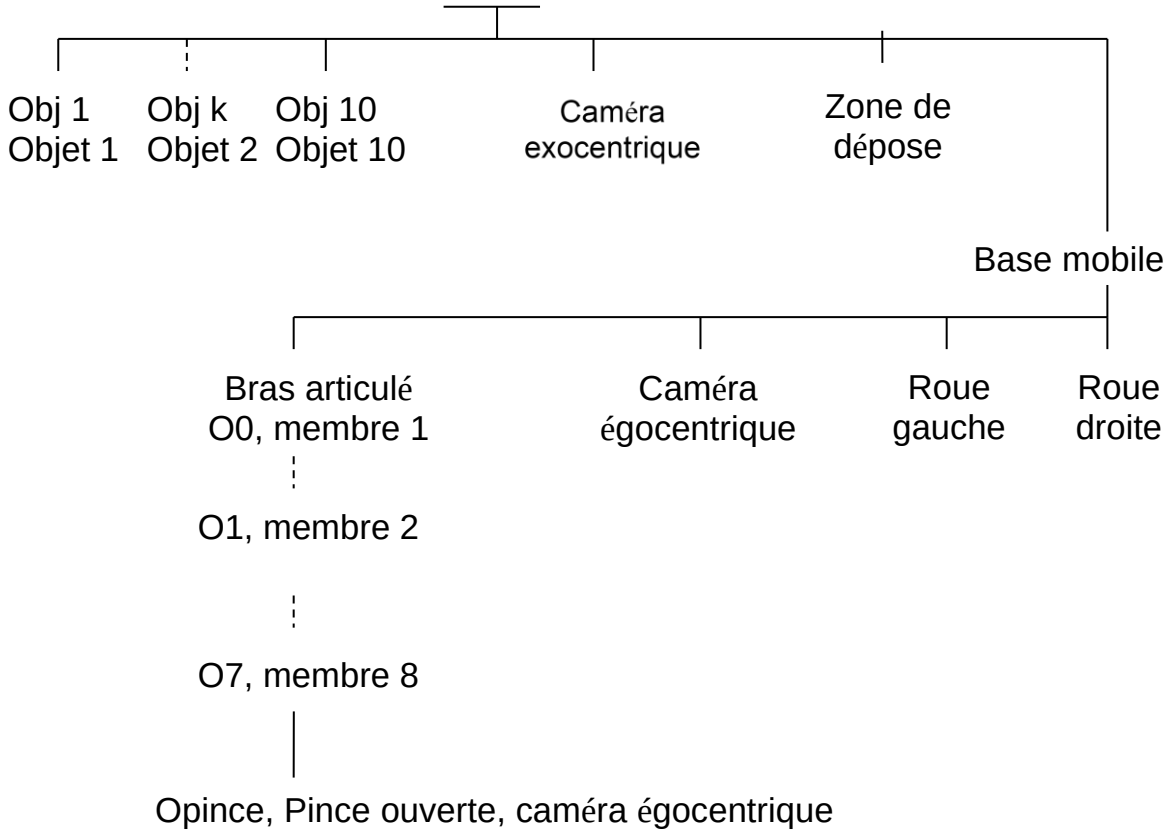
L'objet O0 sera impliqué dans la navigation du robot mobile, et uniquement lui.

Les deux objets *pince gauche* et *pince droite* seront impliqués dans le contrôles d'application permettant la transition entre la phase de sélection et la phase de manipulation (pince "ouverte" vers pince "fermée").

