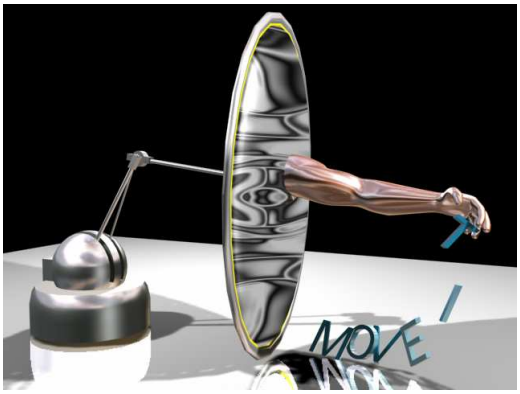
Conception préliminaire



28/11/2005

Historique du Document			
Version	***Date***	***Auteur/Participant***	***Apport au document***
1.0	20/11/05	Chefs d'équipe	Rédaction par module
1.6	25/11/05	Chefs de projet	Rédaction et intégration
1.10	30/11/05	Clément PICOU	Re lecture

SOMMAIRE

TABLE DES FIGURES

TABLE DES DIAGRAMMES DE SEQUENCE

Introduction

Le projet MoveIT est un système informatique permettant la manipulation d'objets dans un environnement virtuel. L'interaction se fait au moyen de diverses modalités telles que l'utilisation d'un bras haptique, d'un système vocal ou plus classiquement à l'aide d'un clavier et d'une souris.

MoveIT se présente sous la forme d'une bibliothèque et d'une application d'exemple utilisant la majorité des fonctionnalités disponibles. Cette application représente un atelier d'assemblage de pièces Lego®.

L'environnement virtuel s'appuie sur des propriétés physiques lui donnant une forte sensation de réalisme. De plus il est entièrement configurable par l'utilisateur.

La bibliothèque étant conçue de manière générique, il est aisé de greffer de nouvelles fonctionnalités développées par la suite. Du point de vue utilisateur de cette bibliothèque, il est possible de créer rapidement une application regroupant de nombreuses notions de réalité virtuelle, disponibles dans la bibliothèque. Ainsi vous pourrez créer, modifier, manipuler ou encore enregistrer des objets dans l'environnement 3D. Vous pourrez interagir avec cet environnement par l'intermédiaire d'un bras haptique mais aussi avoir un aperçu de cet environnement au travers d'un écran ou de lunettes stéréoscopiques. D'autres modalités d'interaction telles que la synthèse vocale permettent une grande souplesse d'interaction.

L'application servant d'exemple met en œuvre les principales fonctionnalités de la bibliothèque MoveIT. En effet, il s'agit d'un atelier d'assemblage de briques LEGO® représenté sous la forme d'une chambre d'enfant. Vous pouvez vous déplacer dans ce monde virtuel, concevoir aisément des constructions à l'aide de toutes les modalités décrites dans la bibliothèque et enregistrer votre travail

Dans ce document nous allons présenter la conception préliminaire de notre projet MoveIt. Ce document présente tout d'abord l'architecture fonctionnelle et matérielle prévue pour notre système. Cette architecture fonctionnelle justifie le découpage en modules que nous présenterons par la suite. Des diagrammes de séquences mettent en scène ces différents modules lors des différentes actions que notre système peut être amené à effectuer.

1. Présentation générale

1.1. Modules

Nous avons divisé l'application en cinq modules :

- Le module Bras Haptique :

Ce module est chargé de concevoir une librairie de fonctions, permettant d'utiliser le bras à retour d'efforts (Virtuose 3D15-25) pour manipuler des objets virtuels 3D dans diverses applications.

Il est composé de 5 étudiants :
BOISSIER Enguerran
CABRILLAT Guillaume
FUDYM Maxime
GUIOMAR Adrien
PEYRUQUEOU Vincent

- Le module GUI – Visualisation 3D :

Ce module est chargé de réaliser la GUI permettant à l'utilisateur d'interagir avec l'application ainsi que de concevoir une librairie de fonctions permettant d'afficher le monde virtuel 3D.

Il est composé de 6 étudiants :
AMIEL Eric
ARQUE Sébastien
CABIECES Julien
DANGOUMAU Guillaume
ENJALBERT Elodie
ORVAIN Emmanuel

- Le module Vocal :

Ce module est chargé de réaliser une bibliothèque de fonctions permettant de traiter automatiquement de la parole en entrée et de permettre un retour vocal à l'utilisateur.

Il est composé de 4 étudiants :
BARREAU Pascal
BRETT Caroline
GALLARD David
PICOU Clément

- Le module Moteur Multimodal :

Ce module est chargé de concevoir le moteur de fusion/fission qui doit permettre à l'utilisateur de réaliser une tâche en utilisant plusieurs outils d'interactions en même temps. Ici, en entrée avec le bras et la parole et en sortie avec le bras, la synthèse d'images et la synthèse de la parole.

Il est composé de 5 étudiants :
COLLIER Alexandre
FONQUERNIE Isabelle
LORTET Fabrice
MENINI Francesco
RABAH El Mehdi

- Le module Réseau – Noyau Fonctionnel :

Ce module est chargé de réaliser un réseau permettant aux différents modules de pouvoir communiquer puisque l'application est distribuée sur différents ordinateurs.

Il est composé de 4 étudiants :
ANTON Lionel
BOURIANES Robin
KAMEL Aurélien
LAURENS Guillaume

1.2. Architecture matérielle

L'application est répartie sur 3 ordinateurs reliés par le réseau comme indiqué sur le schéma suivant.
Pour faire fonctionner le bras haptique (Virtuose 3D15-25), l'application doit obligatoirement utiliser le coffret de contrôle/commande (PC 1). La librairie concernant le module du bras se trouve sur ce PC.
Le Module de Visualisation 3D ayant besoin d'une carte graphique et d'une puissance de calcul suffisamment importante, un deuxième ordinateur (PC 2) permet de faire fonctionner la totalité de ce module.
Enfin, les fonctions des modules Vocal et Moteur Multimodal tournent sur un troisième ordinateur (PC 3).

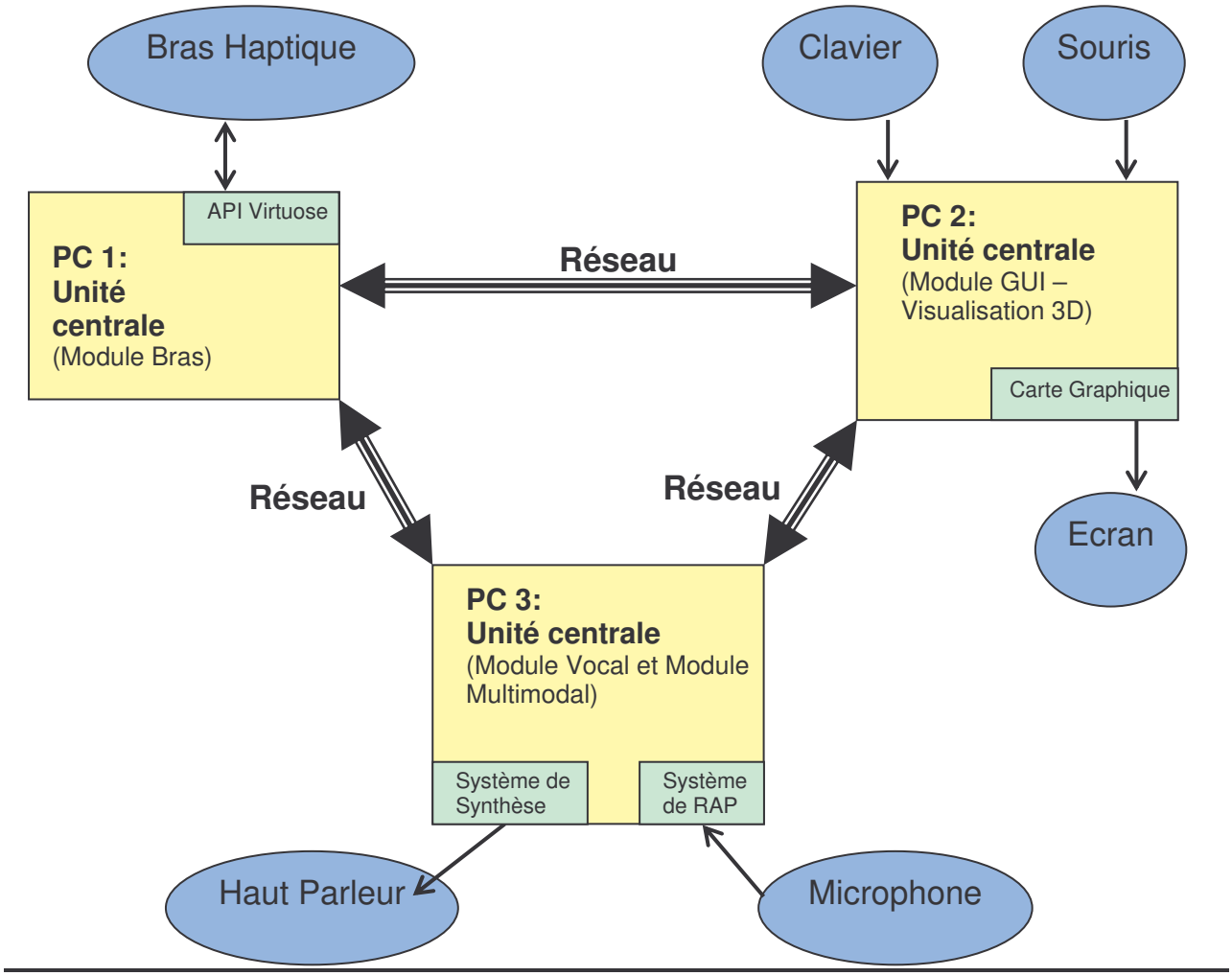


Figure 1 - Architecture matérielle

1.3. Architecture logicielle

L'architecture logicielle de l'application se compose de 6 modules :

→ Le module bras haptique : il est chargé de la gestion du bras haptique et de ses interactions avec l'environnement virtuel. Nous devons principalement analyser les gestes réalisés par l'utilisateur mais aussi faire profiter l'utilisateur d'un retour tactile avec l'environnement virtuel.

→ Le module Visualisation 3D : il prend en charge la gestion de la scène 3D virtuelle et le rendu de celle-ci. Il est le seul à avoir une représentation complète de l'environnement virtuel.

→ Le module GUI : c'est l'interface graphique dont dispose l'utilisateur pour agir sur l'application. Il contient une fenêtre de rendu et des menus apparaissant sur la demande de l'utilisateur.

→ Le module Noyau Fonctionnel : il gère tous les accès à la base de donnée. Celle-ci contient la représentation de l'ensemble des objets pouvant être ajoutés à l'environnement virtuel. De plus ce module renseigne les requêtes venant des modules qui souhaitent recevoir des informations sur l'environnement virtuel.

→ Le module Multimodal : il est chargé de faire la fusion entre les périphériques d'entrée du système et les données qui leurs sont associés. Il doit également réaliser la fission des informations qu'il collecte afin de générer en sortie du système des informations pertinentes pour l'utilisateur.

→ Le module Vocal : il est chargé d'offrir à l'utilisateur la possibilité d'agir sur l'application par la parole ainsi que de bénéficier d'indications orale sur l'état de l'application.

Nous explicitons sur le schéma suivant, l'organisation de ces différents modules et les principales informations qu'ils échangent pour le bon fonctionnement de l'application.

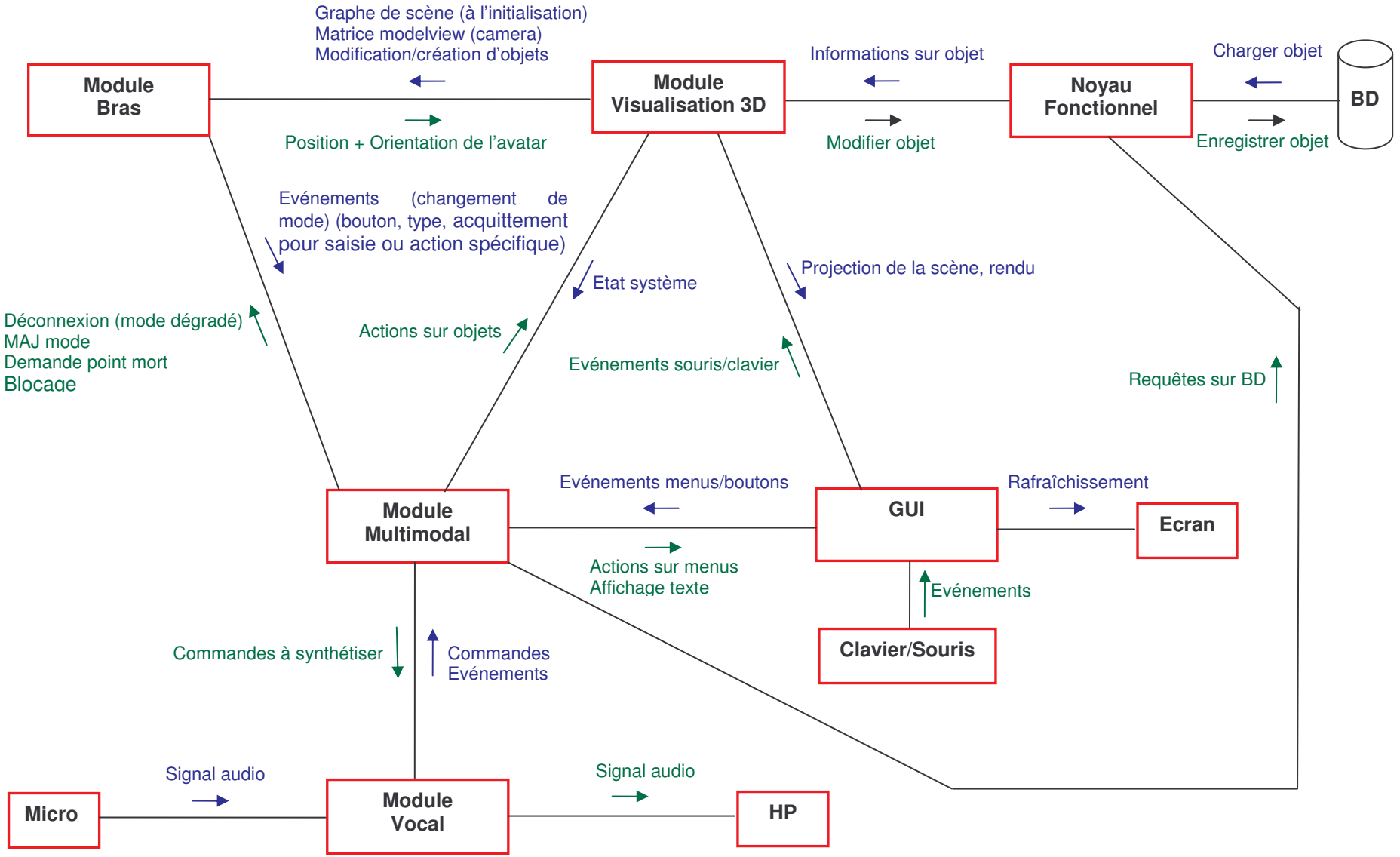


Figure 2 - Architecture logicielle

1.4. Interfaces externes

Les interfaces externes des modules correspondent aux différentes données qu'ils sont capables d'émettre et/ou recevoir. Ces interfaces sont décrites par la suite dans les diverses sections de ce document.

Les interfaces externes au système, pour l'utilisateur, sont les périphériques de l'application. L'application est capable d'interpréter les données provenant de l'utilisateur au travers de ces périphériques mais aussi de faire un retour de l'état du système à l'utilisateur. Pour cela, on dispose des interfaces suivantes :

→ Interface bras haptique (interface du module bras) :
En entrée : les données liées à la manipulation du bras haptique. C'est-à-dire les efforts effectués sur le bras et l'état des boutons qui lui sont associés.
En sortie : un retour d'effort perceptible par l'utilisateur.

→ Interface parole (interface du module vocal) :
En entrée : un flot de donnée audio (voix de l'utilisateur).
En sorite : un flot de donnée audio (synthèse vocale).

→ Interface visuelle (interface du module GUI) :
En sortie : l'affichage de l'environnement 3D et des menus de l'application.

→ Interface clavier/souris (interface su module GUI) :
En entrée : des évènements associés à la souris et au clavier.

1.5. Support de communication

Les modules de l'application communiquent soit directement entre eux (communication inter processus) s'ils sont installés sur la même machine, soit par l'intermédiaire d'un réseau local qui connecte les différentes machines de l'application. Pour les détails liés au réseau, se reporter à la section réseau.

1.6. Diagrammes de séquences globaux

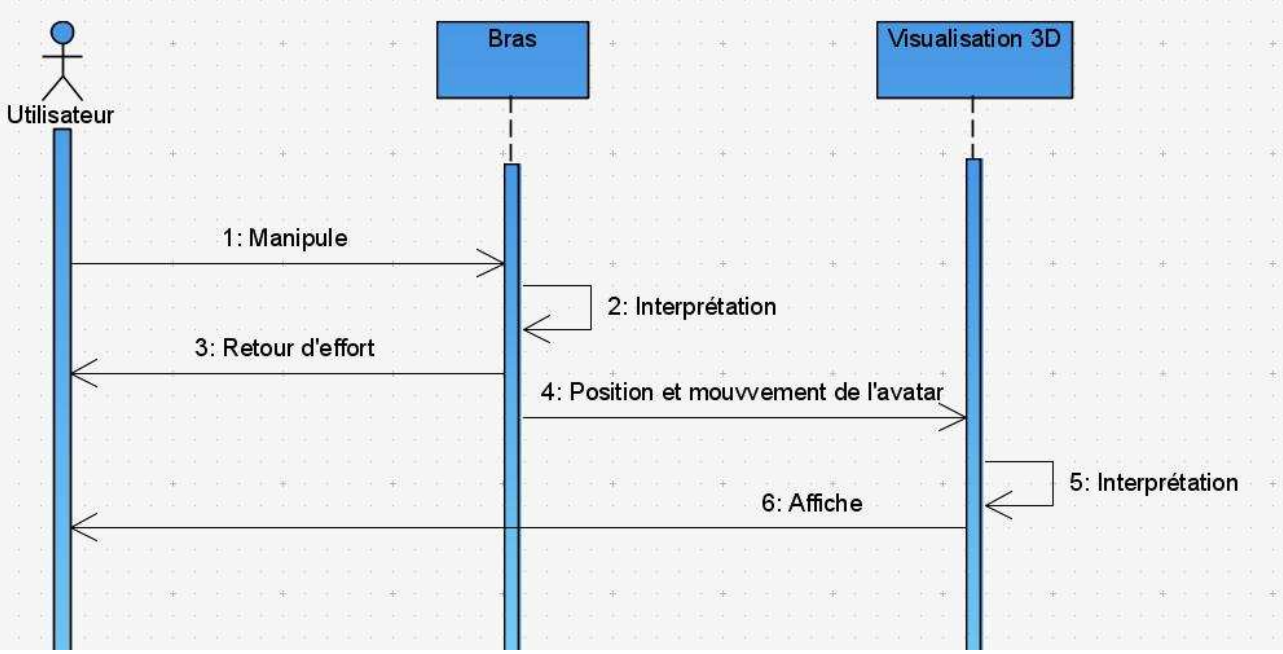


Diagramme de séquence 1 : Actions continues

Ce diagramme représente les actions « continues » effectuées par l'application. Ce sont en fait des fonctions périodiques mais nous avons choisi de les faire apparaître sur ce diagramme afin de ne pas avoir à présenter ces actions dans tous les diagrammes suivants.

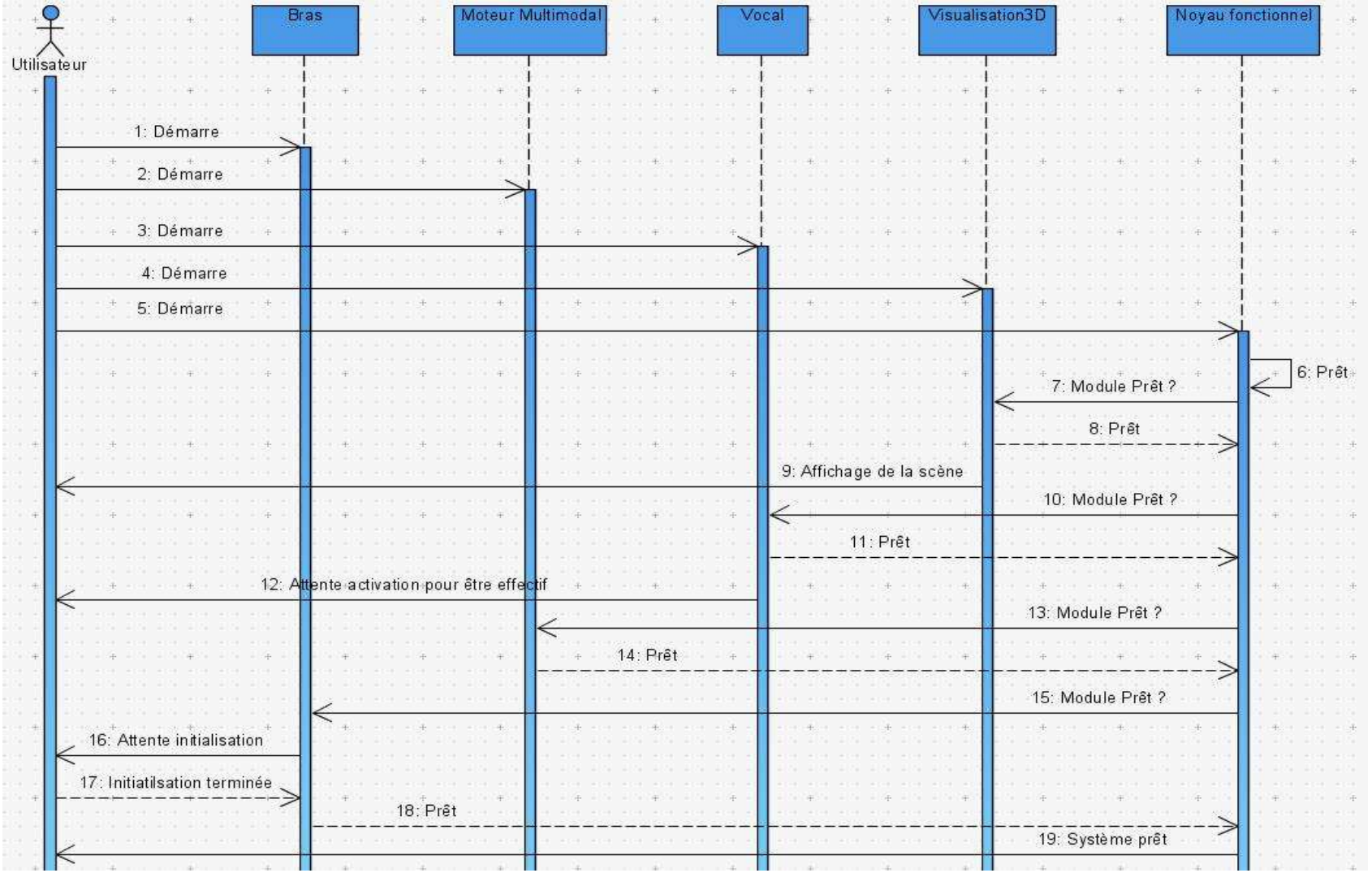


Diagramme de séquence 2 – Démarrage de l'application

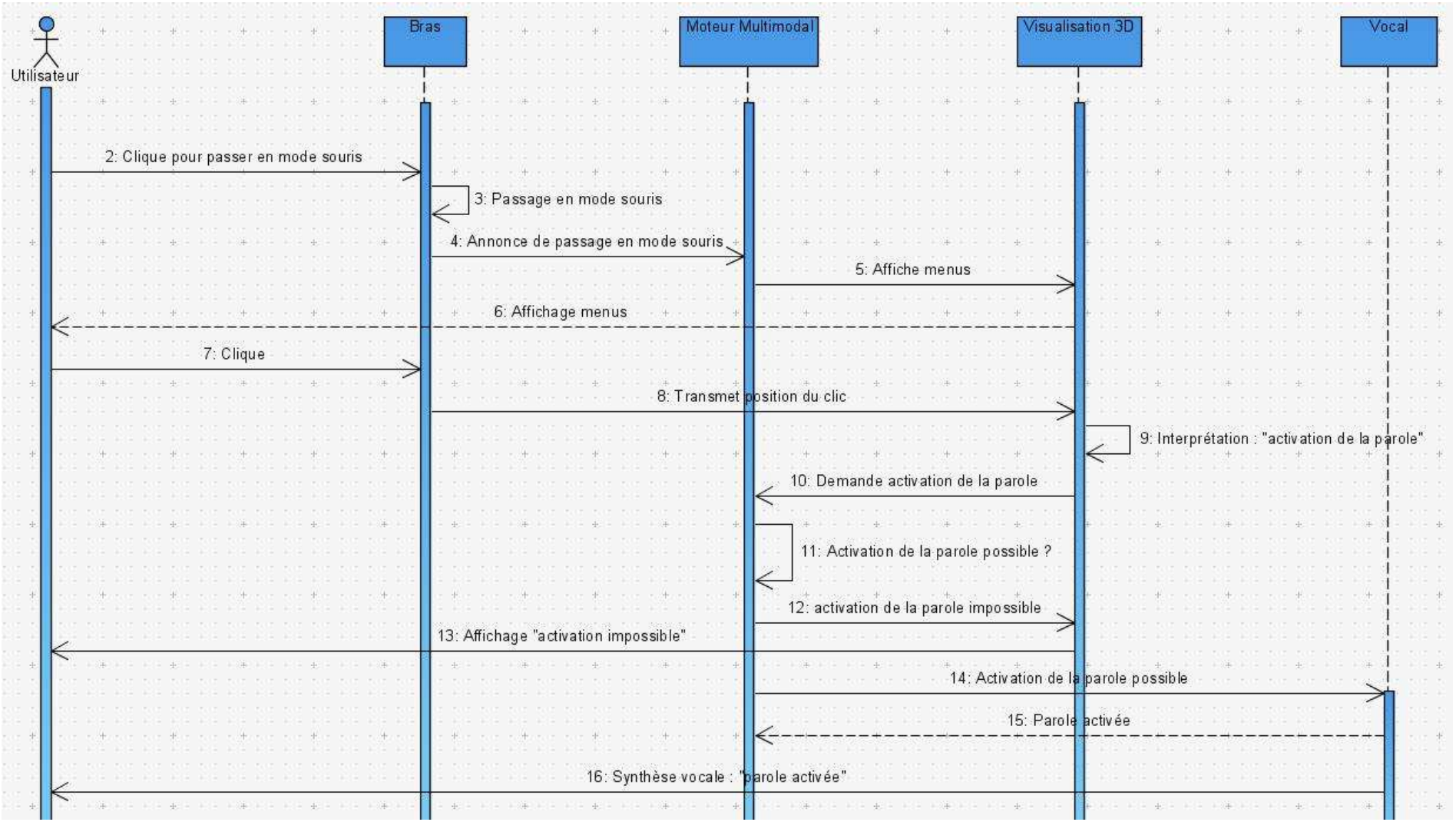


Diagramme de séquence 3 – Activation de la parole

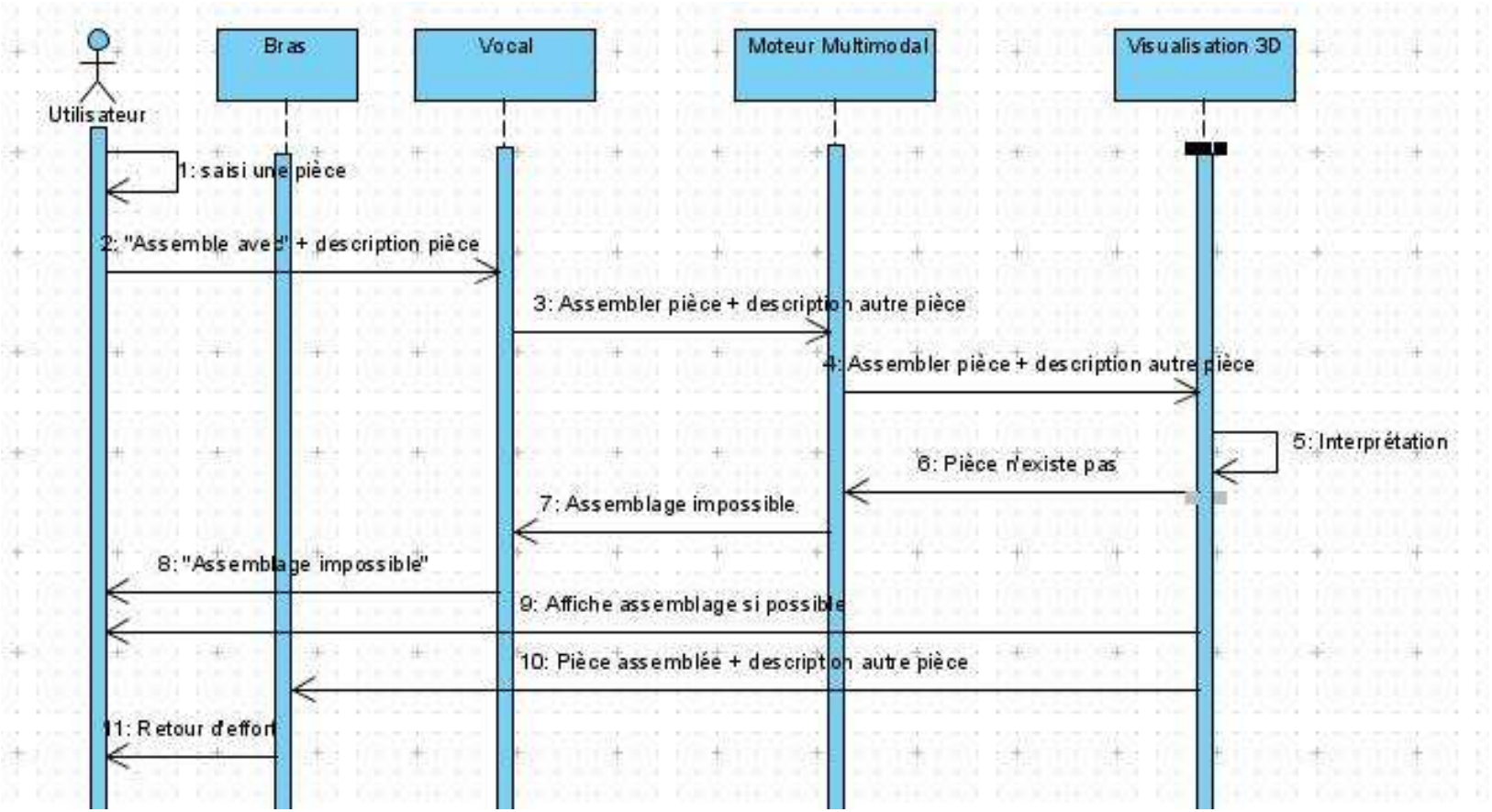


Diagramme de séquence 4 – Assembler deux pièces par la parole

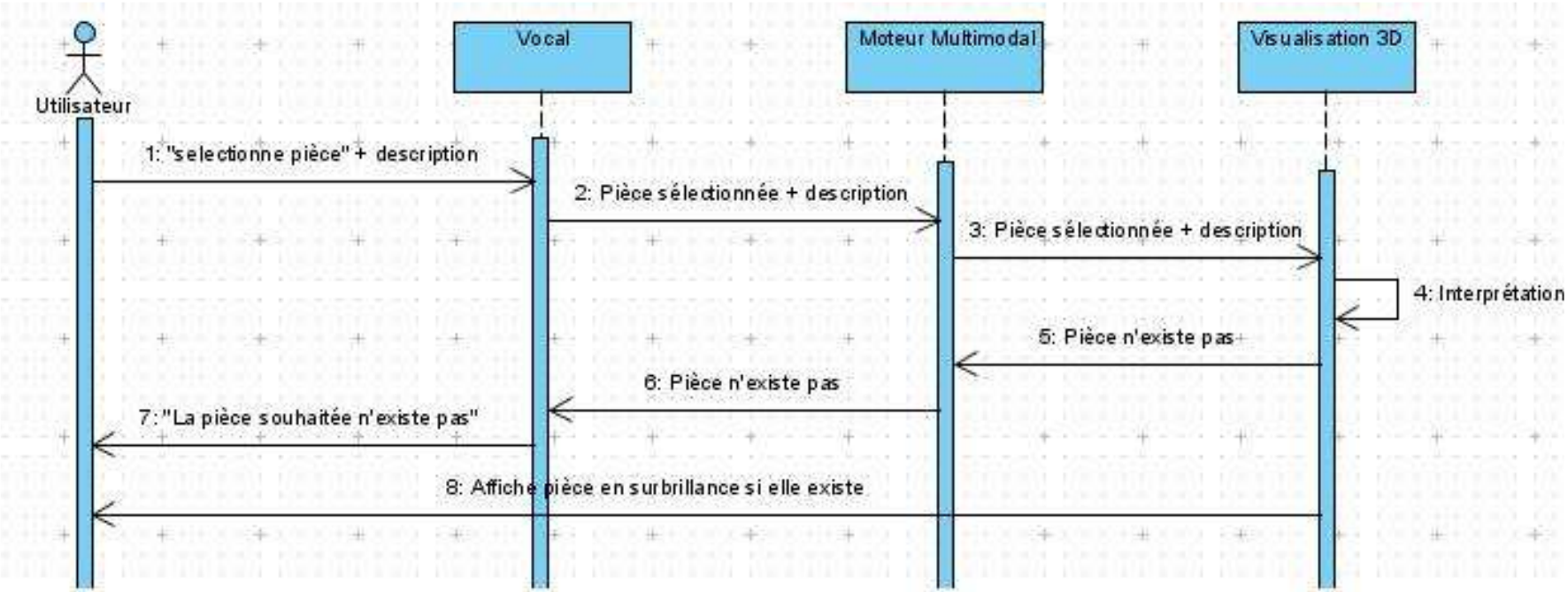


Diagramme de séquence 5 – Désignation d'une pièce par la parole

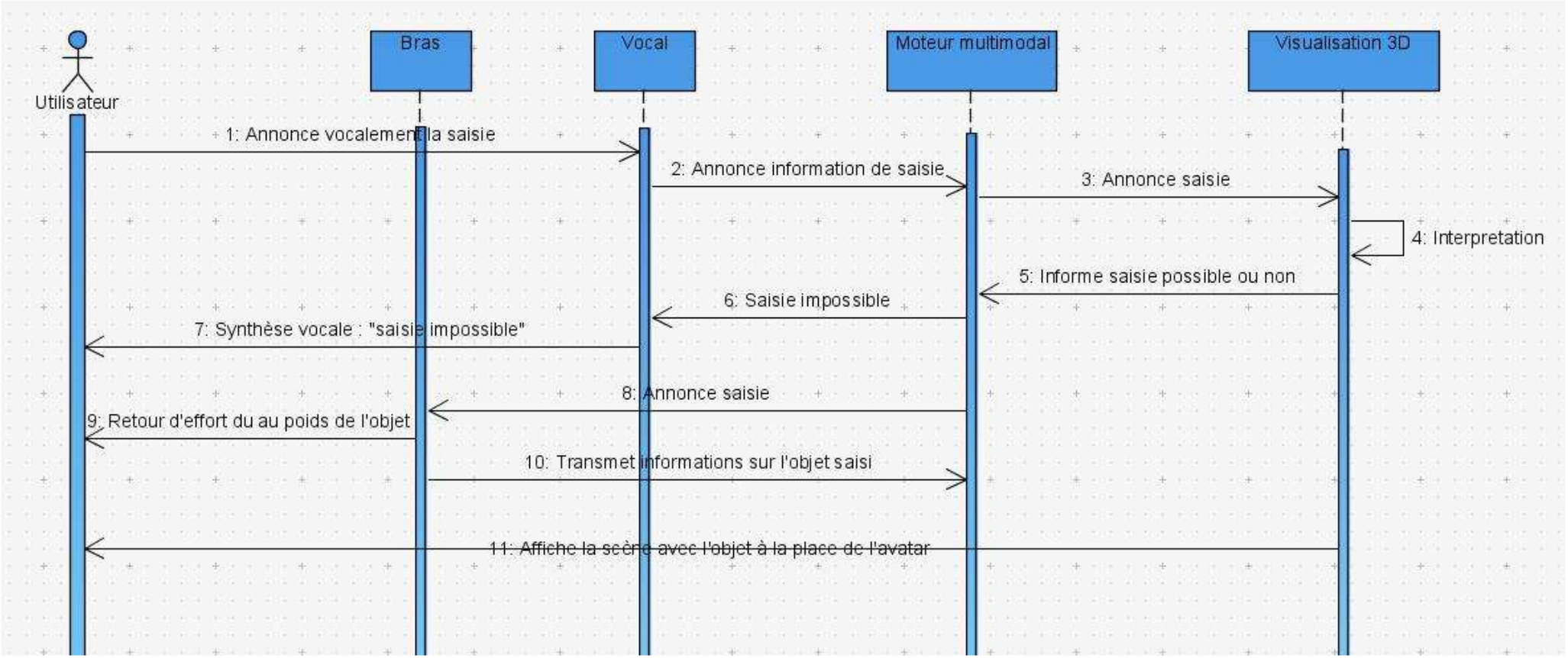


Diagramme de séquence 6 – Saisir un objet par la parole

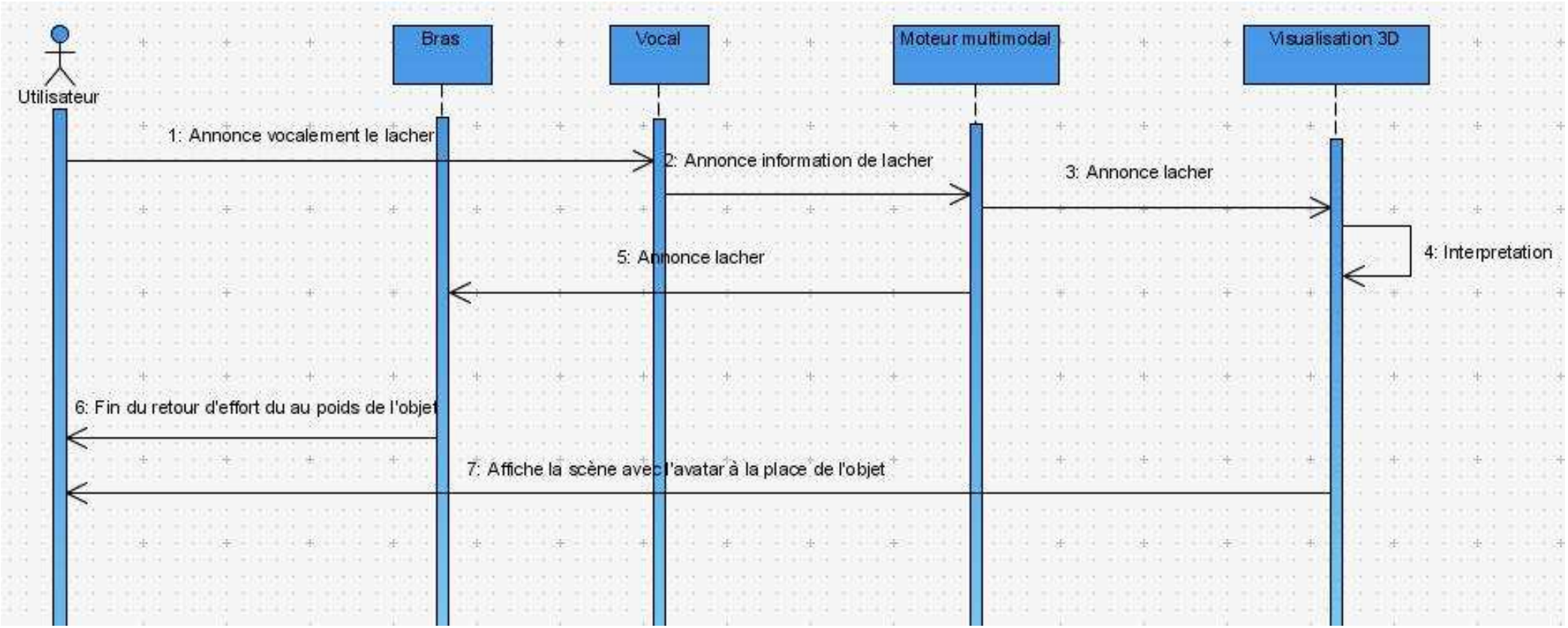


Diagramme de séquence 7 – Lâcher une pièce par la parole

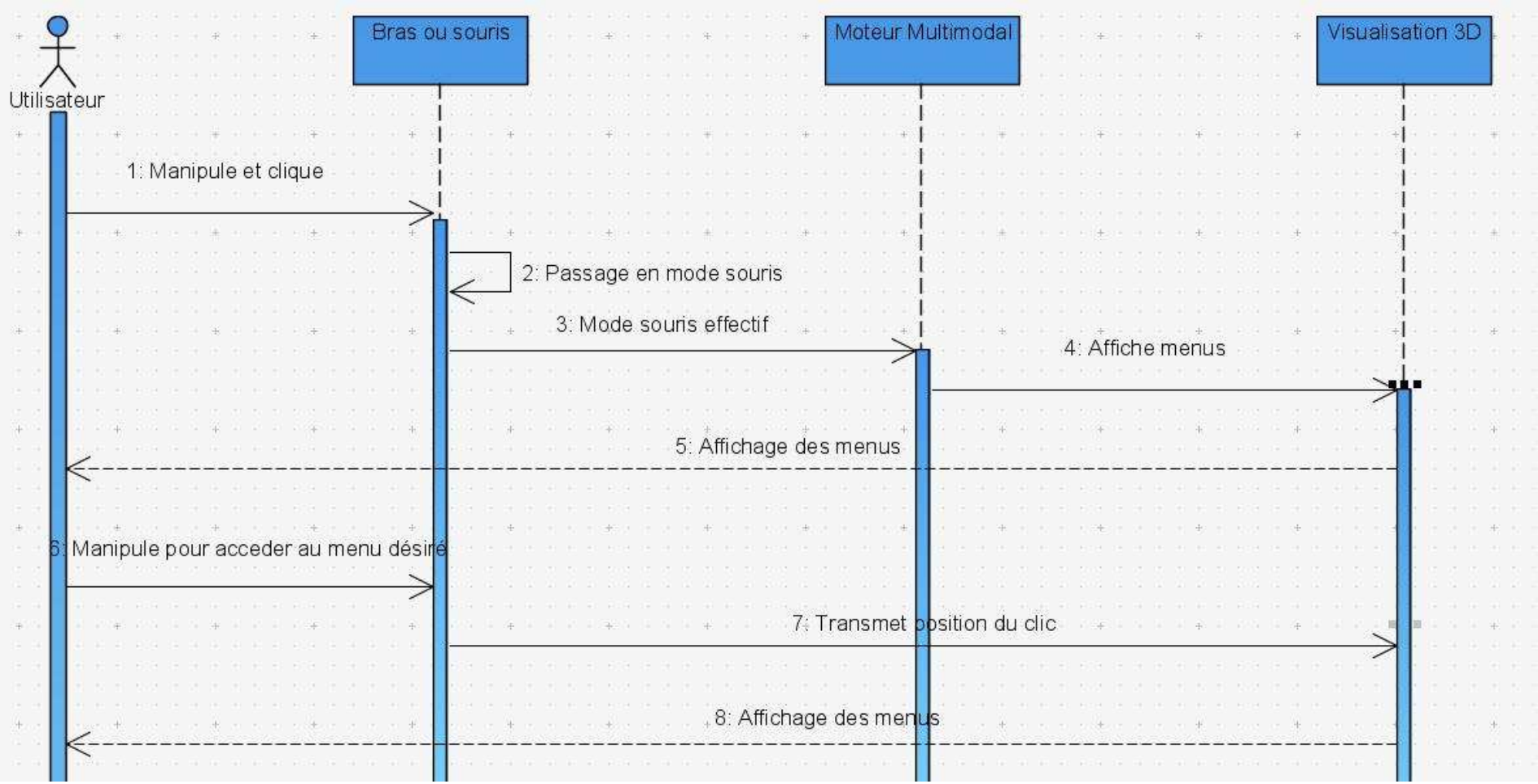


Diagramme de séquence 8 – Accéder aux menus par le bras haptique

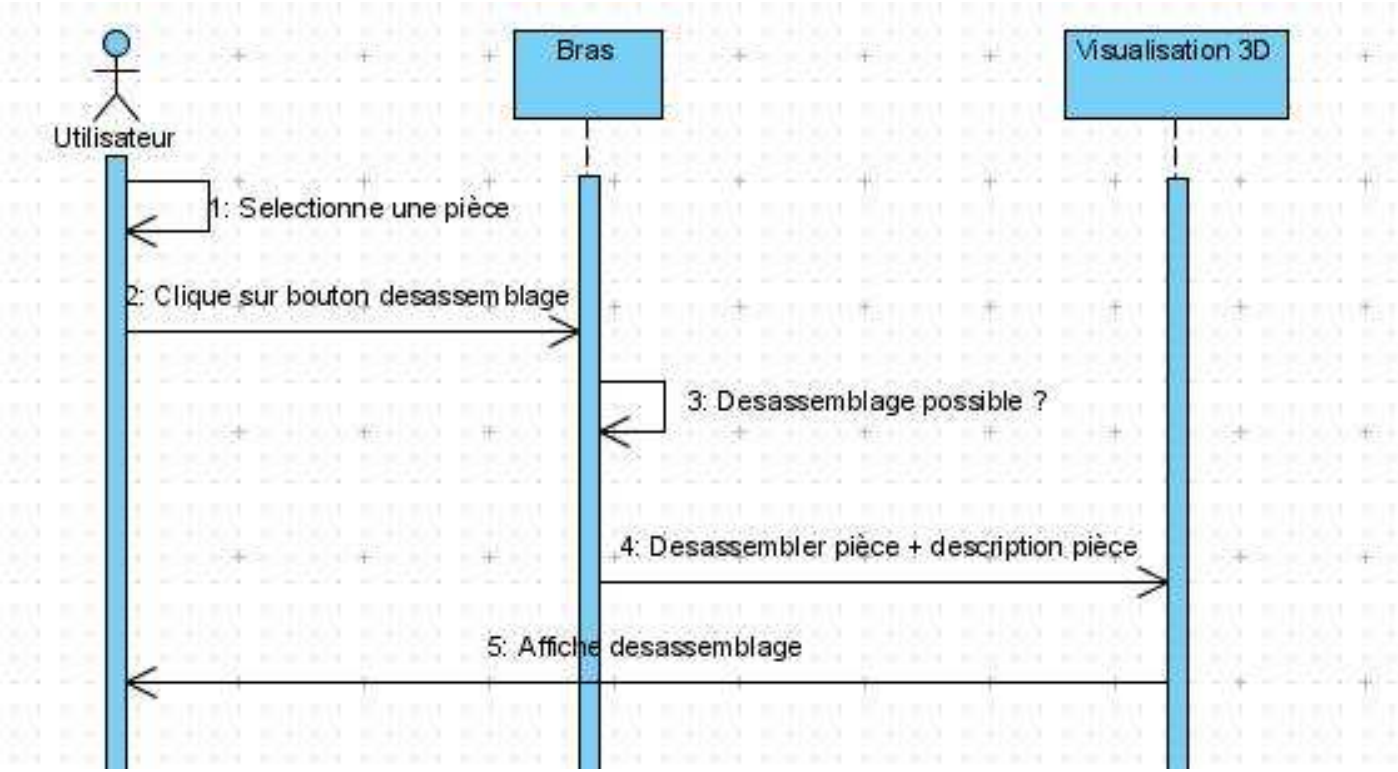


Diagramme de séquence 9 – Désassembler deux pièces par le bras haptique

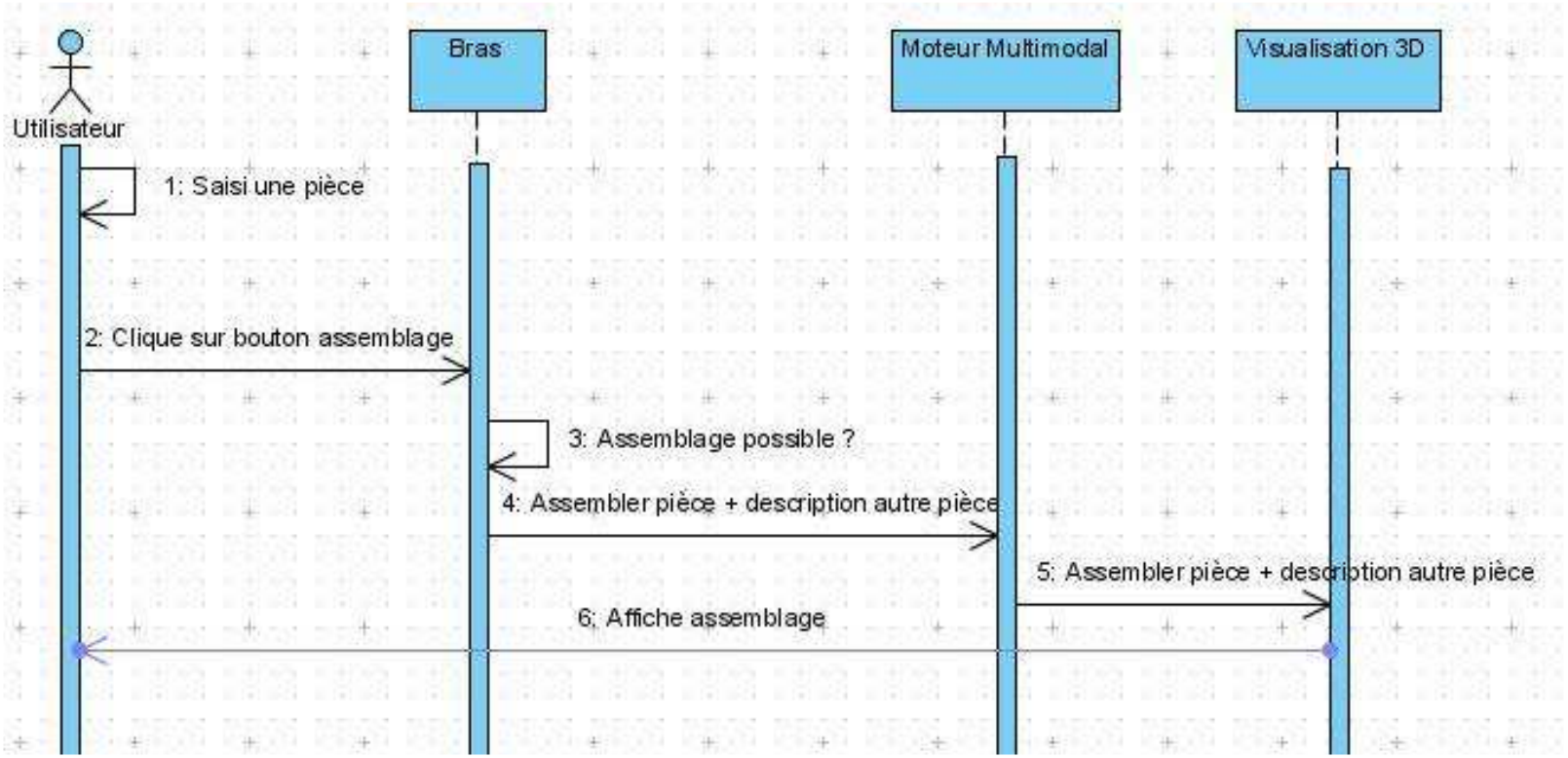


Diagramme de séquence 10 – Assembler deux pièces par le bras haptique

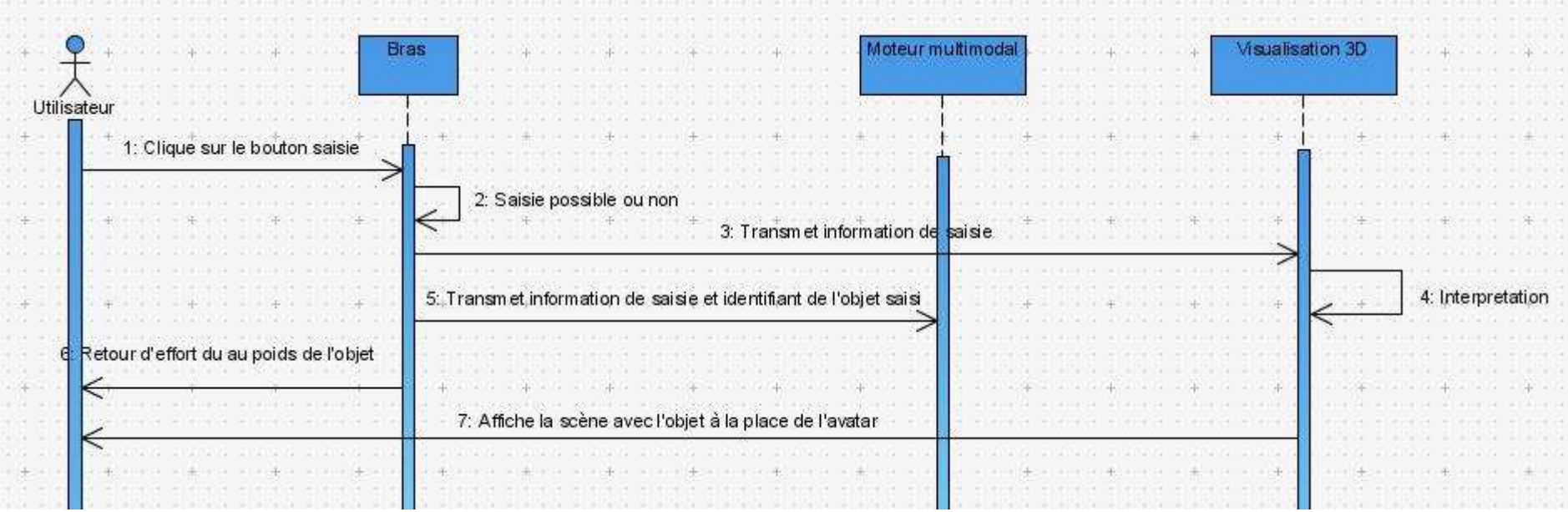


Diagramme de séquence 11 – Saisir une pièce par le bras haptique

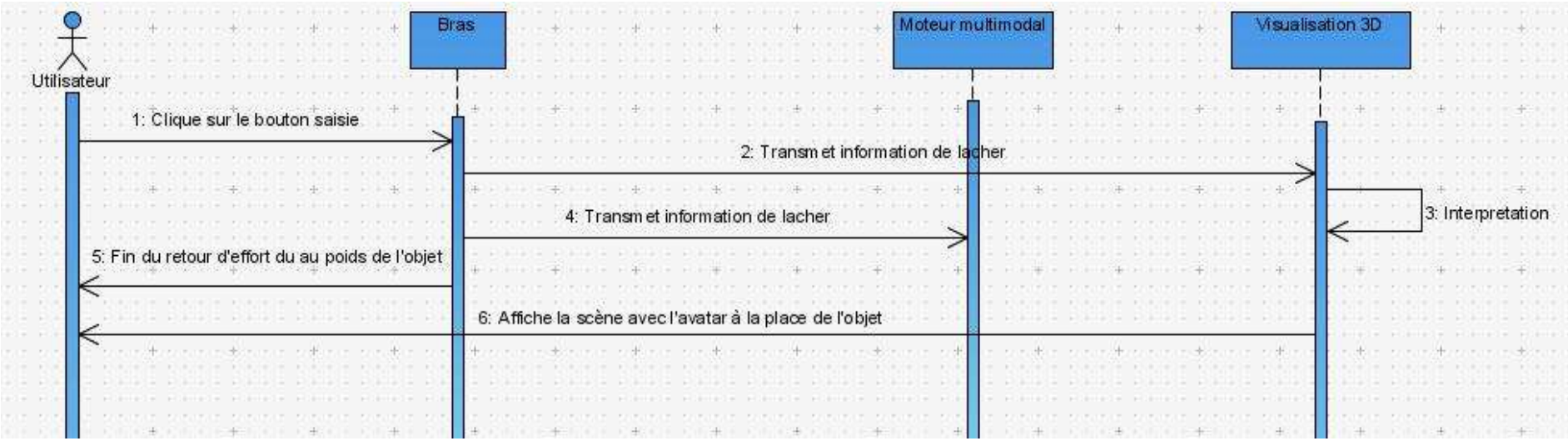


Diagramme de séquence 12 – Lâcher une pièce par le bras haptique