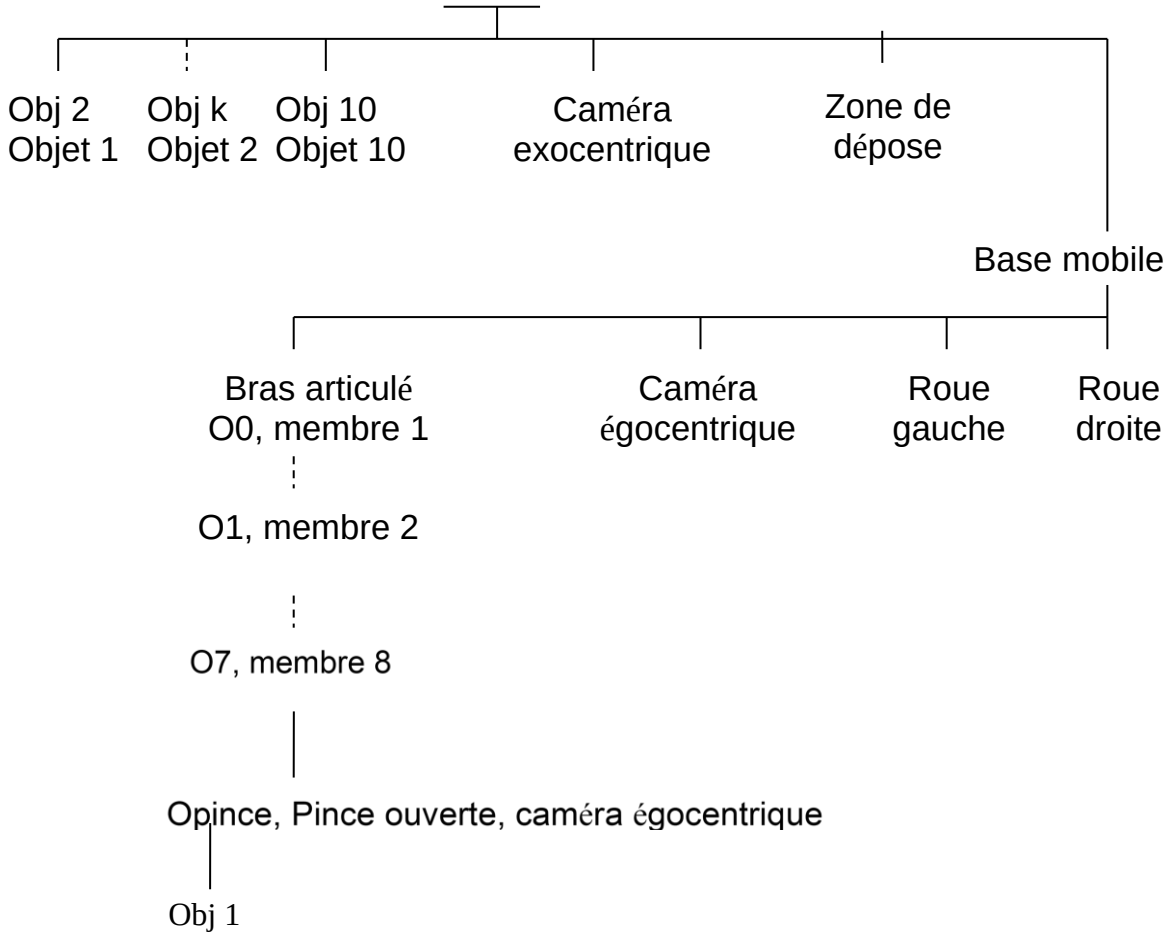
Les 3 caméras sont impliquées dans le contrôle d'application permettant le changement de vue de l'utilisateur.

Les 10 objets *Obj1*, ..., *Obj10* sont impliqués dans les phase de sélection et manipulation, puis dans le fait de savoir si la partie est terminée ou non.