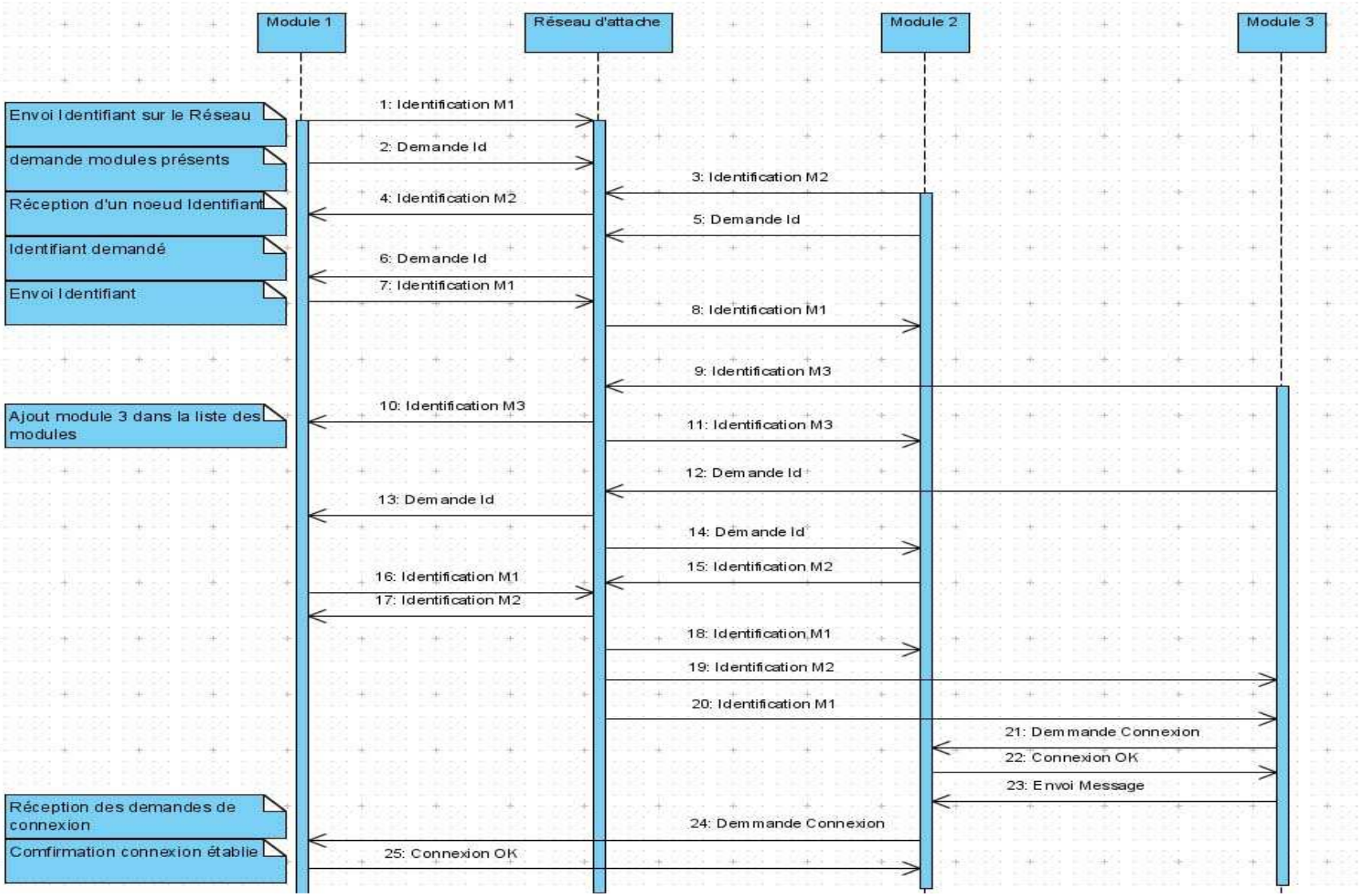


Diagramme de séquence 13 – Exemple de communication entre modules via le réseau

2. Moteur Multimodal

2.1. Présentation

Le but du moteur de fusion multimodal est de permettre à l'utilisateur de réaliser ses différentes tâches le plus naturellement possible. L'utilisateur peut se servir des diverses modalités sans avoir une grande connaissance du système. La fusion multimodale met alors en relation les actions de l'utilisateur avec le noyau fonctionnel de l'application (ou de la librairie). La fusion crée le lien entre les modalités et les fonctionnalités du système.
La présence de plusieurs modes d'interaction pose certains problèmes, en particulier car le système ne connaît pas l'ordre ou la manière dont les modalités vont être utilisées. La présence de modalités concurrentes ou complémentaires impose au minimum qu'un protocole soit proposé à l'utilisateur. Nous avons choisi que le système analyse le comportement de l'utilisateur sur chacune des modalités pour construire une représentation de ses intentions (commande pour le noyau fonctionnel). Ce type de fonctionnement correspond au modèle ARCH.

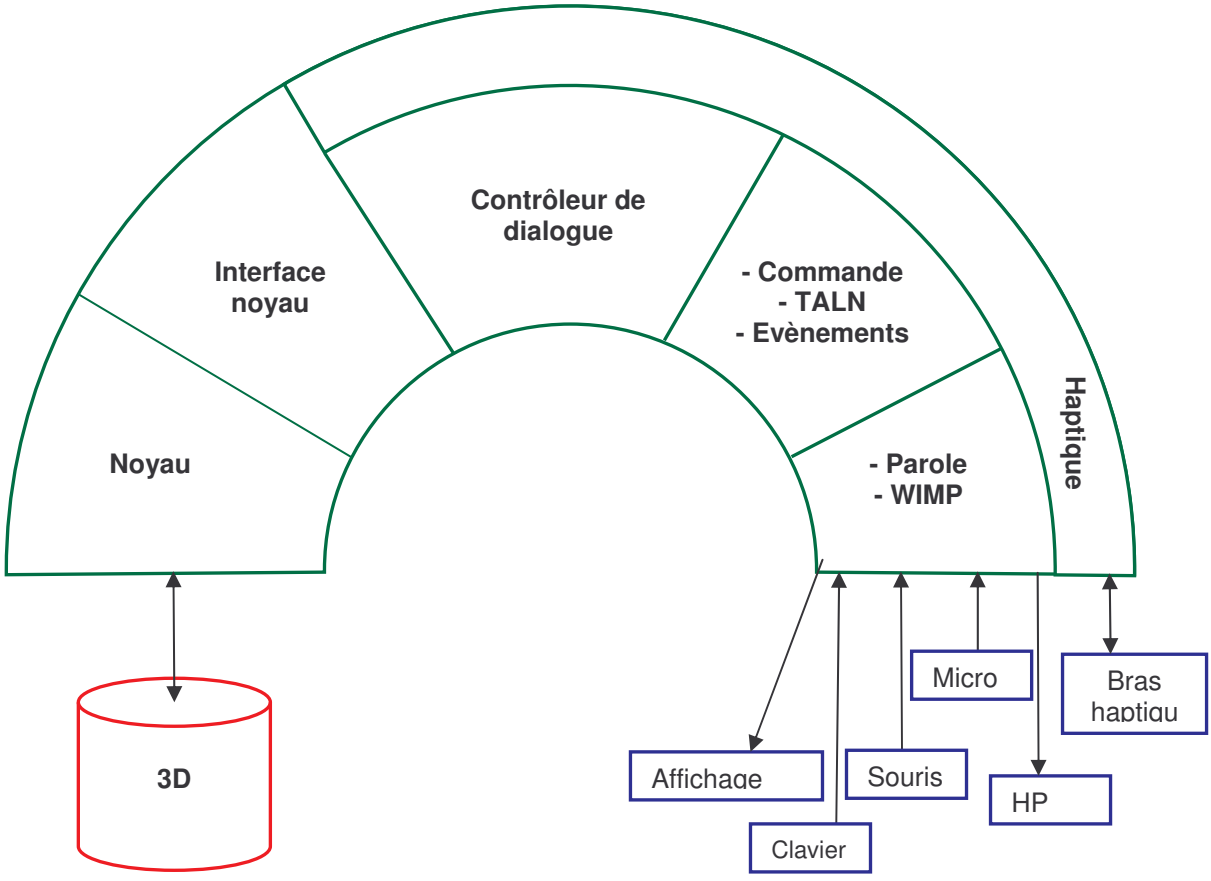


Figure 3 - Modèle de ARCH appliqué au noyau multimodal de MoveIT

Dans le modèle ARCH, chaque modalité possède une interface d'analyse et d'interprétation, ensuite toutes les informations interprétées sont confrontées entre elles par le contrôleur de dialogue. Ce dernier crée une commande qui représente les actions de l'utilisateur sur les périphériques et utilise son interface avec le noyau fonctionnel pour agir sur l'application. Le contrôleur de dialogue récupère également des informations qu'il doit redistribuer aux modalités concernées, l'affichage et la synthèse vocale principalement. Les échanges d'informations dans l'arche se font dans les deux sens, mais ils sont analysés et distribués par le contrôleur de dialogue qui est le seul composant à connaître les extrémités de l'architecture, qui sont le noyau fonctionnel et l'interface homme machine.
Dans le cadre de la librairie MoveIT, nous ne pouvons pas intégrer la modalité haptique directement dans le modèle de ARCH. L'haptique requiert une interface particulière avec le noyau fonctionnel de l'application (ici la base d'objet 3D). Ceci s'explique par le fait que la modalité haptique est fortement liée au rendu de la scène et que le contrôleur de dialogue n'a pas sa place dans la boucle [haptique + vision]. Certains évènements produits par le bras haptique passeront quand même par le contrôleur de dialogue.

2.2. Choix de conception

Pour la conception de notre moteur multimodal nous n'avions pas à notre disposition une librairie contenant tous les outils nécessaires à la mise en place d'un moteur de fusion. Nous nous sommes donc documentés (cours et documents de recherches) afin de chercher la solution la plus adaptée au PGE.
Voici les principaux éléments que nous avons choisi de mettre en avant pour décrire le moteur de fusion que nous souhaitons mettre en place :

→ Situation du moteur de fusion dans l'espace MSM (Multi-Sensori-Moteur)
→ Stratégies mises en œuvre lors de la fusion.
→ Présence ou non d'une modalité dominante.
→ Coopération entre modalité.
→ Gestion du contexte d'utilisation.

2.2.1. Situation dans l'espace MSM

Le moteur de fusion doit permettre à l'utilisateur de réaliser une tâche en utilisant plusieurs outils d'interactions en même temps. Il existe principalement trois types de multimodalité :

- La multimodalité exclusive (une modalité à la fois pour une seule tâche).

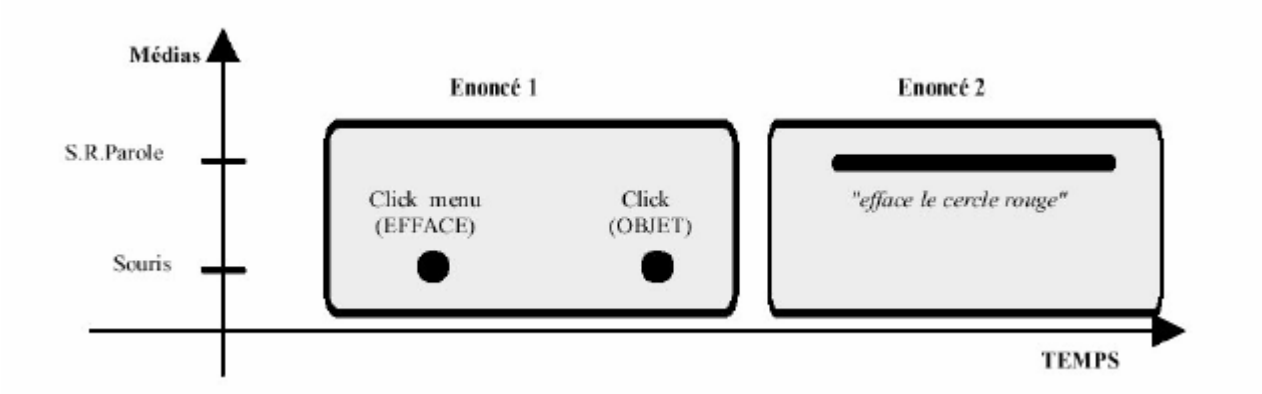


Figure 4 - Exemple d'énoncé exclusif

- La multimodalité alternée (deux modalités déphasées pour une seule tâche).

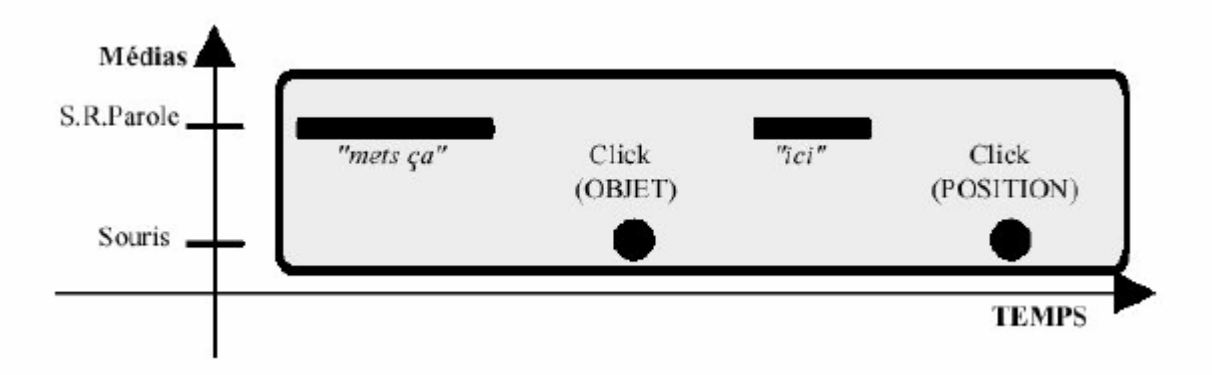


Figure 5 - Exemple d'énoncé alterné

- La multimodalité synergique (deux modalités en même temps pour une seule tâche).

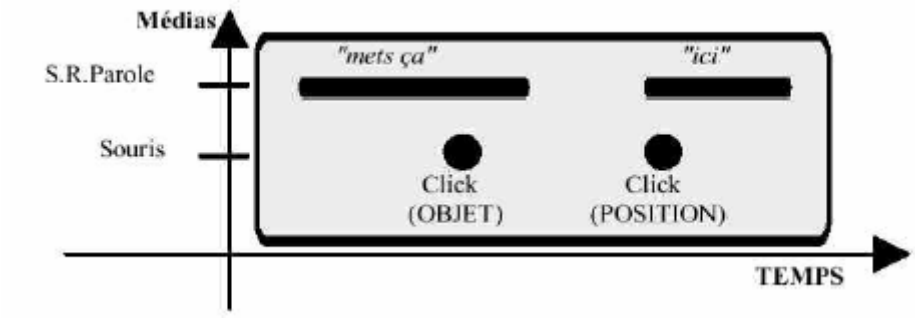


Figure 6 - Exemple d'énoncé synergique

Nous avons donc choisi de mettre en place un moteur de fusion qui permette de comprendre les énoncés synergiques. Cependant, un moteur de fusion capable de comprendre un énoncé synergique est capable de comprendre les énoncés alternés et exclusifs. Ce sont des cas dégradés d'énoncés synergiques.

2.2.2. Stratégie de fusion

La fusion de modalité nécessite l'utilisation de certaines règles afin de savoir quand et comment les informations fournies par les différentes modalités pourront être fusionnées. Nous avons choisi de baser la fusion des modalités sur les principes suivants :

- Proximité temporelle des informations :

Les événements issus de deux modalités complémentaires doivent être proches temporellement. La fusion de deux modalités ne pourra se faire que dans cet intervalle de proximité temporelle. Nous avons choisi de figer cet intervalle : cette décision est basée sur l'hypothèse que l'utilisateur connaît par avance les modalités qu'il va utiliser pour accomplir la commande. Nous estimons que ce paramètre devrait être d'une seconde, mais il pourra être vérifié et adapté qu'après quelques expérimentations.

- Compatibilité sémantique des informations :

Les informations envoyées par les modalités dans cet intervalle de temps ne seront fusionnées que si elles sont compatibles. Par exemple, l'utilisateur prononce « saisi cette pièce » et il clique ensuite avec la souris sur un objet présent dans la scène. Les deux évènements générés sont compatibles, le premier provoque la création d'une commande de saisie d'un objet non référencé, le second fournit une référence sur un objet. La fusion des deux évènements est donc cohérente.

- Compatibilité structurelle :

Les évènements perçus par le moteur de fusion seront placés dans une structure de données particulière. L'acte de fusion passera par la comparaison de deux structures et la vérification que celles-ci sont compatibles.

2.2.3. Dominance d'une modalité

Une modalité dominante est une modalité qui serait à l'initiative de toutes les tâches que l'utilisateur souhaite réaliser. Nous pourrions considérer que dans la librairie MOVEIT, toutes les actions menées par l'utilisateur seraient dépendantes d'une action sur le bras haptique, cependant la librairie doit être générique. Il ne peut donc pas y avoir une modalité dominante. Toute modalité doit pouvoir être à l'origine d'une action de l'utilisateur final sur le système et toute modalité d'entrée doit pouvoir être remplacée ou supprimée.

2.2.4. Liens entre modalités

Voici les liens entres les différentes modalités. Nous nous sommes basés sur les définitions CARE.

- Complémentarité (C) : les deux modalités se complètent pour permettre à l'utilisateur de réaliser une tâche.
- Assignation (A) : une modalité peut être assignée à un rôle précis vis-à-vis d'une tâche particulière ou du fonctionnement de l'application.

- Redondance (R) : la même information peut être produite par plusieurs modalités en même temps.
- Equivalence (E) : deux modalités peuvent produire la même information.

Modalités d'entrée :

	Bras haptique	Parole	Souris	Clavier
Bras haptique		C et E	E	E
Parole			C et E	C et E
Souris				C
Clavier				

Modalités de sortie :

	Synthèse vocale	Affichage 2D	Visualisation 3D	Bras haptique
Synthèse vocale		R et E	C	C
Affichage 2D			C	C
Visualisation 3D				C
Bras haptique				

En entrée du système, quasiment toutes les modalités sont équivalentes. Ceci permet de laisser l'utilisateur libre de choisir sa technique d'interaction sans reconfigurer le système. En revanche la fission d'information se basera sur la coopérativité des modalités de sortie pour choisir le media adapté à la production d'information vers l'utilisateur.

2.2.5. Gestion du contexte d'utilisation

La présence de la parole et le fait que ce soit une modalité équivalente en entrée aux autres modalités nous impose de pouvoir résoudre les références indirectes. Pour cela nous avons choisi de garder en mémoire les dernières commandes effectuées par l'utilisateur sur le monde afin de pouvoir les utiliser comme référence. Par exemple quand l'utilisateur prononce : « crée la même pièce », le moteur doit savoir de quelle pièce il s'agit. Les informations contenues dans la mémoire sont directement utilisables par le moteur de fusion

2.3. Choix d'implémentation

2.3.1. Langage

Nous avons choisi d'utiliser le langage Java pour implémenter le moteur de fusion. Ceci représente deux avantages : la simplicité de mise en œuvre et la garantie de la portabilité.

2.3.2. Généricité de la librairie

La généricité de la librairie fait partie du cahier des charges. Ceci pose un problème pour la conception du moteur de fusion. La fusion de modalité requiert énormément d'informations sur les modalités mises en jeu et sur les tâches réalisables. Nous nous proposons de rendre notre moteur suffisamment paramétrable pour qu'il soit générique et facilement extensible.

La généricité est obtenue de deux manières :

2.3.2.1. Répartition de l'analyse syntaxique

Chaque module, étant en interaction avec le moteur de fusion/fission multimodal, connaît la syntaxe utilisée par le moteur multimodal. Cette syntaxe sera attendue en entrée du moteur de fusion et sera fournie en sortie à destination des autres modules. Chaque module devra être capable de parler dans un langage compréhensible par le moteur de fusion. De même qu'ils devront être capables de traduire les commandes du moteur de fusion dans leur base de connaissance.

2.3.2.2. Utilise les règles de fonctionnement spécifiques à l'application

Chaque application possède son propre fonctionnement, le moteur doit donc être spécialisé afin de savoir quelles sont les actions à réaliser en réponse aux évènements perçus. Les actions réalisables par chacun des modules doivent être détaillées. La spécialisation de moteur passe par la description des différentes fonctionnalités offertes par chacun des acteurs présents dans le système. Ainsi le moteur de fusion connaît les compétences de chacun et sait comment les utiliser.

2.4. Interface externe du module de fusion/fission

Le module de fusion utilise les interfaces du module réseau pour dialoguer avec les autres modules. Il ne possède que deux interfaces normalisées : une interface d'entrée et une interface de sortie. Voici une représentation des interfaces du MF/F.

Figure 7 - Interface du moteur Fusion/Fission

Le moteur multimodal va permettre à l'utilisateur d'utiliser ses différents moyens d'interactions afin d'agir sur le monde ou l'interface graphique.
Les entrées du Moteur de Fusion/Fission (MF/F) sont donc des évènements ou des commandes générés par chacun des acteurs présents dans le système.
Le MF/F peut recevoir une commande d'erreur ou une commande erronée. Dans ce cas on génèrera en sortie le code erreur général associé. Ce code d'erreur sera interprété et diffusé par la ou les modalités de sortie concernées. Ceci implique que toutes les modalités de sortie ont le même fichier d'erreur qu'elles interprètent à leur manière. Le MF/F peut également récupérer la réponse d'une requête sur la base de données de gestion du monde 3D et ensuite envoyer cette réponse à la ou les modalités de sortie concernées.

Les sorties du MF/F sont des actions ou des commandes à transmettre aux différents acteurs à l'écoute du MF/F, et concernés par la commande.

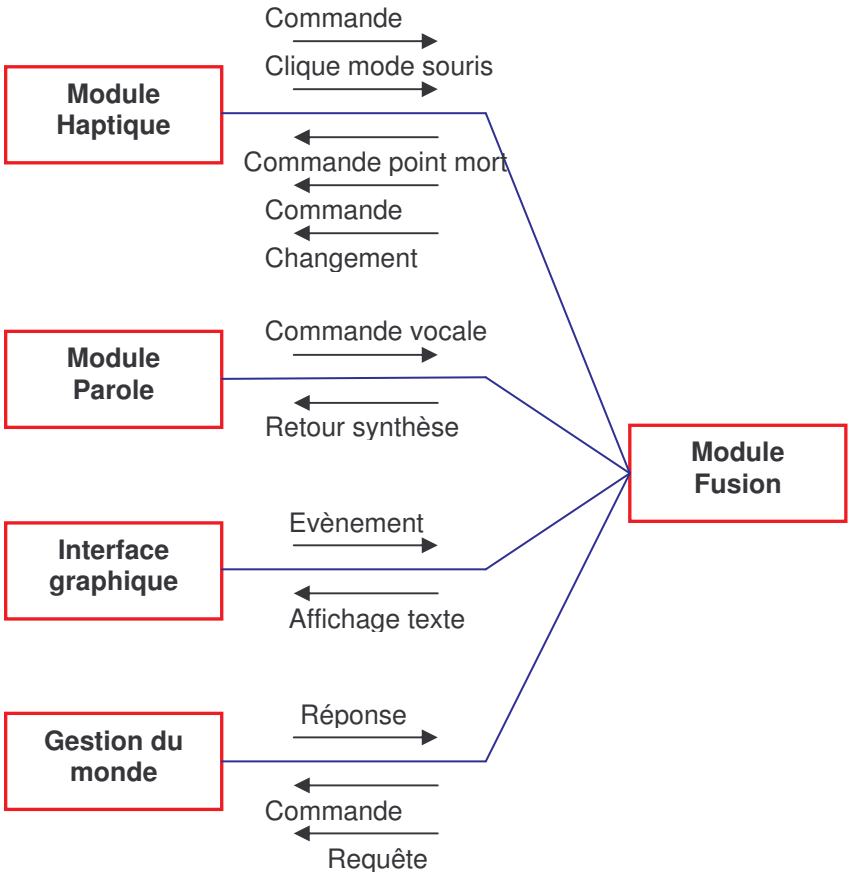


Figure 8 - Diagramme de collaboration du Moteur Multimodal

2.5. Modèle de commande

Les commandes qui transitent entre les modules sont de trois catégories :

- Commandes d'action : nom_acteur + nom_action + {ensemble d'attributs}
- Commandes d'information : nom_acteur + type_info + {ensemble d'attributs}
- Commandes d'erreur : nom_acteur + code_erreur + {ensemble d'attributs}

Toutes les commandes doivent contenir le nom de l'émetteur afin de pouvoir lui répondre. Le deuxième paramètre permettra d'identifier de quel type de commande il s'agit. Les attributs dépendent directement de la commande.

2.6. Présentation du fonctionnement interne du module

Voici une représentation schématique de la chaîne de traitement de fusion/fission.

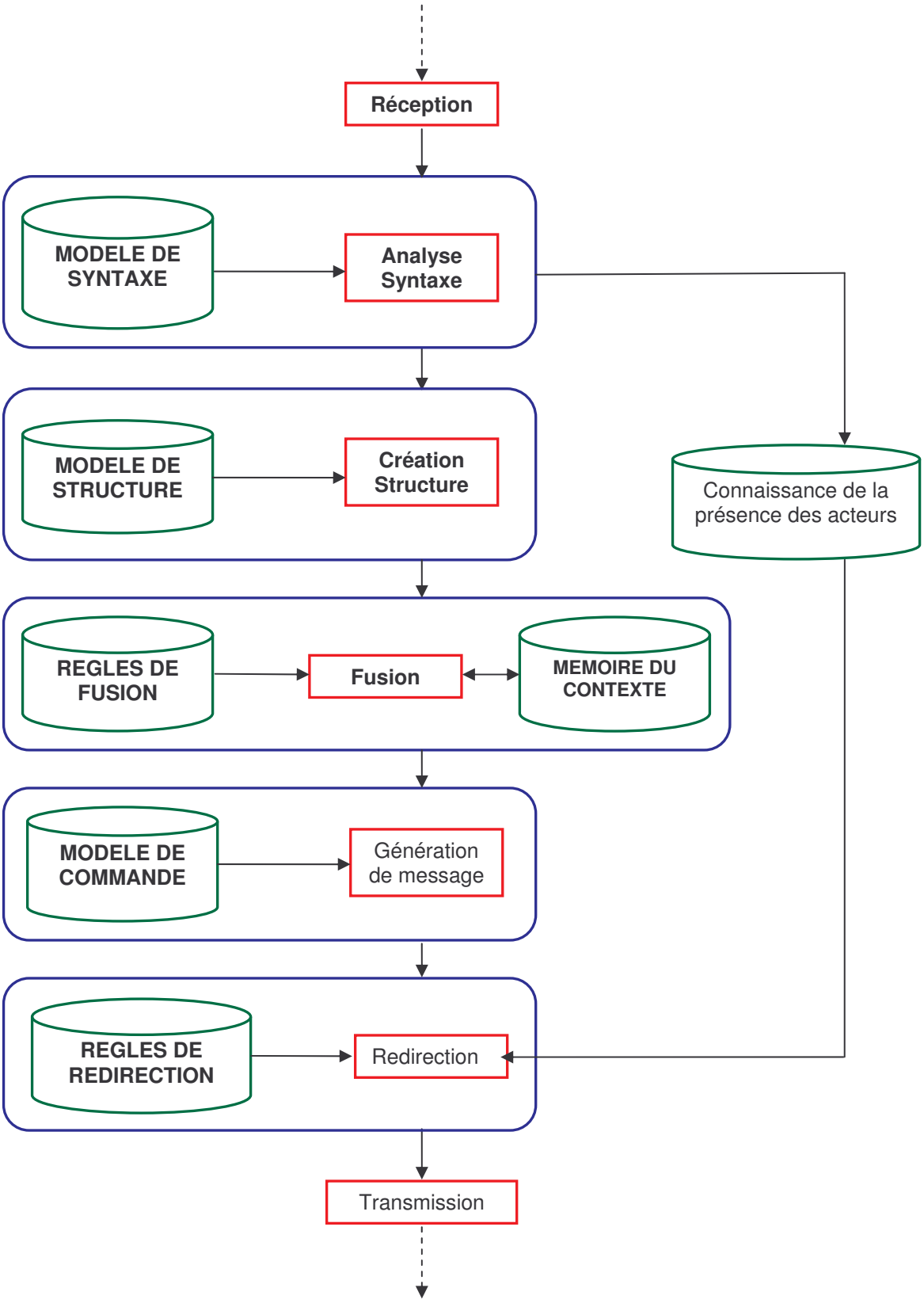


Figure 9 - Décomposition du processus de fusion/fission

Description des différentes étapes de traitement :

- **Réception** : s'occupe de la réception des messages envoyés par les autres modules, par l'intermédiaire du module de communication. Les types de messages (décris en détail plus bas) sont : commande, erreur ou déclaration de présence.