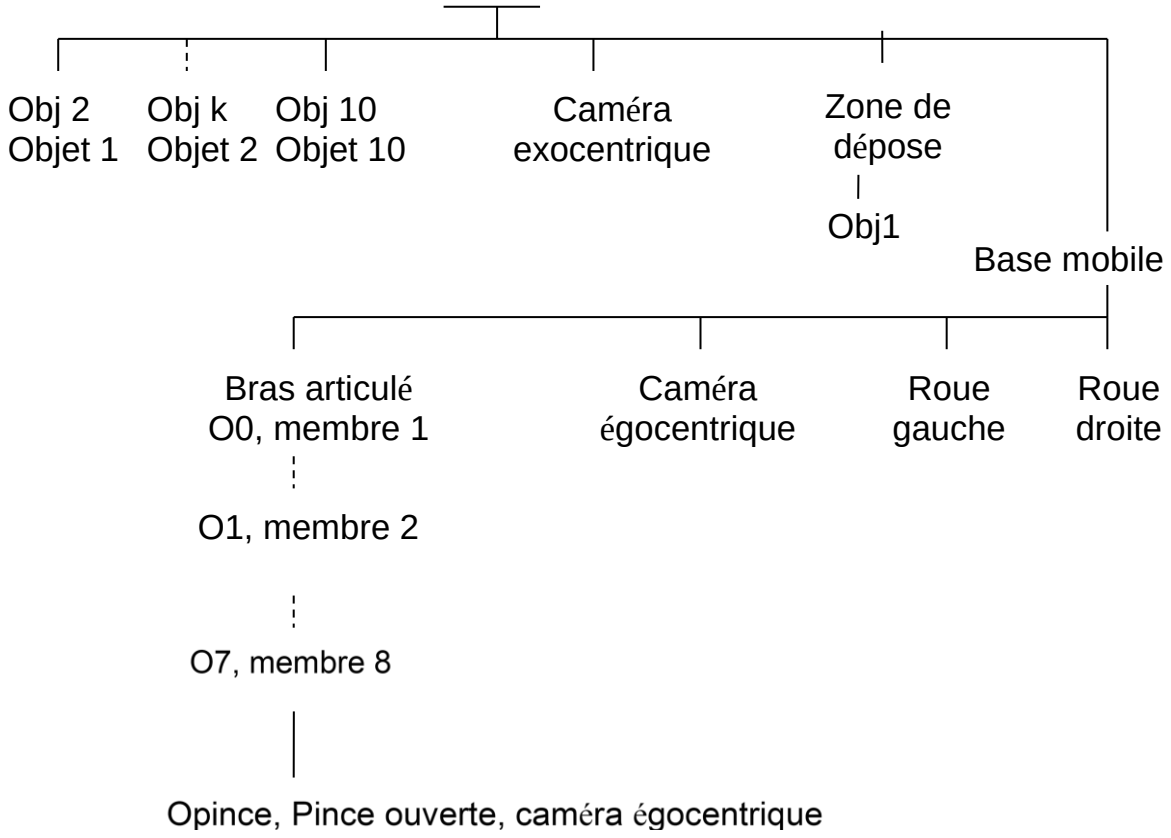
2.1.2. Donner un graphe de scène initial compatible avec le cahier des charges, tenant compte des différents éléments du robot et de leur hiérarchie éventuelle.



Dans le cas où Obj 1 est sélectionné, le graphe de scène devient:



Dans le cas où Obj 1 est déposé dans la zone de dépose, le graphe de scène devient:



Le graphe de scène se retrouvera dans la partie ***Hierachy*** de Unity 3D (à construire au cours du TP n°1)

2.1.3. Donnez les techniques d'I3D que vous allez utiliser (voir cours, partie II).

- variables d'entrée ;
- variables de sortie ;
- lien fonctionnel entre les entrées et les sorties.
- concernant le contrôle d'application, le cahier des charge stipule que 3 contrôles d'application **maximum** sont autorisés.

Tâche de navigation: Nous optons pour la technique *Grabbing The Air* [Mapes et al., 1995]. Pour cela, nous supposons que nous disposons d'un *pinch glove* et d'un périphérique permettant de retourner une position 3D (par exemple un Flystick) sur notre plateforme de RV. La technique *Grabbing The Air* retourne à chaque pas de temps un vecteur 3D correspondant à la vitesse de déplacement de la base mobile (orientation et amplitude de la vitesse de déplacement).

Tâche de sélection: Nous allons utiliser la technique de la *Main Virtuelle Simple* (*MVS*), demandant de connaître à tout moment la position (voire l'orientation) de la main de l'utilisateur sur la plateforme de RV. L'utilisation du Flystick (voir la tâche de navigation) nous permet de faire cela. L'utilisation de *MVS* va s'appliquer sur la position de la pince du bras articulé du robot mobile virtuel. Cela suppose que nous connaissions le modèle inverse du bras articulé pour connaître à tout moment les positions angulaires des différents éléments du bras articulé.

Tâche de manipulation: Idem que pour la tâche de sélection.

Tâche de contrôles d'application: Nous allons utiliser 2 contrôles d'application différents, qui correspondront à 3 processus pouvant s'exécuter d'une manière asynchrone:

- Le changement de tâche élémentaire d'I3D (sélection, manipulation et navigation) ;
- Le changement de point de vue (1 vue parmi 3 vues possibles) ;
- Le changement d'état de la pince (ouverte ou fermée).