  Entrée : message générique encapsulé par le module de communication
  Sortie : message générique

- **Analyse Syntaxe** : consiste à analyser les messages pour vérifier qu'ils sont compatibles avec les messages prévus par le système. L'ensemble des messages permis par le système sont définis dans un modèle de syntaxe. Les agents du système peuvent se déclarer présents, et sont ajoutés à une base de connaissance de la présence des acteurs.

  Entrée : message générique
  Sortie : message générique validé ou information sur un agent

- **Création Structure** : correspond à la création de la structure de données attribuée à la tache que l'utilisateur veut réaliser. Le modèle de structure associé à chaque message permettra la fusion entre deux messages.

  Entrée : message générique
  Sortie : structure représentant le message

- **Fusion** : consiste à prendre en compte plusieurs messages pour une même commande. La fusion sera réalisée à l'aide d'un système de creuset et utilisera des règles de fonctionnement (la fenêtre temporelle et les règles de fusion des messages).

  Entrée : structure incomplète ou invalide
  Sortie : structure complète et valide représentant une commande ou une erreur

- **Génération de messages** : C'est la première étape de la fission. Elle consiste à créer les messages qui seront envoyés à l'utilisateur ou aux autres modules.

  Entrée : structure complète représentant une commande ou une erreur
  Sortie : message de commande ou d'erreur

- **Redirection** : indique à quels modules envoyer le message, et ceci grâce à des règles de redirection indépendantes du système. Ces règles doivent prendre en compte l'absence ou la présence des modalités sans quoi le message ne pourra pas être transmis. La présence des différentes modalités est inscrite depuis le module d'analyse syntaxe et peut être vérifiée à ce stade.

  Entrée : message de commande ou d'erreur
  Sortie : message avec information sur le destinataire

- **Transmission** : l'envoi des commandes se fera de manière transparente par le biais de l'interface réseau.

  Entrée : message suivi et son destinataire
  Sortie : message générique encapsulé par le module de communication

3. GUI

3.1. Description du système

3.1.1. Fonctionnalités principales :

Le module GUI(Graphical User Interface) fournit une fenêtre de visualisation de la scène 3D en plein écran. Il offre aussi la possibilité de créer des dispositifs d'interaction Homme/Machine de type WIMP :

- Une **MenuBar** (Barre de menu déroulant) fixe se trouvant dans le coin supérieur gauche de l'écran. Il sera possible d'ajouter (respectivement supprimer) autant de menus et sous-menus que voulu de façon dynamique (c'est à dire pendant l'exécution de l'application).
- Une **ToolKit** (boite à outil) flottante. Possibilité d'ajouter (respectivement supprimer) un ensemble de boutons dynamiquement.
- Un **terminal** fixe se trouvant dans le coin inférieur gauche de l'écran.

Il sera aussi possible de cacher/montrer les dispositifs GUI sans déformer l'espace d'affichage 3D.
Le module GUI gèrera un ensemble de variables d'état qui pourront être mis à jour par l'intermédiaire de la ToolKit ou de fenêtre pop-up.

3.1.2. Prototype :

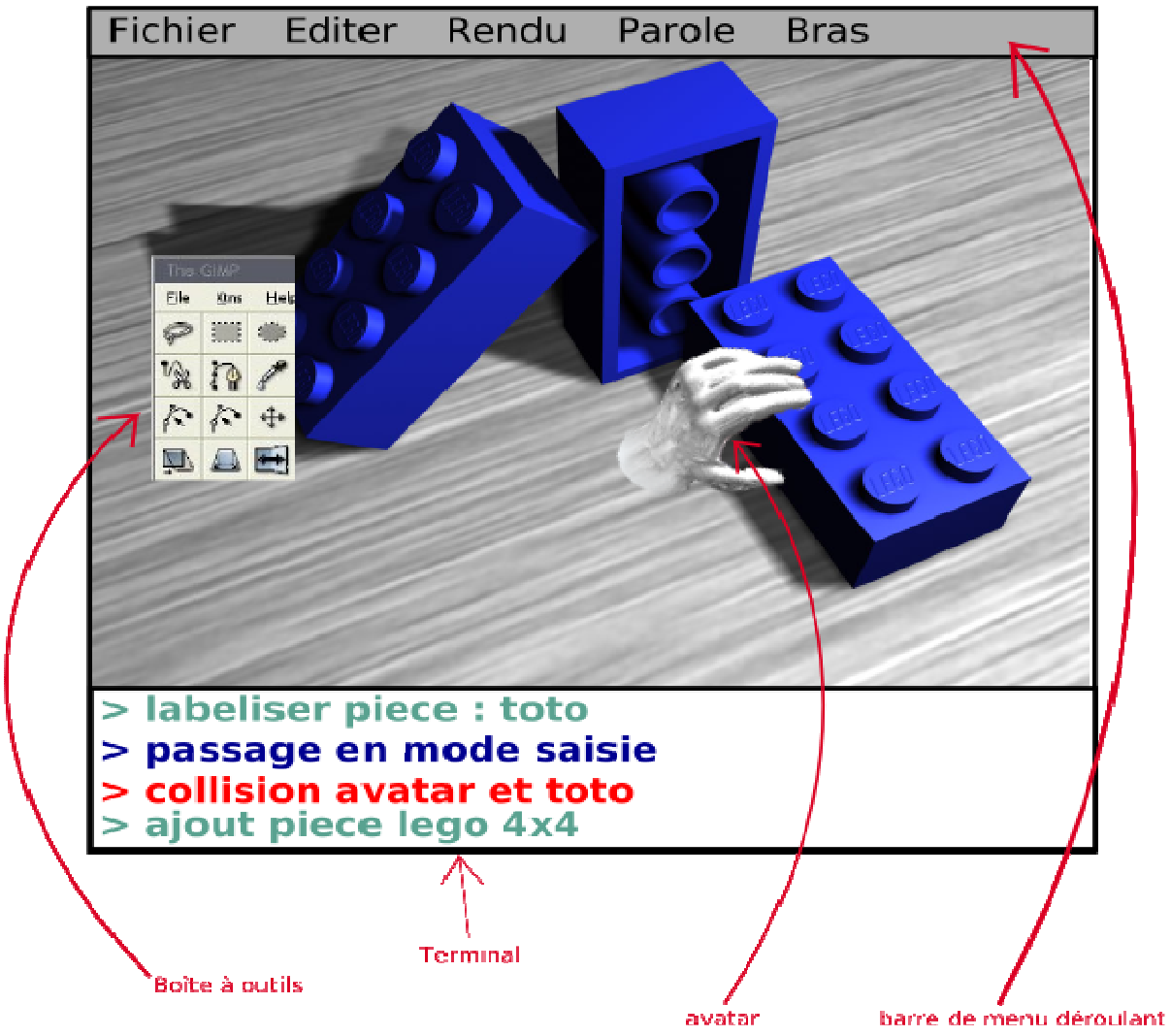


Figure 10 - Prototype de la GUI

3.1.3. Environnement :

Le module GUI sera développé en C++ et sera portable sur n'importe quel système (Windows 2000/XP, Linux/Unix, MacOS) équipé des librairies :

- ***Gtkmm 2.8*** : permet de créer des interfaces graphiques utilisateurs. Elle a été choisie car elle est portable, gratuite (Licence LGPL = Lesser Gnu Public License), et orientée objet (C++).
- ***gtkglextmm 1.0.1*** : extension de ***Gtkmm*** permettant d'intégrer du code openGL dans une fenêtre ***Gtkmm***.

3.2. Interfaces externes :

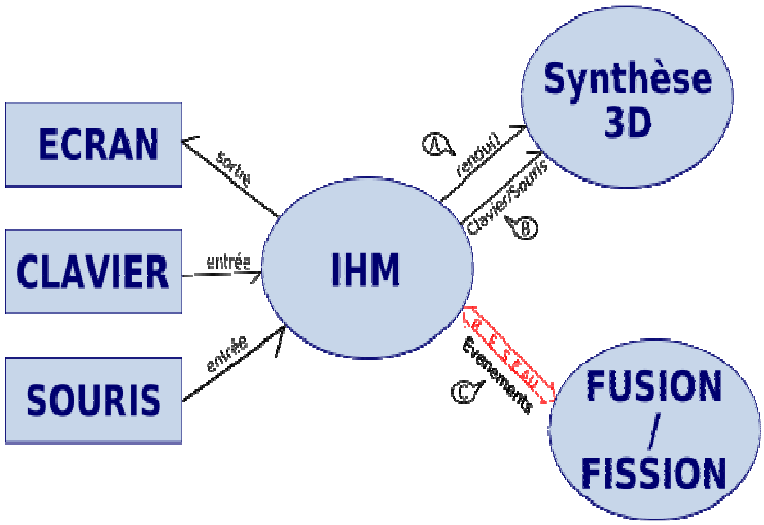


Figure 11 - Interfaces externes de la GUI

A – La GUI met à jour la fenêtre de visualisation 3D à 25Hz (ou 100Hz si vision stéréoscopique) en appelant une méthode du module synthèse 3D.

B – Certains évènements Clavier/Souris sont propagés au module Synthèse 3D (ex : picking, mode déplacement clavier/souris).

C - Le module fusion est avertit des différentes interactions sur la GUI :

- Sélection d'un menu/item.
- Appui bouton raccourci.
- Picking : Sélection d'un objet 3D dans la scène par la souris.
- Cacher/Afficher les dispositifs GUI.
- Modification de variables d'état.

- Ouverture une fenêtre pop-up de saisie d'information.

Le module Fusion peut avertir le module GUI en envoyant des évènements :

- Sélection d'un menu/item.
- Message à afficher dans le terminal.
- Ouvrir une fenêtre pop-up de saisie d'information.
- Afficher/Cacher des dispositifs GUI.
- Mise à jour de variables d'état.
- Ajout d'un menu correspondant à un module (ex : module reconnaissance vocale).
- Ajout d'items dans la ToolKit.

4. Visualisation 3D

4.1. Présentation

4.1.1. Architecture logicielle interne

Cette architecture est la même pour la librairie et l'application. La base de donnée « Init » est propre et différente à chaque application. Elle ne fait pas partie de la librairie.

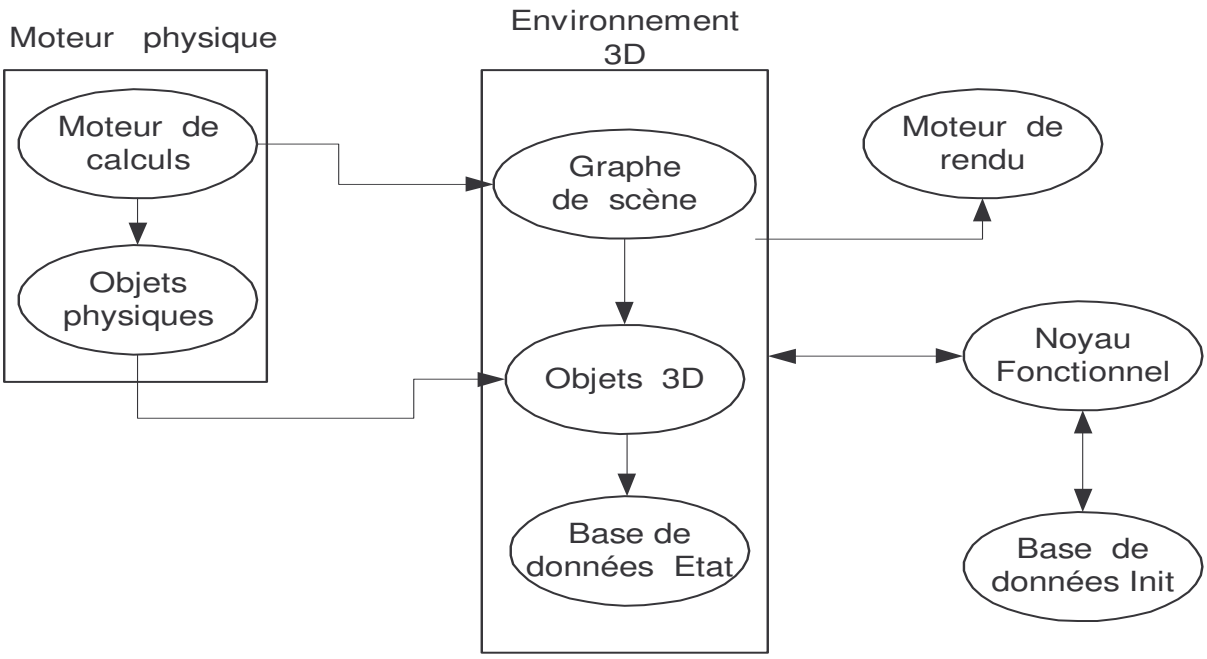


Figure 12 - "Architecture logicielle interne" du module de visualisation

Nous allons maintenant décrire les différentes parties composant cette architecture :

- **Moteur physique :**
  → *Moteur de calculs :* chargé du calcul des forces et des collisions sur les objets présents dans la scène.
  → *Objets physiques :* composant contenant les caractéristiques physiques des objets de l'environnement. C'est sur ce composant que le moteur de calcul base ses connaissances. Il s'initialise grâce aux informations contenues dans la base de donnée « Init ».

- **Environnement 3D :**
  → *Objets 3D* : contient tous les objets 3D de la scène de l'application (décors, LEGO, camera, lumière, avatar).
  → *Graphe de scène* : définit la composition de la scène au travers de toutes les caractéristiques propres à chaque objet chargé lors de l'initialisation, comme leur position et leur orientation.
  → *Base de donnée Etat* : cette base de donnée est mise à jour à chaque requête d'information du Noyau Fonctionnel. En effet, lors d'une requête sur cette base de données, le Noyau Fonctionnel demande au module Environnement 3D de lui retourner les infos demandées. A ce moment là, le graphe de scène met à jour cette base de donnée. Les informations sont ensuite retournées au Noyau Fonctionnel.

- **Moteur de rendu :**
Il est chargé du rendu de la scène.

- **Noyau Fonctionnel :**
Ce module a deux fonctionnalités principales :
  → Envoyer toutes les informations contenues dans la base de données « Init » au module Environnement 3D afin que ce dernier puisse créer tous les objets de la scène, et remplir la base de données Etat.
  → Accepter des requêtes venant du module multimodal en lui permettant de recueillir les informations demandées, contenues dans la base de données Etat, sur les objets présents dans la scène.

- **Base de données « Init » :**
Elle répertorie toutes les caractéristiques des objets qui peuvent être utilisées par le module de visualisation 3D pour la création de ces derniers. Elle ne varie pas au cours du temps, hormis si la librairie permet d'enregistrer un objet 3D accessible lors d'une prochaine utilisation de l'application. L'architecture de cette base de données sera détaillée dans le document de Conception Détaillée.

4.1.2. Architecture matérielle interne

Le module de Visualisation 3D nécessite, au vue des méthodes qui seront employées, une carte graphique suffisamment performante. Nous préférons la marque « NVIDIA » de part son architecture mieux adaptée à l'utilisation de la librairie OpenGL.

Ex : NVIDIA GeForce 6xxx

Lors d'une future utilisation de lunettes de stéréovision, il faudra impérativement utiliser une carte graphique possédant deux sorties vidéo, et possédant une puissance de calcul suffisante.

Ex : NVIDIA Quadro xxxx

4.1.3. Résumé des fonctionnalités

- **Moteur physique :**
  → Gère la manipulation des objets (déplacement, saisie).
  → Gère les interactions entre plusieurs objets (assemblage, désassemblage, rebonds).
  → Gère les collisions avec l'environnement et les objets.
  → Gère le déplacement de l'avatar et des caméras.
  → Gère le passage du mode déplacement au mode manipulation et vice versa.

- **Environnement 3D :**
  → Gère la création et la modification des propriétés des objets.
  → Interroge la base de données.
  → Met à jour la base de données (ex : ajoute un objet labellisé).

- **Moteur de rendu :**
  → Gère les lumières, les ombres, les angles de vue, les textures.

4.2. Apports logiciels

4.2.1. Moteur de rendu

Il existe 2 standards en matière d'API de rendu graphique : DirectX et OpenGL. Nous avons décidé d'utiliser OpenGL pour des raisons de portabilité, car DirectX ne fonctionne pas sous d'autres OS que Windows.

4.2.2. Moteur physique

Il existe plusieurs moteurs physiques tels que ODE, Newton, Tokamac, Havok. Tokamac ne supporte pas d'autres OS que Windows, il a donc été rejeté. Havok est un moteur physique puissant mais non gratuit et a donc été rejeté. La librairie ODE(Open Dynamic Engine) a été choisie car elle est multi plateforme, bien diffusée et sa prise en mains semble plus aisée.

4.2.3. Langage de programmation

Nous avons choisi de programmer en langage C++ :

- Il est supporté par toutes les librairies que nous avons décidé d'intégrer.
- Le langage objet est plus approprié à la représentation d'un monde 3D.

4.3. Entrées, sorties et interfaces

4.3.1. Entrées

Le module bras envoie des informations liées à la position et l'orientation de l'avatar.
Le module multimodal fait des requêtes sur la base de données et des actions sur la scène.
La base de données envoie les réponses aux requêtes demandées. Plus précisément, elle renvoie des informations liées aux objets.
La GUI envoie les événements venant de la souris et du clavier qui concernant la zone de visualisation 3D (ex : picking). De plus, en mode dégradé, elle envoie la position et l'orientation venant de la souris et du clavier.

4.3.2. Sorties

Nous envoyons des informations au module bras concernant le graphe de scène, la matrice caméra, la modification et la création d'objets.
Nous envoyons au module multimodal des réponses à ses requêtes ainsi que des validations aux actions demandées.
Nous mettons à jour la base de données Etat. Nous envoyons les modifications sur les objets et la scène.
Nous envoyons à la GUI la projection de la scène (le rendu).

4.3.3. Interfaces

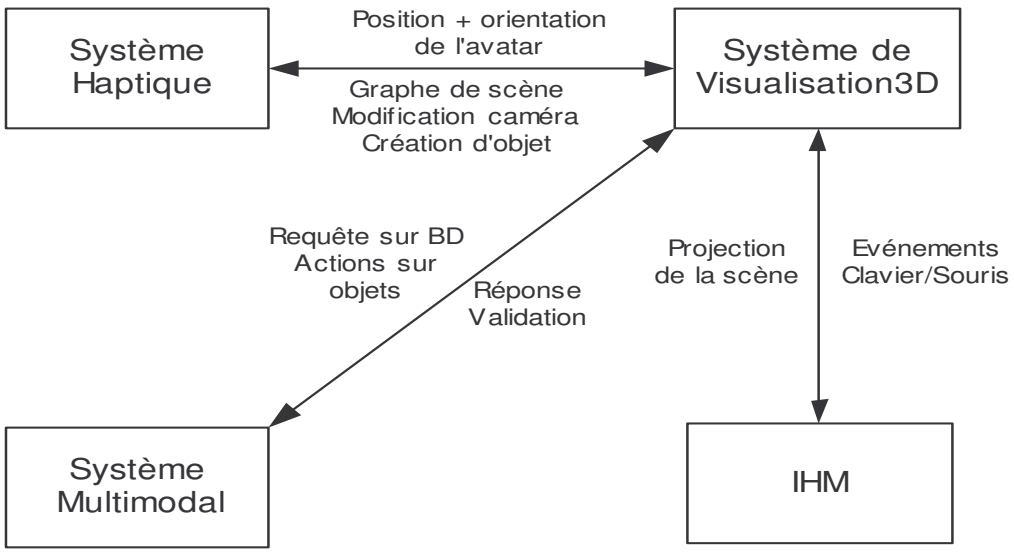


Figure 13 - Interface externe

4.4. Diagrammes de séquence

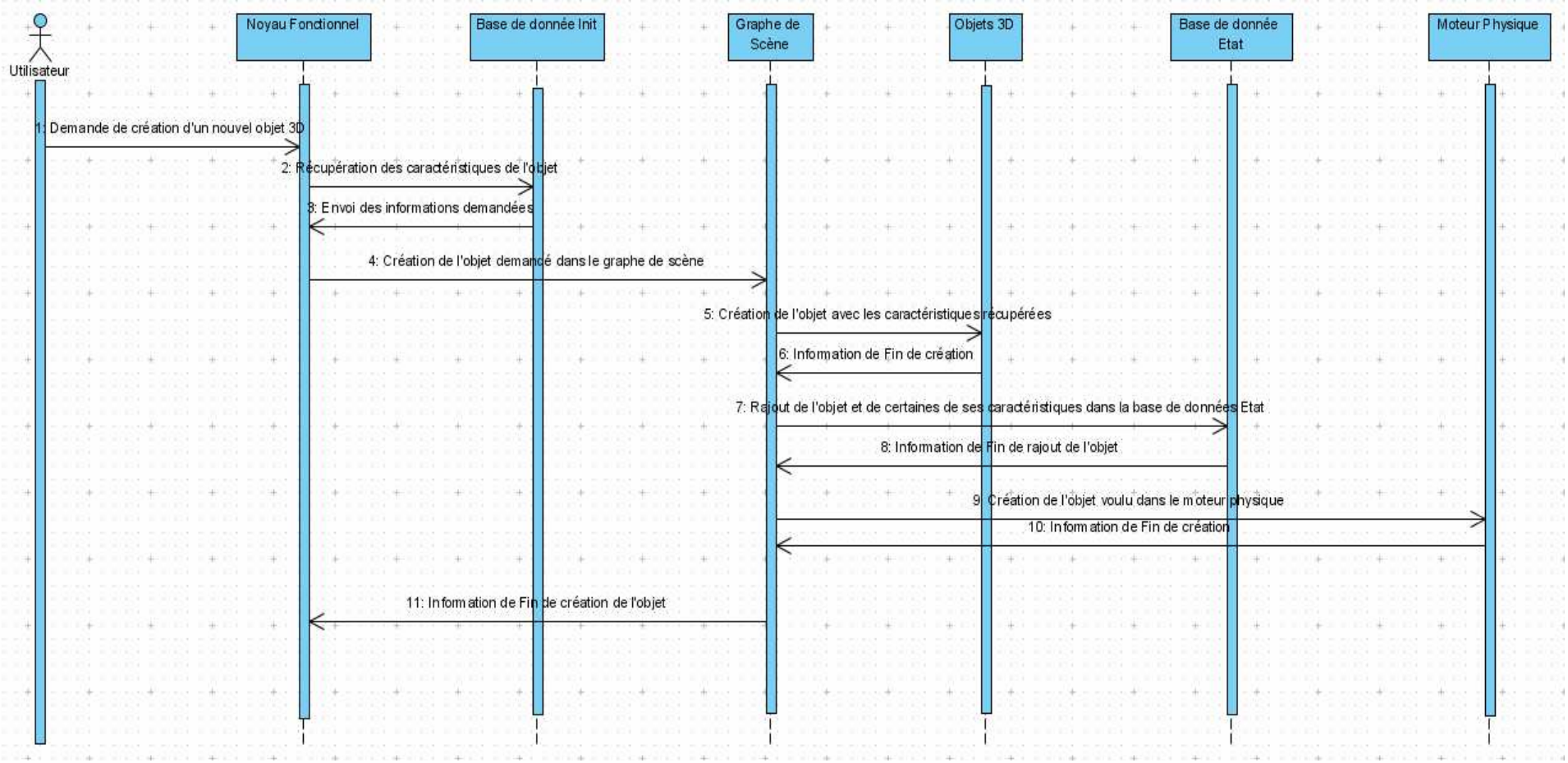


Diagramme de séquence 14 - Initialisation du module de visualisation

.

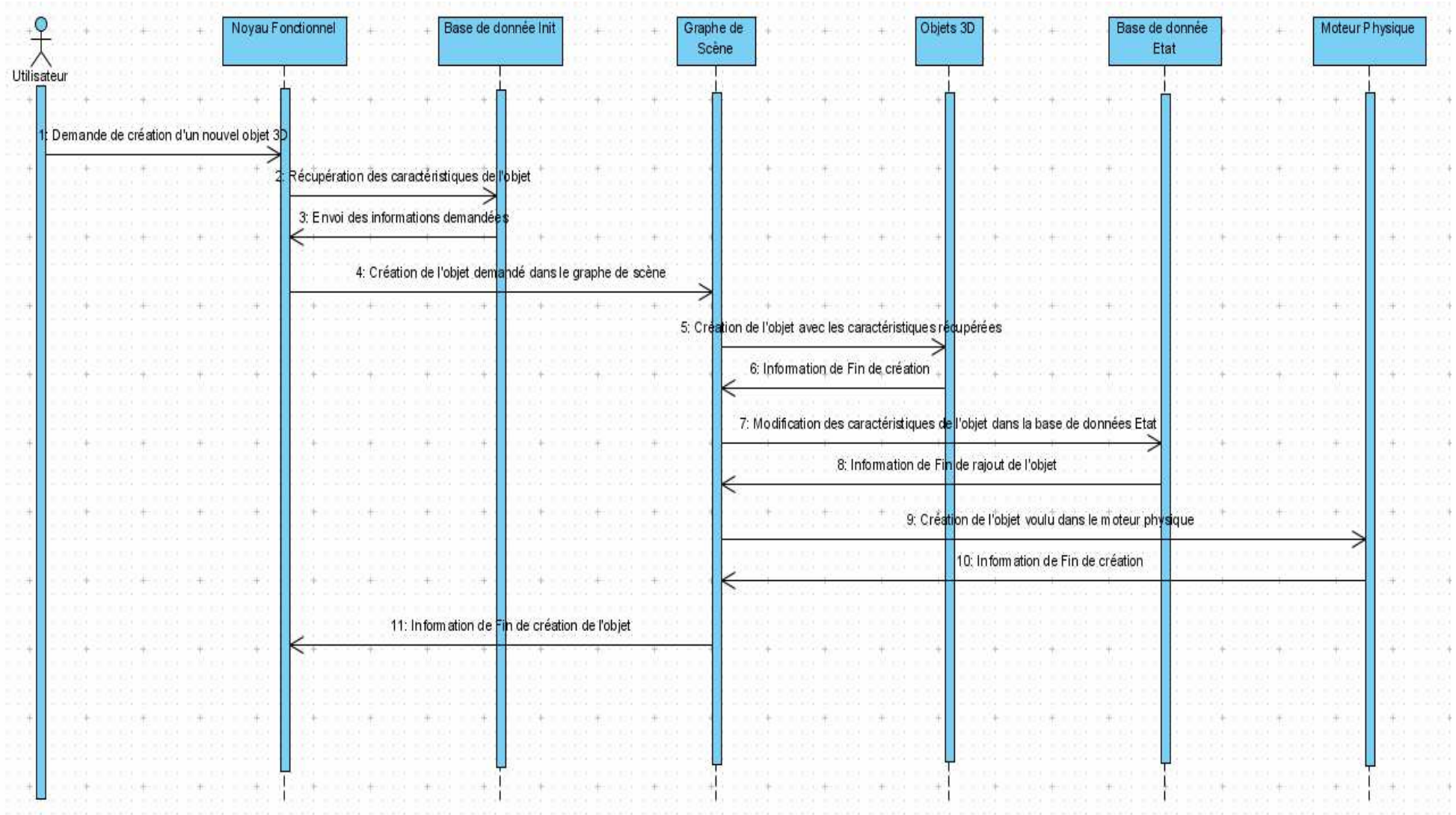


Diagramme de séquence 15 - création d'une nouvelle instance d'un objet existant

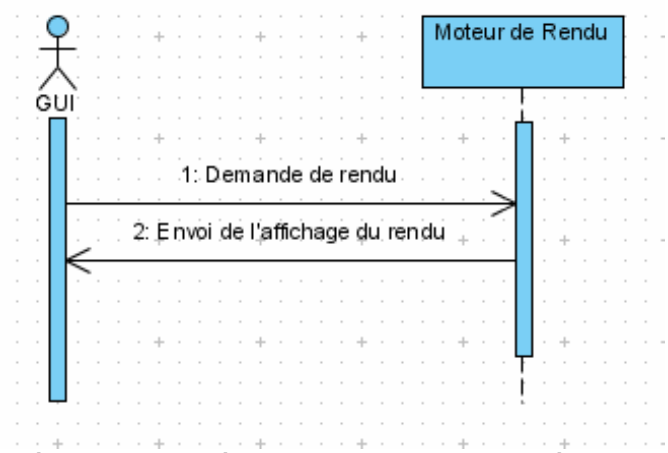


Diagramme de séquence 16 - Séquence d'envoi du rendu à la GUI

4.5. Performances et engagements

Les performances liées au rendu de la scène dépendent du type d'affichage utilisé :

→ En monovision, le taux de rafraîchissement doit respecter 25 Hz.

→ En stéréovision, le taux de rafraîchissement doit respecter 100 Hz. Ce mode d'affichage est optionnel.

Nous veillons, particulièrement, à ce que le monde 3D représenté dans notre module de visualisation soit en accord avec le monde 3D implanté sur le module du Bras Haptique. En conséquence, nous recevons fréquemment les données nécessaires provenant du module Haptique.

5. Bras Haptique

5.1. Présentation

L'application que l'on va mettre en œuvre sera dotée d'une composante haptique permettant d'intégrer un retour d'effort dans les futures applications construites. L'apport d'une composante haptique constitue un gain important en terme de réalisme dans une application de réalité virtuelle. En effet, la technologie du contact 3d favorise une interaction plus naturelle, efficace et intuitive en permettant à des utilisateurs d'avoir une interaction continue et bidirectionnelle avec leur travail. Le toucher est considéré comme étant le seul sens entièrement bidirectionnel, permettant à une personne d'envoyer et de recevoir de l'information en même temps.

Une fois connecté à un ordinateur et malgré les différentes applications réalisables, on retrouve toujours un même schéma de fonctionnement général d'un dispositif haptique :

- D'abord, le dispositif transmet à l'ordinateur (via une interface spécifique) l'ensemble des informations de position de l'effecteur, d'éventuelles actions (pression d'un bouton etc) et forces exercées par l'utilisateur.
- Ensuite, l'ordinateur calcule, en fonction de différents critères, la nouvelle position de l'effecteur et les éventuelles forces à exercer.
- Enfin, l'ordinateur transmet les forces à appliquer au dispositif qui sont donc, au final, ressenties par la main et le poignet de l'utilisateur.

C'est ce que l'on nomme la boucle haptique. C'est cette boucle qui devra, pour permettre le réalisme du retour d'effort, travailler à une fréquence d'environ 1KHz.

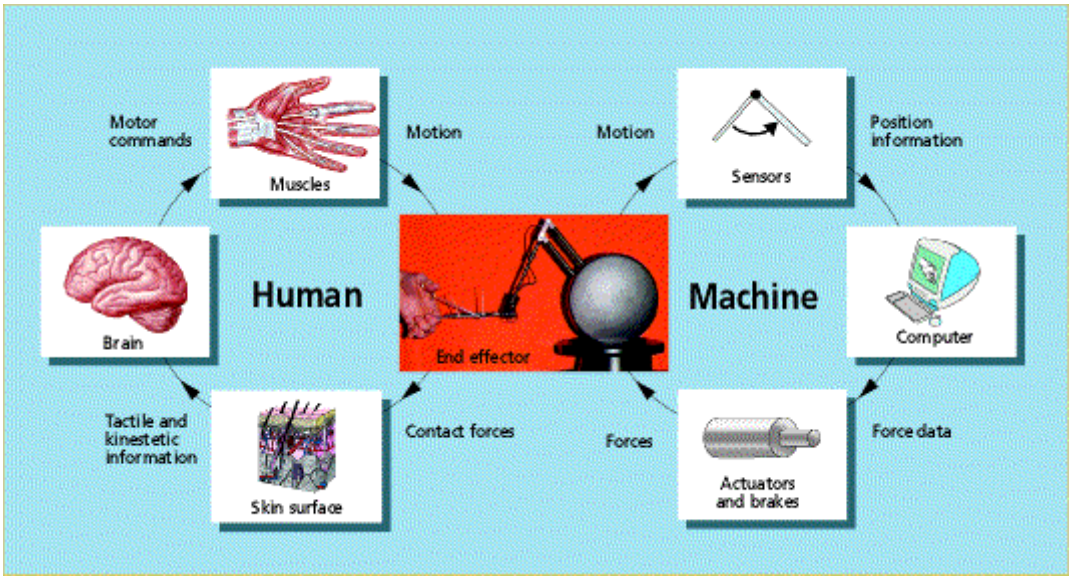


Figure 14 - Schéma général d'une application haptique

Nous allons définir la notion de point haptique : c'est le point qui représente le point de contact dans le monde virtuel (avatar) et qui est, dans le monde réel, le point par lequel on touche le monde virtuel (extrémité de l'effecteur).

5.2. Architecture/Interfaces

Dans le cadre de notre projet, et tout en respectant la philosophie décrite précédemment, la mise en œuvre du module haptique suivra le schéma d'architecture suivant :

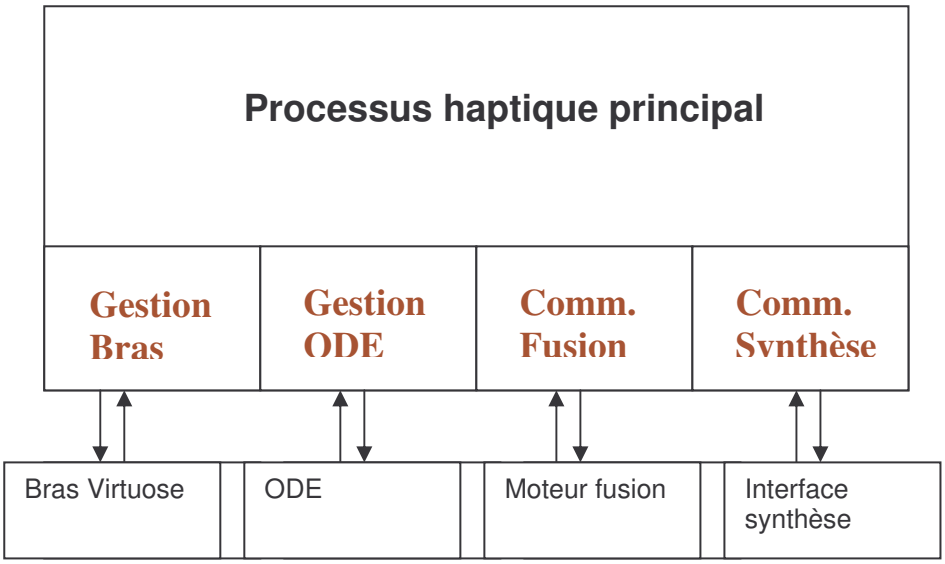


Figure 15 - Architecture

Depuis cette architecture, on distingue plusieurs processus fonctionnels :

- Tout d'abord, un processus principal, le **processus haptique principal**, contenant la machine à état du système et chargé d'ordonnancer et gérer les interactions entre les différents éléments.

- Un processus **gestion du bras**, s'appuyant sur la librairie spécifique du bras, et chargé de contrôler les interactions en entrée/sortie avec le dispositif haptique. Ces interactions tourneront principalement autour de la récupération des données fournies par le bras et la transmission des données de force à ce dernier. Ce processus tiendra lieu d'interface entre le dispositif matériel et le reste du module haptique via le processus principal.

- Un processus **gestion du moteur physique**, s'appuyant sur le moteur physique ODE, et chargé de gérer les interactions entre le moteur physique et le module haptique via le processus principal. Ces interactions tourneront principalement autour du calcul des interactions (force, vitesse, position, etc.) entre point haptique et objets de la scène virtuelle. Le résultat de ces calculs sera ensuite utilisé comme données retour sur le bras. Ce processus tiendra lieu d'interface entre le moteur physique ODE et le reste du module haptique via le processus principal.

- Un processus de **communication avec le reste du système MOVEIT** chargé de gérer les interactions entre le module haptique et le reste du système global. Ces interactions tourneront principalement autour de la transmission des données liées au fonctionnement du module haptique vers le module de fusion multimodale. Ainsi, les informations de mode de fonctionnement, d'événements

liés à la pression de boutons, les erreurs, etc. sont transmis vers le moteur de fusion. En retour, le moteur pourra transmettre vers le module haptique les informations de changement de mode à effectuer, les éventuelles requêtes du moteur pour obtenir des informations sur le fonctionnement du module etc. Ce processus servira d'interface externe avec le reste du système.

- Un processus de **synchronisation avec la synthèse**, sera chargé d'assurer la synchronisation entre le modèle physique du module haptique et le moteur physique du module synthèse. Ce processus sera chargé de recevoir les informations concernant la scène virtuelle en provenance du module synthèse comme la scène virtuelle d'initialisation. De plus, ce processus sera chargé de communiquer au module synthèse les informations concernant l'avatar (position, vitesse, orientation…) pour permettre la synchronisation entre les deux moteurs physique.

5.3. Communication

Afin de satisfaire les contraintes temps réel de notre module, nous devons garantir une bonne gestion des communications à l'intérieur de notre module et avec le reste du système. Pour cela, on privilégiera un type de communication synchrone entre les différents processus du module, cela afin de palier principalement aux problèmes de partage de données entre ceux ci. Ce type de communication sera également utilisé pour communiquer avec le module synthèse car dans un soucis de cohérence, on s'attachera à effectuer un traitement périodique de la tache de synchronisation tout comme un traitement « instantané » des données de scènes envoyé par le module synthèse comme la modification du point de vue, etc. Enfin, les traitements principaux des différents processus liés aux bras et aux calculs d'interactions seront traités de manière asynchrone. Le processus principal chargé de l'ordonnancement global sera lui de type synchrone afin d'assurer la bonne cohérence des taches effectués par l'intégralité du module haptique.

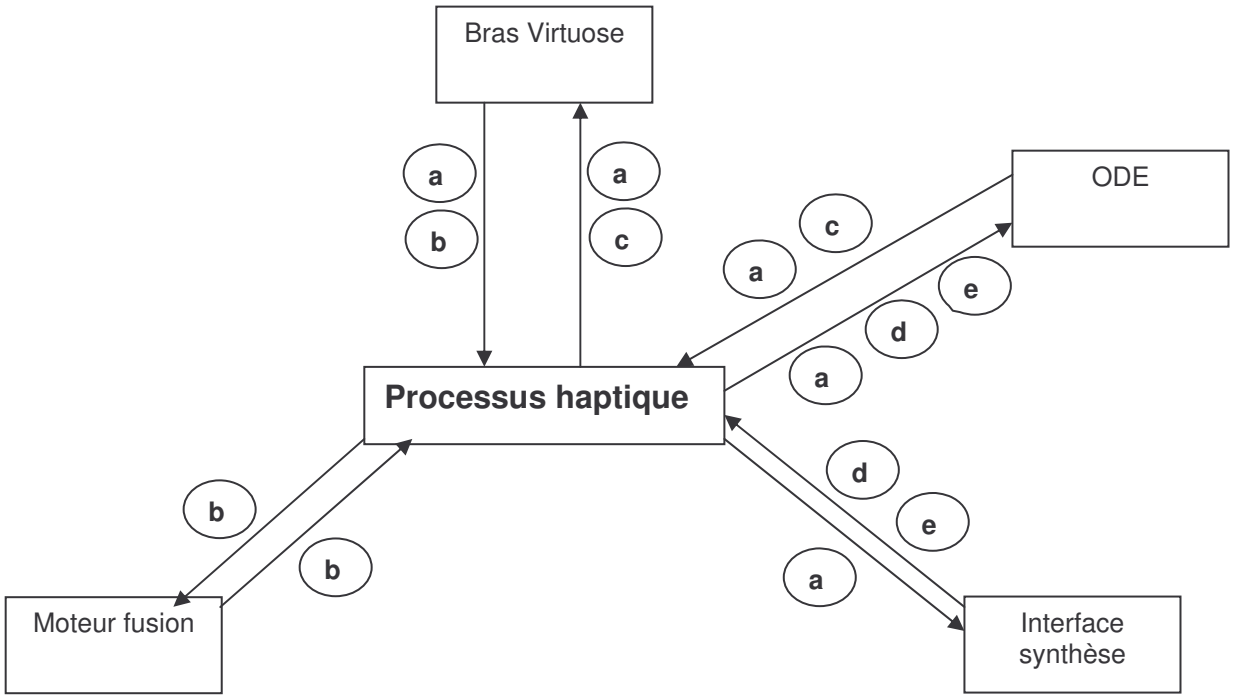


Figure 16 - Communication des interfaces externes

Structures de données principales :

- **a** : Structure de données représentant les caractéristiques de l'avatar dans l'environnement (coordonnées de l'avatar dans l'environnement)
- **b** : Structure de données représentant l'état du bras (mode souris, saisie/relâche)
- **c** : Structure de données représentant les efforts à transmettre à l'avatar et donc au bras haptique
- **d** : Structure de données décrivant le monde physique ODE (caractéristiques de l'environnement, ajout/suppression d'objets dans l'environnement)
- **e** : Structure de données représentant les caractéristiques de la caméra dans l'environnement (coordonnées de la caméra dans l'environnement)

5.4. Représentation de l'environnement

L'environnement sera physiquement modélisé avec le moteur physique ODE. Celui-ci ce chargera de gérer les interactions entre l'avatar, les objets de l'environnement et les limites de l'environnement.

ODE (Open Dynamic Engaine) :
La librairie ODE est constituée de deux parties distinctes : un moteur de détection de collision et un moteur de simulation dynamique de corps rigide.

Le moteur de collision possède les informations détaillées sur la forme de chaque corps. Cela lui permet, à chaque étape de la simulation, de trouver quel corps touche tel autre corps. Il fournit ensuite à l'utilisateur les points de contacts de ces collisions.
Le moteur de collision possède les informations sur la position, orientation, vitesse, centre de masse, distribution de masse de chaque corps. Cela lui permet, à chaque étape de la simulation, de faire évoluer la position et l'orientation de chaque corps selon la gravité et les forces de frictions.

6. Vocal

Ce chapitre a pour objectif de présenter des éléments de conception générale relatifs au module Vocal.