Cela permet en particulier d'effectuer un changement de point de vue au cours d'une des tâches élémentaires d'I3D afin d'améliorer le confort de l'utilisateur et peut-être la précision de la saisie d'objets.
Le contrôle d'application s'effectuera avec 2 des boutons du Flystick.

Variables contrôlées par le système d'interaction

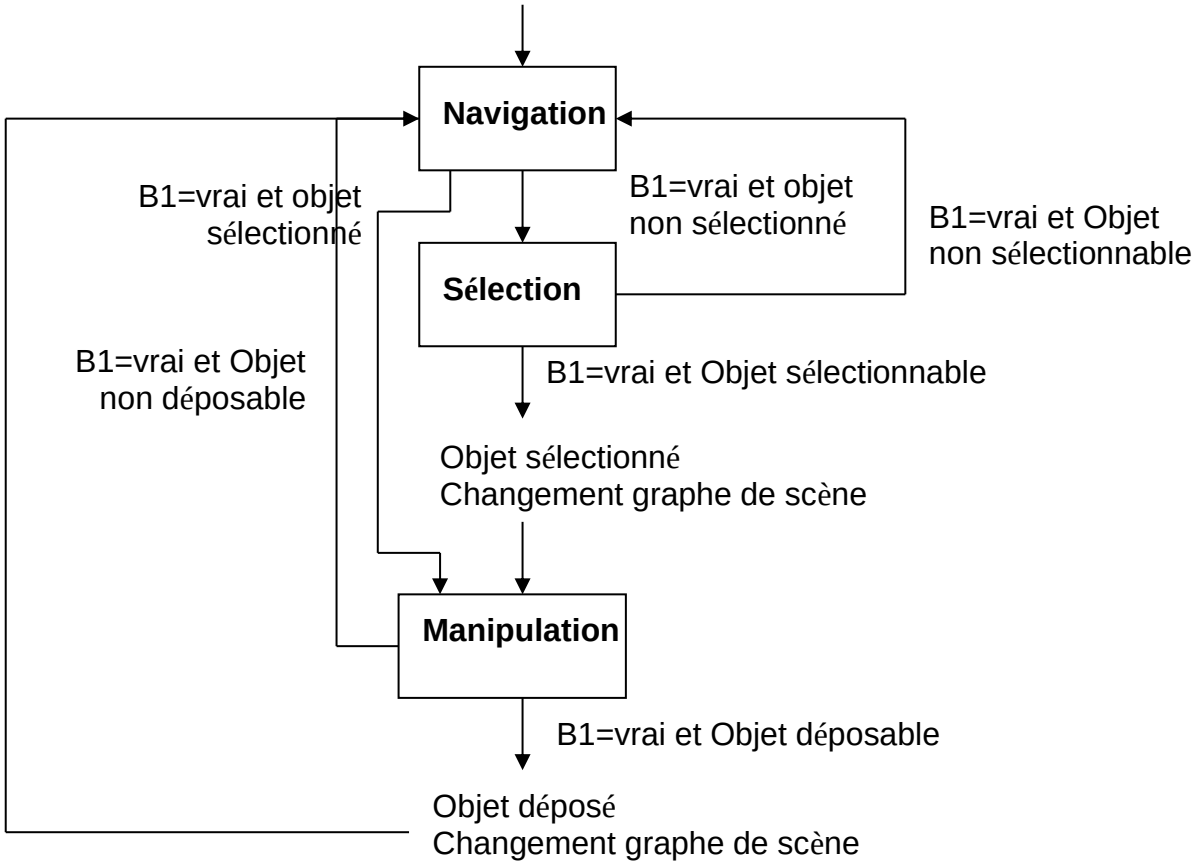
- Vitesse de la base mobile V. Vecteur 2D, déduit des données 3D Flystick par projection sur le plan horizontal, qui se traduit par une orientation (angle Θ) et une amplitude de vitesse A ;
- Position P de la pince du robot, vecteur 3D ;
- 2 booléens *B1* et *B2*, permettant le changement d'état dans les 3 processus associés aux 2 contrôles d'application.

L'ensemble de ces variables permet de déterminer d'une manière univoque l'évolution du robot virtuel.