6.1. Présentation

Nous allons dans un premier temps rappeler les fonctionnalités propres à ce module. La modalité vocale en entrée pourra donc intervenir dans les situations suivantes :

- Gestion du monde (agir sur l'éclairage, ajouter des éléments de décor),
- Gestion des objets (créer et détruire des objets, se renseigner sur leurs caractéristiques et les modifier…),
- Manipulation (saisir et lâcher des objets…),
- Interaction Homme Machine (ouverture/fermeture de menus/fichiers, sauvegarde).

En entrée, la synthèse vocale interviendra pour donner un feedback à l'utilisateur sur ses actions. Le choix de la synthèse se justifie d'une part car c'est un feedback naturel lors d'une communication orale. D'autre part cela permet d'alléger les messages visuels afin que l'attention de l'utilisateur soit plus concentrée sur l'environnement virtuel.

6.2. Architecture

Ce schéma résume l'architecture de notre module. Pour le système de reconnaissance vocale ainsi que de synthèse vocale, nous avons choisi une solution logicielle existante. Ce choix sera justifié plus loin.
Les deux entités que nous allons développer sont le système de compréhension et le système de génération de texte. Ces deux systèmes sont génériques et font appel à des ressources linguistiques et bases de connaissances spécifiques à l'application (grammaire, lexique, base de connaissances et phrases préécrites). Au niveau de ces fichiers, nous faisons apparaître uniquement celles que nous aurons à élaborer. Les modèles acoustico-phonétiques par exemple sont intégrés au système de reconnaissance. Le Moteur de Fusion/Fission apparaît sur ce schéma car de nombreux échanges interviennent avec lui et c'est aussi lui qui gère les réactions du système à une commande vocale.

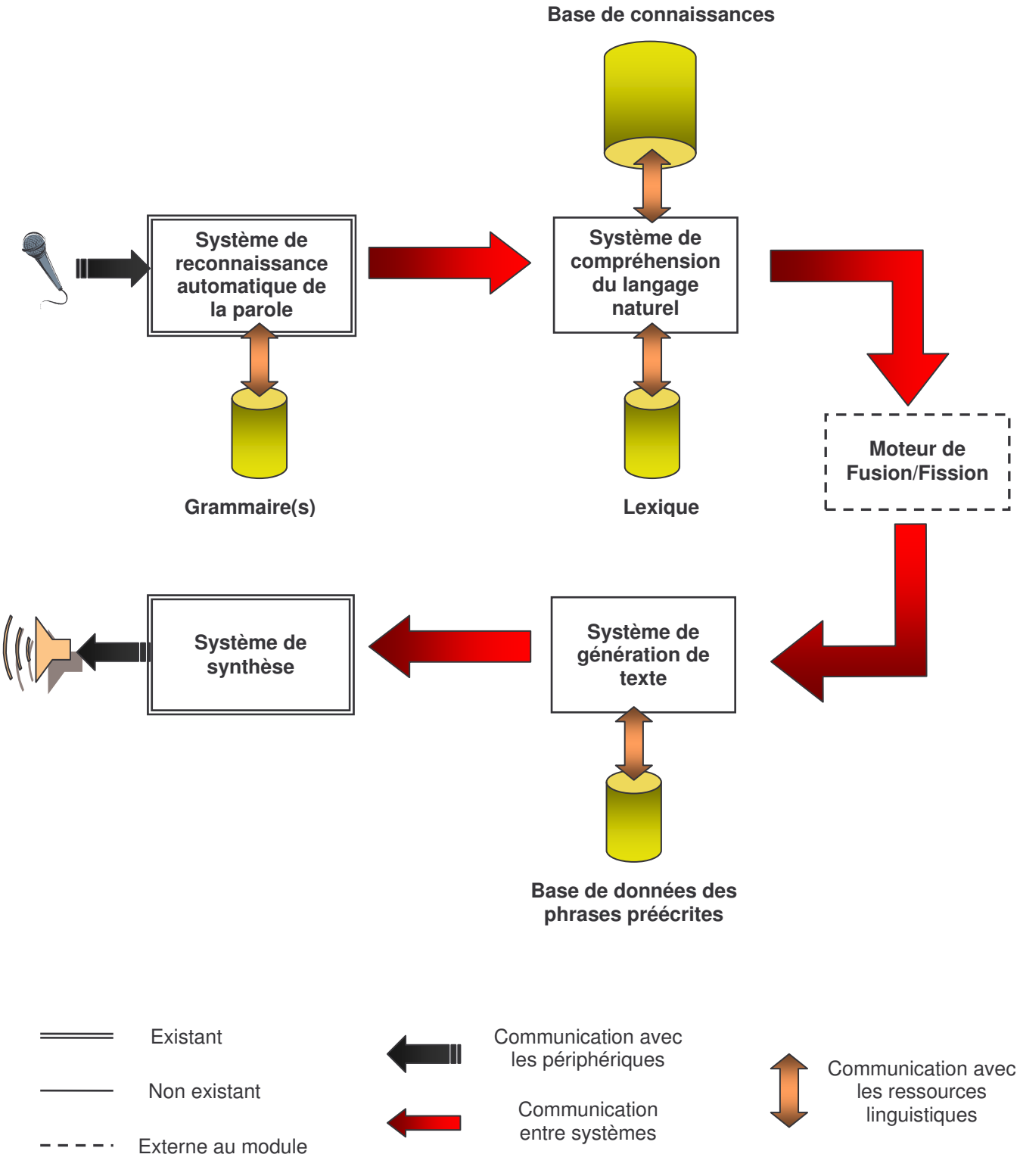


Figure 17 - Architecture du module Vocal

6.3. Choix de conception

6.3.1. Système de reconnaissance automatique de la parole

Nous avons choisi d'utiliser une plateforme de reconnaissance existante développée par ***Nuance*** (http://www.nuance.com). Celle-ci nous a paru la plus adaptée à l'application à développer pour les raisons suivantes :

- Moyennant l'achat d'une licence adéquate, elle peut être multi OS.
- Etant destinée à des applications téléphoniques, elle s'avère extrêmement robuste pour une utilisation dans un environnement bruité.
- L'utilisation des grammaires dynamiques est possible.
- Elle est compatible avec ***VoiceXML***, laissant possible de nombreux ajouts.

Bien que ***Nuance*** possède une bonne détection de début et de fin d'élocution, nous n'excluons pas l'utilisation d'un déclencheur externe de reconnaissance (bouton poussoir, pédale, mot clé).

Ce que nous (et ***Nuance***) appelons « grammaires dynamiques », sont en réalité des grammaires statiques (car intégralement écrites au préalable) mais qui sont chargées dynamiquement. On peut ainsi basculer d'une grammaire à l'autre en fonction des tâches en cours. Par exemple si l'utilisateur a besoin de mots absents de la grammaire « de base » on peut basculer sur une autre plus complète, ou plus spécifique. De même, si des mots sont inadaptés à la tâche on peut filtrer une partie du vocabulaire, éviter des erreurs de reconnaissance et gagner en temps de calcul.

6.3.2. Système de synthèse

Nous avons choisi d'employer un agent de synthèse à partir de texte (Text To Speech) plutôt que d'enregistrer des fragments de parole.
Notre choix à ce sujet n'est pas définitivement arrêté. Pour l'instant, nous penchons plus pour l'emploi de ***ppilot*** (http://www.irit.fr/recherches/MODEL/DIAM/ztp) en raison de sa simplicité d'utilisation et d'une qualité de synthèse relativement correcte.
Toutefois ce choix n'est pas déterminant pour la conception de notre module. Comme nous avons choisi une approche par agents répartis (un agent de reconnaissance, un agent de synthèse vocale), les protocoles de communication restent simples à adapter.

6.3.3. Langages employés

Pour assurer la portabilité des librairies, nous utiliserons le langage ***Java***. De plus, le seul autre module avec lequel nous sommes en relation est le Moteur de Fusion/Fission et il emploie également ce langage.
XML sera utilisé pour la réalisation de certaines ressources linguistiques.

6.4. Données manipulées

6.4.1. Données de communication internes et externes

Lorsque l'utilisateur prononce une phrase, un signal sonore entre dans le système de Reconnaissance Automatique de la Parole. Celui-ci en extrait une ou plusieurs phrases reconnues associées à un score de vraisemblance. On en choisit les n meilleures que l'on transmet au système de compréhension. Ce système a pour but de construire une structure. Ces structures sont de plusieurs types, directement associés aux fonctions à exécuter.

Exemple indicatif et non définitif de structure associée à la fonction « changer la couleur de l'éclairage » :

```
CHANGER_COULEUR : [ LUMIERE : [id_lumière, nom_lumière, couleur_actuelle],
                  COULEUR : [nouvelle_couleur, intensité_lumière]
                  ]
```

Nous avons choisi ce format de données car il est aisément manipulable, pour rajouter des paramètres, fusionner des structures…

Cependant, afin d'adopter un langage commun avec les autres modules (puisque la fonction pourra être sollicitée par une autre modalité) et parce que ces derniers n'ont pas forcément l'utilité des structures, nous la simplifierons avant de la transmettre au moteur de fusion. Cette structure sera donc convertie en une commande dont le format a été définit précédemment par le Moteur de Fusion/Fission.

Le moteur de fusion se charge de traiter ce message et la fission envoie un retour aux modalités de sortie. Ce message est du même format que le précédent, mais étant donné qu'il est destiné à générer un texte, il doit contenir un identifiant associé aux phrases préécrites de la base de données. Des paramètres peuvent venir compléter ce message pour particulariser le texte synthétisé en fonction de ce qu'a dit l'utilisateur.

C'est donc une phrase qui est finalement transmise au système de synthèse vocale.

6.4.2. Ressources linguistiques

- **Grammaire :** elle contient tout ce que le système est capable de reconnaître. Afin de la rendre la plus naturelle possible, nous avons pensé à un programme d'étude auprès d'utilisateurs extérieurs au projet pour son élaboration. Après une mise en situation de ces utilisateurs, nous recueillerons le vocabulaire et la structure des phrases employées (Magicien d'Oz, voir protocole ci-dessous).

- **Lexique :** il contient les mots de la grammaire ainsi que les concepts associés. Par exemple au mot « brique », on associe le concept « objet ».

- **Base de connaissances :** elle contient les applications des concepts précédents dans le domaine de la tâche. Par exemple au concept « objet », on définit ses caractéristiques (taille, couleur…).

- **Base de données des phrases préécrites :**
  Elle contient les phrases à générer par la synthèse, associées à un identifiant commun aux autres modules. Ces phrases peuvent éventuellement avoir des champs manquants, qui seront renseignés par les paramètres transmis par le moteur de fusion.

6.4.3. Protocole du magicien d'Oz relatif à l'écriture de la grammaire :

Nous avons décidé d'utiliser la technique du magicien d'Oz qui permet de simuler les fonctionnalités de notre module afin de d'établir une liste la plus complète possible du vocabulaire employé et des structures de phrases.
Pour cela, nous avons à disposition les pièces Lego qui seront en place dans l'application (les couleurs, les formes) ainsi qu'un dictaphone.
Nous devrons donc aller interviewer un certain nombre de personnes (un panel assez large : âge, profession…) afin de les mettre en scène : ils devront agir comme s'ils utilisaient l'application. L'un des membres de l'équipe jouera le rôle du bras haptique et un autre se tiendra près du sujet afin d'enregistrer tous ses comportements vocaux. Nous lui demanderons ensuite d'utiliser chacune des fonctionnalités gérées par le module vocal, ceci dans différentes situations afin d'établir une liste de mots et de structures de phrases.

Nous le mettrons tout d'abord en scène, c'est-à-dire que nous lui détaillerons l'environnement dans lequel il se trouve : une chambre d'enfant éclairée par différentes sources de lumière. Le stock (représenté par des tiroirs, des coffres à jouets, …) est initialement vide, il devra donc commencer par le remplir. Nous lui expliquerons aussi les différents modes (déplacement, souris, saisie).

Afin d'étudier le comportement devant chacune des fonctionnalités que nous avons définies, nous les lui listerons :

Le sujet agira à sa guise en formulant ses requêtes. Il bénéficiera d'un retour vocal et pourra voir se réaliser, par l'intermédiaire de la personne assurant le rôle du bras haptique, les actions qu'il demande.

7. Réseau

7.1. Présentation

Le projet MOVEIT est découpé en différents modules. La communication inter modules se fait via le réseau. Certains de ces modules sont indispensables et d'autres sont optionnels. La bibliothèque réseau met à disposition de chaque autre module des services leur permettant de s'inter connecter et de s'échanger des informations. Elle est donc indispensable au bon fonctionnement de MOVEIT. Sur chaque ordinateur hébergeant un ou plusieurs modules, la bibliothèque réseau doit être disponible.

7.2. Contraintes

Certains modules ayant des besoins de communication importants, l'implémentation du réseau doit garantir un temps de réponse optimal.

Comme exprimé dans le cahier des charges, l'emplacement des modules sur le réseau n'est pas fixé. Les modules peuvent être répartis sur différentes machines. Il est également possible d'avoir plusieurs modules sur une même machine. Il est donc indispensable que la gestion du réseau soit dynamique.

Le projet MOVEIT étant multi plateforme, la bibliothèque réseau doit supporter les systèmes d'exploitation les plus répandus (Windows 9x/NT, Linux, MacOS).

Certains modules seront implémentés en Java et d'autres en C++. La bibliothèque réseau devra donc être disponible dans ces deux langages.

Les modules peuvent avoir à transporter tout type de message, la bibliothèque réseau ne doit pas faire de suppositions sur ce qui est transporté ni imposer de contraintes sur ces messages.

7.3. Fonctionnalités

Le réseau permet l'échange de messages quelconques. Les modules peuvent et doivent communiquer selon un format qui leur est propre. Le réseau n'impose aucune contrainte sur le contenu des messages (les différents modules s'accordant ainsi entre eux sur une norme de communication).

A tout moment, un module peut s'abonner ou se désabonner du réseau. Dès son arrivée sur le réseau, il est en mesure de se faire connaître des autres modules et de connaître la liste des autres modules déjà présents.

Un module ayant déterminé les modules présents est à même d'établir une connexion avec le ou les modules qui l'intéressent permettant ainsi une communication bidirectionnelle. Un module peut également prendre l'initiative de se connecter à un autre une fois que ce dernier s'est fait connaître sur le réseau.

L'envoi de message est « dirigé ». Un module n'émet pas de message au hasard, il émet vers un destinataire, qu'il soit un module seul ou un groupe de module.

Il est possible d'adresser un ou plusieurs modules de manière transparente. Un module peut émettre un message vers un groupe de modules comme il émettrait vers un unique module.

Lors de la réception d'un message, le destinataire connaît l'identité de l'expéditeur ainsi que le groupe de celui-ci, s'il fait partie d'un groupe.

7.4. Solutions

Pour offrir les fonctionnalités décrites précédemment, nous avons pensé à une solution mélangeant l'idée d'un bus réseau et l'utilisation de liaisons directes pour répondre aux différentes contraintes.
Le bus est utilisé comme « point de rendez-vous » pour les modules. C'est ce que nous appelons le « réseau d'attache ». Les modules s'y connectent lorsqu'ils souhaitent se faire connaître et découvrir les autres modules disponibles.
Lorsqu'ensuite ils souhaiteront communiquer, ils utiliseront les informations obtenues sur le réseau d'attache pour se connecter directement aux modules. Tout ceci étant bien entendu transparent pour l'utilisateur (voir les fonctionnalités du point de vue du programmeur ci-dessous).

D'autre part, la nature des messages étant à définir par les modules concernés, certains peuvent avoir à transporter du texte, certains des données brutes. Nous avons choisi de transporter des données binaires sans faire de suppositions sur leur type.

7.4.1. Solutions existantes

Nous avons effectué des recherches sur les moyens existants permettant de faire ce que nous souhaitions mettre en place. Nous présentons ici un rapide tour des solutions que nous avons rencontrées, ainsi que la justification des choix que nous avons fait.

- Le bus logiciel Ivy (http://www.tls.cena.fr/products/ivy/)

Le bus logiciel Ivy est ce qui a été utilisé l'an dernier pour le PGE. Il s'agit d'un bus orienté messages. Il connecte des agents qui peuvent ensuite émettre et recevoir des messages sous forme de texte.

La principale raison qui fait que nous n'utiliserons pas Ivy est le fait qu'on ne peut transporter des messages que sous forme de texte.

Dans le cadre de notre PGE, il va falloir connecter des entités qui communiquent des données textuelles, mais aussi des entités qui communiquent des positions, des sommets, des résultats de calculs physiques etc.
Il est impensable, pour ces entités qui vont avoir des besoins de réponse important, de transformer toutes ces valeurs binaires en texte (ou sous n'importe quelle autre forme), les faire transiter sur le bus et de l'autre côté refaire l'opération inverse.

- Le bus D-BUS (http://dbus.freedesktop.org/doc/dbus-tutorial.html)

D-BUS est un système spécialisé la communication inter processus (IPC). Il se compose généralement de clients et d'un démon chargé d'assurer le routage des messages émis. Il est utilisé par exemple au sein du projet GNOME pour transmettre à l'utilisateur ou aux applications toute sorte de messages (une nouvelle clé usb est disponible, un contact vient de se connecter, la batterie du portable est faible ...).

Nous n'avons pas utilisé D-BUS car il est surtout destiné à une utilisation mono-machine, lors d'une session utilisateur sur cette machine par exemple. L'autre ennui est qu'il dépend d'un démon, et nous souhaitions mettre en place une architecture décentralisée.

- L'architecture CORBA (http://www.omg.org/gettingstarted/corbafaq.htm)

CORBA est l'acronyme de **C**ommon **O**bject **R**equest **B**roker **A**rchitecture. Il s'agit d'un ensemble de spécifications dont il existe quantité d'implémentations. CORBA est orienté appel de procédures distantes sur des objets.

Nous avons écarté CORBA pour deux raisons essentielles. La première c'est que nous avons à mettre en place un moyen de communication orienté messages (même s'il est possible au travers de ce moyen de communication de faire de l'appel de procédure à distance). La deuxième c'est que CORBA est quelque chose de très lourd à utiliser et à mettre en place. Nous avons jugé qu'étant donnés les délais impartis, il n'était pas judicieux de se lancer là dedans.

- Java Remote Method Invocation (http://java.sun.com/products/jdk/rmi/index.jsp)

Les RMI de Java sont une extension du langage Java pour supporter nativement l'appel de procédures distantes sur des objets partagés.

Le problème est que c'est disponible nativement uniquement avec Java, or nous allons devoir fournir une interface réseau à des modules codés en Java et C++.

- Le bus Mbus (http://www.mbus.org/index.html)

Le bus Mbus est une implémentation d'une RFC (RFC 3259). Il s'agit à la base d'un « bus orienté messages pour la coordination locale » et à été élaboré comme un moyen de coordination pour la communication entre différents composants applicatifs. Dans le principe cela ressemble beaucoup au bus Ivy.

La limitation de ce bus est la même que celle d'Ivy, à savoir qu'il ne transporte que des messages sous forme textuelles.

7.4.2. Choix de conception

A la lumière de toutes ces solutions nous avons donc choisi d'implémenter notre bibliothèque réseau afin d'allier nos fonctionnalités à des performances optimales.

Pour des raisons de clarté, nous allons ici expliciter certains termes que nous employons.

On distingue un certain nombre de concepts :

- **Les messages :**
  Ce sont les données que vont s'échanger les modules entre eux. Ils peuvent être de n'importe quelle sorte, cela n'influera en rien le comportement de la bibliothèque.

- **Les modules :**
  Ils vont utiliser cette bibliothèque pour communiquer entre eux. Ils ont des messages à se transmettre, quelle que soit leur nature. Ils se mettent à disposition des autres par l'intermédiaire du réseau d'attache.

- **Le réseau d'attache :**
  C'est un espace où les modules se connectent et se font connaître. Ils peuvent également y découvrir quels sont les autres modules et groupes connectés. C'est à partir de là qu'un module décide d'établir une liaison avec un autre module ou avec un groupe.

- **Les liaisons :**
  C'est au travers de ces liaisons que les modules vont échanger leurs messages avec d'autres modules ou avec différents groupes. Ces liaisons sont indépendantes du réseau d'attache. Elles sont établies par les modules vers d'autres modules ou d'autres groupes.

- **Les groupes :**
  Ce sont des ensembles de modules auxquels un module peut se connecter. Par exemple un groupe « Input » qui contiendrait plusieurs modules (« clavier » et « bras »). Ce concept permet à un module d'envoyer des messages ou d'en recevoir à partir d'un ensemble de modules sans connaître la nature des modules du groupe. C'est à chaque module de spécifier s'il appartient à un groupe.

- **Le réseau :**
  C'est l'ensemble des concepts précédents. C'est la vue qu'auront les autres modules du module réseau, à savoir une boîte offrant les services définis.

D'un point de vue programmeur, voilà ce qu'il sera possible de faire avec notre bibliothèque :

→ Joindre et quitter un réseau d'attache.
→ Connaître à tout moment la liste des autres modules présents sur le réseau d'attache que l'on a rejoint.
→ Ouvrir des liaisons avec un ou plusieurs modules (on parle alors de groupe) présents sur le réseau d'attache sur lequel on se trouve.

→ Envoyer et recevoir des messages par le biais de ces liaisons de manière transparente, que l'on s'adresse à un module seul ou à un groupe.
→ Spécifier des actions à effectuer à l'occurrence de certains évènements, ces évènements étant pour le moment au nombre de trois :
- ✓ arrivée / départ d'un module du réseau d'attache
- ✓ connexion entrante d'un module
- ✓ réception d'un message sur une liaison

Concernant la forme des messages envoyés, nous avons dit qu'il était du devoir des modules souhaitant communiquer entre eux de définir une manière de se comprendre. Nous avons pensé à proposer un mécanisme basique de typage de message. Un message serait alors constitué de trois parties :

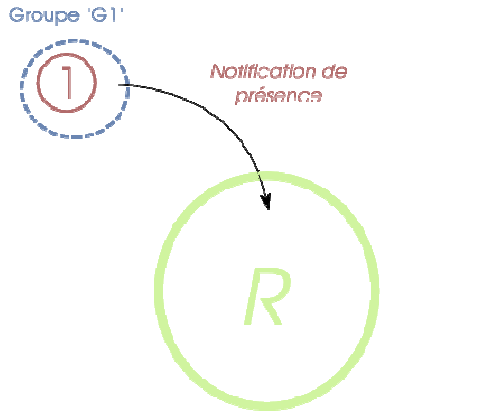
- un domaine. Ce domaine représente l'identifiant d'un groupe de commandes ; par exemple, le domaine « Clavier.Touche » indique que le message s'applique à une touche clavier.
- un code. Ce code représente la commande envoyée; par exemple, le code « Appui » indique que dans le domaine « Clavier.Touche », on notifie un appui.
- les données qui y correspondent. Pour coller avec l'exemple, les données ici consisteraient seulement en un caractère.

De la même façon que nous n'imposons pas de contraintes sur le format de données à transmettre, les domaines et les codes peuvent être représentés par des entiers, des chaînes de caractères, des tableaux de flottants ...

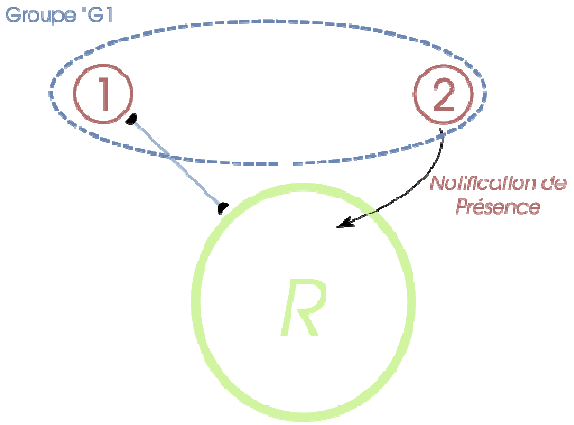
De cette façon il sera plus aisé pour les modules de définir des domaines et des codes pour chacun des messages types qu'ils peuvent émettre et recevoir, plutôt que de redéfinir un moyen d'identifier les messages qu'ils s'envoient.

Les modules du projet MOVEIT peuvent être répartis sur plusieurs ordinateurs. On peut également en rassembler certains sur une même machine. L'organisation retenue est déterminée en fonction d'un certain nombre de contraintes (quantité d'informations communiquées entre deux modules, besoin de rapidité entre les modules, puissance des ordinateurs à disposition, qualité du réseau disponible, encombrement ...).

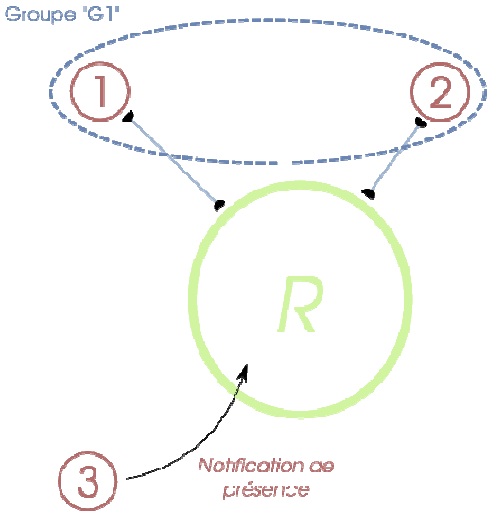
La figure ci dessous est un exemple d'interaction possible des modules sur le réseau.



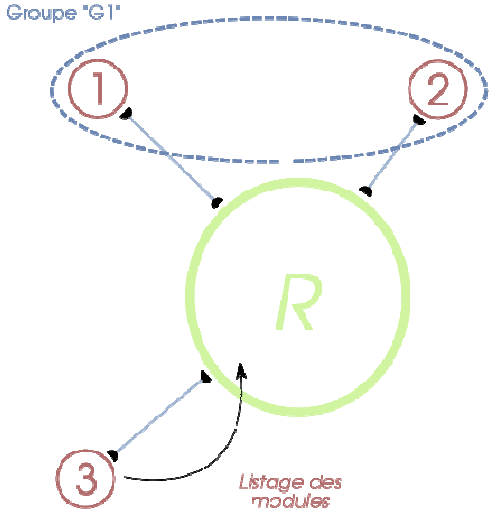
1 I Arrivée d'un module dans le réseau d'attache.
Le module 1 fait partie du groupe "G1"



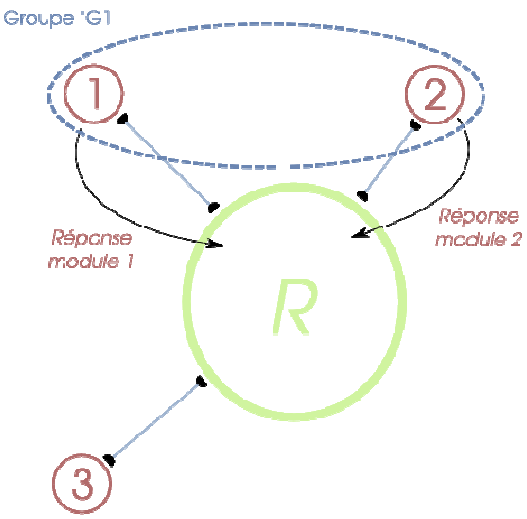
2 I Arrivée d'un module dans le réseau d'attache.
Le module 2 fait aussi partie du groupe "G1"



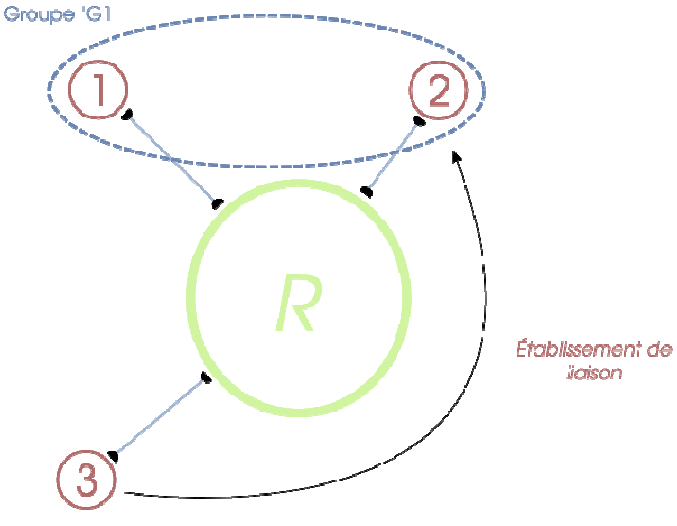
3 I Arrivée d'un module dans le réseau d'attache.



4 Requête du module 3 sur le réseau d'attache.



5 I Réponse des modules disponibles.



6 I Demande de liaison avec le groupe 'G1"
à l'initiative du module 3.

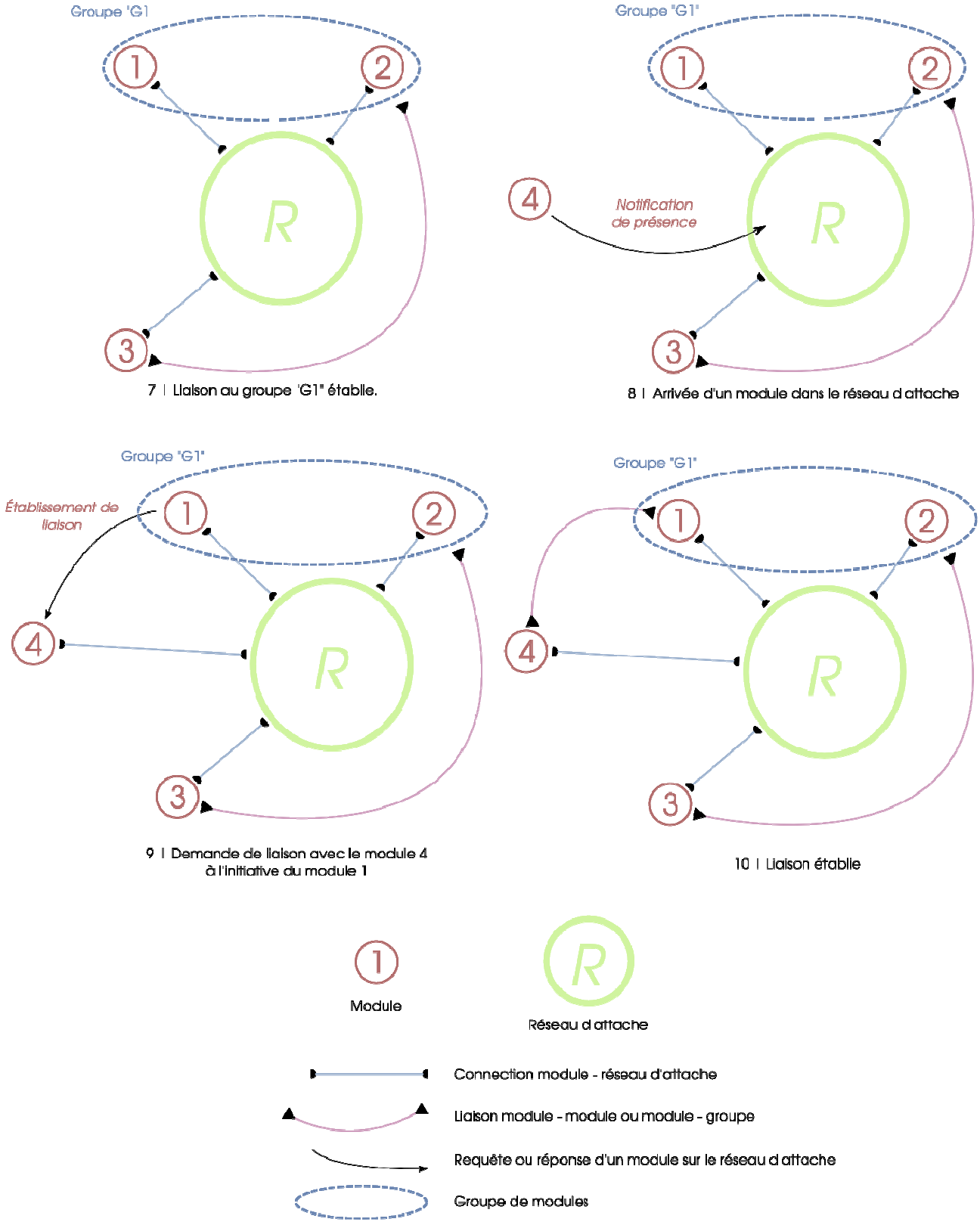


Figure 18 – Exemple d'utilisation de la bibliothèque réseau

8. La librairie MoveIT

La librairie MoveIT que nous développons et servant de base à l'application MoveIT se présente sous la forme d'une liste de fonctions permettant de réaliser l'ensemble des fonctionnalités spécifiées dans le document d'analyse fonctionnelle.

Ces Librairies sont découpées selon les mêmes modules que ceux de l'application. On développe ainsi une librairie par module. Les interfaces de ces librairies permettent d'envoyer et de recevoir des données. De même, ces librairies sont prévues pour qu'un appel distant à une de leurs fonctions soit possible.

9. Bilan organisationnel

9.1. Moyens

9.1.1. Moyens humains

Suite à la réunion d'initiation du projet, personne n'était vraiment pressenti pour s'occuper de la gestion de projet. C'est après la première réunion du 6/10 avec notre conseiller permanent (M. Grandjean) que 3 personnes se sont proposées pour s'occuper de la gestion de projet. C'est à l'approbation générale de la promotion que ces personnes ont pris les responsabilités de la gestion de projet (Enguerran Boissier, Caroline Brett et Maxime Fudym).

Autour du 15 octobre, l'équipe de gestion de projet a demandé 2 volontaires : un pour la qualité, l'autre pour la gestion du site. Pascal Barreau s'est proposé pour le premier point et Julien Cabieces pour le second.

En ce qui concerne la répartition du travail, chacun d'entre nous a fait ses propres recherches pour commencer. Tout le monde a ainsi pu trouver avec quel domaine du projet il avait le plus d'affinité.
Ensuite l'équipe de gestion de projet a demandé à ce que les recherches soient plus ciblées et efficaces. C'est alors que nous nous sommes repartis en plusieurs groupes de recherche, chacun étant chargé d'établir l'état de l'art afin de mieux apprécier les difficultés du projet.
Initialement prévues pour les recherches, ces équipes ont été réaménagées en fonction des besoins de chaque module. Par la suite, une personne de chaque groupe s'est naturellement proposée pour être chef d'équipe. Nous sommes cependant conscients que les responsabilités des chefs d'équipe risquent de changer en cours de projet du fait des souhaits de chacun. De plus, il est envisageable que ces équipes soient remaniées afin de satisfaire aux besoins les plus urgents.

Voici un tableau récapitulatif de la répartition des personnes composant la promotion dans les équipes (les chefs d'équipe sont en gras) :

Multimodalité	Réseau	Bras Haptique	Visualisation 3D/Rendu	Vocal
Alexandre Collier Isabelle Fonquernie Fabrice Lortet Mehdi Rabah Francesco Menini	**Guillaume Laurens** Aurélien Kamel Lionel Anton Robin Bourianes	**Guillaume Cabrillat** Adrien Guiomar Enguerrran Boissier Maxime Fudym Vincent Peyruqueou	**Eric Amiel** Guillaume Dangoumau Emmanuel Orvain Elodie Enjalbert Sébastien Arque Julien Cabieces	**Clément Picou** David Gallard Pascal Barreau Caroline Brett

9.1.2. Ressources

En ce qui concerne les ressources, nous avons utilisé aussi bien des ressources personnelles que des ressources liées à l'université.

Pour les locaux, nous avons utilisé les salles du bâtiment U3 pour réaliser les réunions concernant l'ensemble de la promotion. Ensuite, chaque équipe s'est réunie au domicile d'un de ses membres chaque fois que nécessaire pour l'avancement des travaux du module.

Pour les ressources informatiques, nous avons principalement utilisé des ordinateurs personnels. Cependant, nous avons ponctuellement profité des salles informatiques de l'U3 ainsi que de l'AIP.
Nous avons d'ailleurs pris contact avec les responsables de l'AIP pour savoir comment mettre en place les ressources liées au bras haptique et au développement. Dans l'idéal, nous souhaiterions disposer du bras haptique autant que possible et nous voudrions disposer de 5 ordinateurs mis à notre disposition pour avancer le projet (un par groupe de travail).

9.2. Réunions

Nous avons mis en place plusieurs réunions. Voici les dates de ces réunions ainsi que le propos de ces rendez-vous.

1 - Les réunions avec les experts/clients :
- →28/09 : présentation du projet
- →11/10 : initialisation du projet

2 - Les réunions avec le conseiller permanent :
- →6/10 : toute la promotion était présente. Nous avons commencé à réfléchir au moyen de réaliser la gestion de projet. M. Grandjean nous a expliqué quel serait son rôle lors du déroulement du projet.

→19/10 : réunion des chefs de projet avec le conseiller : aide à la gestion de projet, conseils sur le planning et les spécifications fonctionnelles.

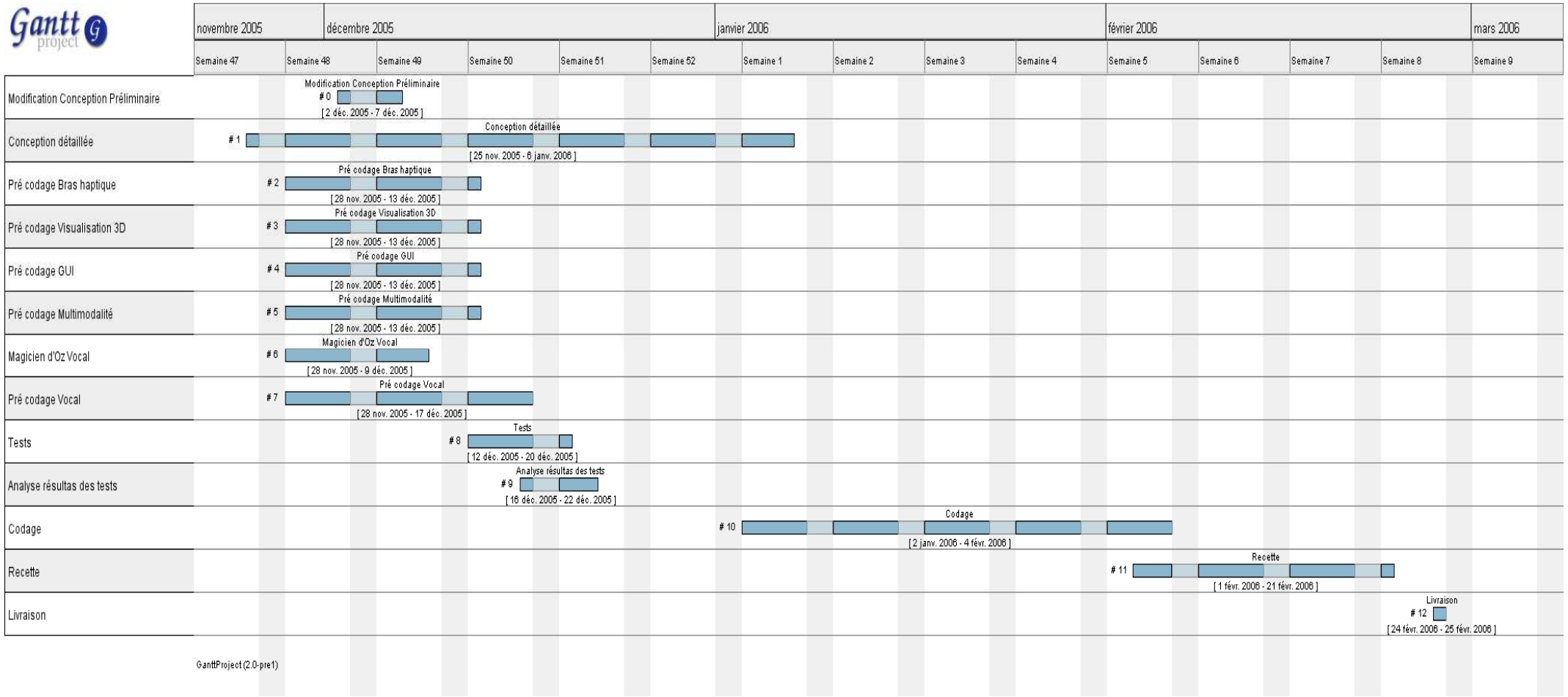
3 - Les réunions avec toute la promotion :

→ 4/10 : état de l'art
→ 8/10 : préparation à la réunion d'initialisation.
→13/10 : réunion pour se répartir la rédaction des documents (le cahier des charges, le plan de développement, le planning prévisionnel)
→20/10 : réunion de mise au point : voir où en sont les recherches, recadrer ces dernières en formant des groupes.
→ 27/10 : réunion de mise au point

4 - Les réunions en groupes restreints

→16/10 : réunion pour la rédaction du plan de développement et le planning
→Entre le 20/10 et le 01/11 : chaque équipe se regroupe afin de réaliser une première ébauche de dossier de spécification
→1/11 : réunion de l'équipe de gestion de projet avec les chefs d'équipe afin de mettre en commun chacune des spécifications : voir si toutes les parties étaient cohérentes.
→2/11 : réunion des chefs de projet : rédaction du dossier de spécification
→3/11 : réunion des chefs de projet : suite et fin de la rédaction du dossier de spécification
→5/11 : réunion des chefs de projet, rédaction du bilan organisationnel
→5/11 : réunion des chefs d'équipe : rédaction de la présentation en vue de la réunion du 7/11.

10. Planning prévisionnel



→ La tâche de « Modification Conception Préliminaire » correspond aux éventuelles modifications apportées à la conception lors de la revue du 02/12/2006.

→ La tâche de Conception Détaillée permet d'approfondir la conception et de s'approcher au plus de la réalisation du projet.

→ Les phases de pré codage correspondent à des tâches de prise en main des différents outils existant que nous réutilisons pour le projet. De plus c'est l'occasion de commencer à mettre à l'épreuve la conception réalisée jusqu'à maintenant et d'en changer au plus vite les points importants si cela s'avère utile au vue du pré codage.

→ La tâche de Magicien d'Oz permet au module vocal de créer un corpus de dialogue leur servant de base pour l'application. C'est également l'occasion de vérifier qu'il n'y a pas de lacunes importantes par rapport à l'utilisation décrite lors de la conception préliminaire.

→ La phase de test permet de vérifier le bon fonctionnement des composants réalisés lors de la phase de pré codage pour une éventuelle réutilisation lors de la phase de codage. Les résultats des tests seront donc analysés en conséquence