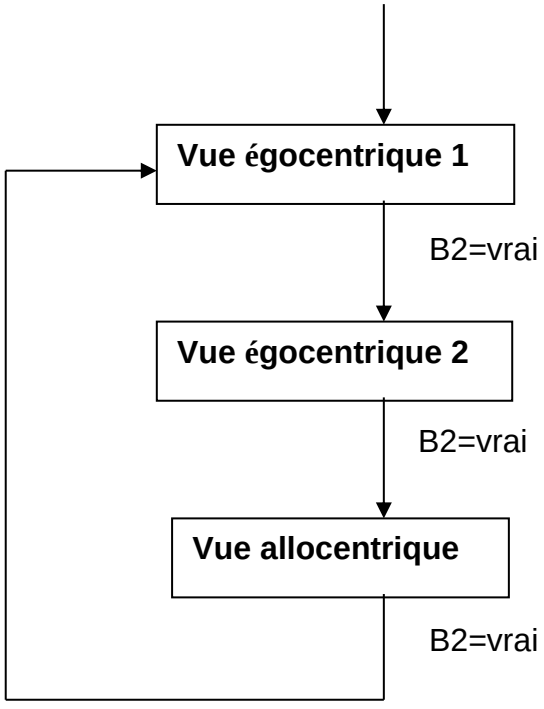
2.1.4. Donner les séquences de tâche d'I3D élémentaires et construire un graphe d'état tenant compte de toutes les évolutions possibles du système interactif, sachant que la tâche d'I3D initial pour le robot mobile est la navigation.

- ne pas oublier que les transitions dans le graphe d'états s'effectuent au moyen de contrôles d'application et de leur variable (souvent booléenne) associée.

1. Processus de gestion des tâches élémentaires d'I3D

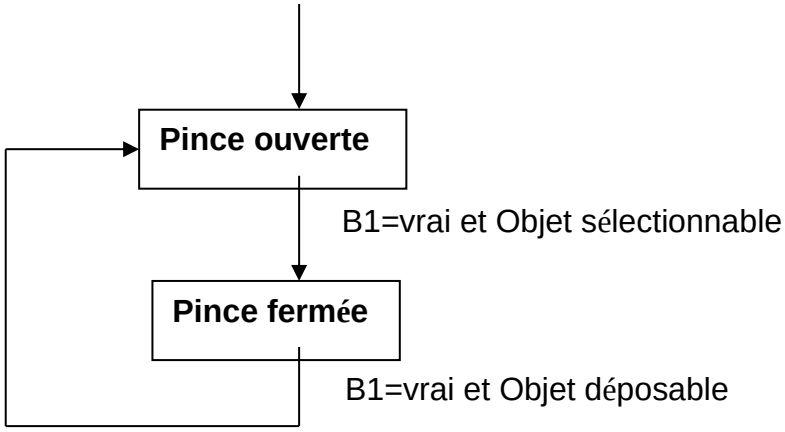


2. Processus de gestion des points de vue



3. Processus gérant la pince

Je considère que l'état de la pince dépend de l'état du processus 1: lorsqu'on est dans les états **Navigation** ou **Sélection**, la pince est ouverte. La pince se ferme lors de la transition de l'état **Sélection** à l'état **Manipulation**. La pince s'ouvre lors de la transition entre l'état **Manipulation** à l'état **Navigation**.



2.2. Conception de la partie Application.

- Donner la ou les variables associées à cette partie ;

Le booléen *finished*.

- Donner la ou les fonctions (avec les entrées et sorties) associées à cette partie.

Les fonctions *is_finished()* et *trace_on()*

2.3. Conception de la partie Interface Matérielle

- Donner la ou les variables associées à cette partie ;

Nous mettons à jour un incrément *inc* qui permettra aux variables utilisées comme entrée dans la couche d'I3D, c'est-à-dire *V* (navigation) et la position de *Opince* (sélection et manipulation) d'évoluer.

D'autre part, nous mettrons à jour les booléens *B1* et *B2*.

- Donner la ou les fonctions (avec les entrées et sorties) associées à cette partie.

Les fonctions *Update_inc()*, *Update_B1()* et *Update_B2()* réaliseront les tâches de la partie 2.3.

Ces trois fonctions seront réécrites selon les périphériques utilisés. En salle machine, *inc*, *B1* et *B2* seront générées en utilisant le clavier et/ou la souris de l'ordinateur.

FIN