Le modèle de l'interface dans les environnements virtuels de formation

Grégory Faudet

22 *juin* 2004

Rapport de stage de DEA

Sous la responsabilité de Ronan Querrec

Remerciements

Je souhaite tout d'abord remercier toutes les personnes qui ont permis aux étudiants de l'ENIB de suivre la formation du DEA informatique de l'IFSIC. En particulier M. François Bodin, responsable du DEA, M. Antoine Beugnard, responsable de la filière brestoise du DEA et enfin M. Jacques Tisseau, directeur du Laboratoire d'Informatique Industrielle de l'ENIB (LI2) et du Centre Européen de Réalité Virtuelle (CERV).

Je tiens également à remercier les personnes m'ayant guidé lors de mon stage, notamment M. Ronan Querrec et M. Cédric Buche. Plus globalement, je remercie toutes les personnes du LI2/CERV pour leur sympathie et leur accueil.

Table des matières

1 Introduction

« Tu me dis, j'oublie.
Tu m'enseignes, je me souviens.
Tu m'impliques, j'apprends. »

Cette phrase de Benjamin FRANKLIN montre le défi que se donnent les recherches sur les environnements virtuels de formation.

En effet, depuis quelques années, la réalité virtuelle est utilisée pour l'éducation. Ceci est rendu possible par des avancées dans les recherches sur les environnements virtuels pour la formation (VET : *Virtual Environment for Training*). Les VET (c.f. figure 1), sont des environnements virtuels spécialement développés pour offrir un support à l'apprentissage. Car la réalité virtuelle présente de nombreux avantages pour l'apprentissage, comme il l'est décrit dans [Fuchs et al. 03b] et [Lourdeaux 01]. Comme cas le plus intéressant, nous pouvons citer la mise en situation quand la réalité ne le permet pas.

Cependant, les VET trop souvent basés sur les univers virtuels ont rapidement montrés des limites en terme d'usage et d'efficacité. Car malgré tous les avantages que présente la réalité virtuelle, et malgré le soin apporté au développement d'un système fait pour communiquer un savoir et une expérience à l'apprenant, l'aspect pédagogique a été négligé. Il manquait à ces systèmes des guides et des organisateurs dont le rôle est d'aider les utilisateurs-apprenants dans l'univers virtuel, non pas pour suppléer le rôle du formateur mais pour le compléter ou l'assister.

FIG. 1 – SécuRéVi : un exemple de VET [Querrec 02]

Pour répondre à ce besoin, les systèmes tutoriels intelligents (ITS : *Intelligent Tutoring Systems*) sont une des solutions proposées. Des recherches à ce sujet ont permis de développer des outils visant à simuler des compétences tutorales dans

la conduite d'interaction avec l'apprenant. Ces outils cherchent à reproduire les interactions observées entre un enseignant (ou formateur) et un apprenant (ou étudiant). Ou encore, les interactions existantes entre apprenants à l'intérieur d'un binôme. Selon [Woolf 92], les ITS sont décomposés en quatre modèles : le modèle du domaine (ou modèle de l'expert) [Anderson 88], le modèle de l'apprenant [VanLehn 88], le modèle pédagogique [Halff 88], et le modèle de l'interface :

- Le modèle du domaine représente la connaissance de l'expert sur le domaine. Il est aussi appelé modèle de l'expert. Ce modèle contient une représentation des connaissances à transmettre et doit être en mesure de proposer plusieurs solutions pour arriver à un objectif.

- Le modèle de l'apprenant apporte une mesure des connaissances de l'étudiant sur le problème. Il fournit un profil détaillé de l'étudiant. Et doit aussi donner toutes les informations relatives au comportement et aux connaissances de l'étudiant. Ce profil est mis à jour régulièrement en fonction des interactions qu'a l'étudiant avec le système.

- Le modèle pédagoqique permet de simuler le comportement qu'aurait un enseignant devant une situation pédagogique. Il se base sur les différences qui existent entre le modèle du domaine et le modèle de l'apprenant. Il doit proposer des assistances pédagogiques s'adaptant au niveau de l'étudiant.

- Le dernier modèle, le modèle de l'interface s'occupe de la communication entre l'étudiant et le reste du système. Il fait une traduction bidirectionnelle entre la représentation interne du système et une interface compréhensible par l'étudiant.

Des travaux de recherche menés par [Fuchs et al. 03b] sur la détection d'action ont permis de mettre au point les Primitives Comportementales Virtuelles (PCV) qui sont un découpage de toutes actions en activités élémentaires (voir section 2.1.3 page 10). Notre objectif est de présenter une solution d'implémenetation de PCV.

Dans ce rapport, nous commençons par faire un état de l'art sur le modèle de l'interface dans les environnements virtuels de formation déjà présent dans la littérature. Puis, nous présentons notre modèle, en détaillons chaque composant et en montrons un exemple d'application dans un des environnements virtuels de formation du laboratoire.

2 Étude biliographique

Dans cet étude bibliographique, nous commençons par donner une description du modèle de l'interface en montrant ce qu'il apporte à un système utilisant les ITS. Puis, nous détaillons les concepts importants à intégrer lors de l'élaboration du modèle de l'interface. Enfin, nous présentons quatre exemples de systèmes (un hypermédia, une application d'apprentissage et deux environnements de réalité virtuelle) et voyons comment ils ont tenté d'aborder le problème du modèle de l'interface.

2.1 Le modèle de l'interface

Dans cette partie, nous décrivons d'abord ce qu'est le modèle de l'interface en partant du concept général d'interface. Cette discussion nous amène à distinguer deux facettes du modèle de l'interface. La suite de cette partie détaille ces deux facettes en spécifiant quels sont les aspects importants à inclure dans le modèle.

2.1.1 Description

De manière générale, une interface peut être définie comme l'espace où l'interaction avec une personne physique et un objet prend place [Vigano et al. 03].

Dans le cadre des ITS, le modèle de l'interface rend le même service : il s'occupe de la communication entre l'étudiant et le reste du système et doit faire une traduction bidirectionnelle (c.f. figure 2) entre la représentation interne du système et une interface compréhensible par l'étudiant [Wenger 87]. C'est donc pour cela qu'il est important. Pourtant, il n'a pas encore été bien traité.

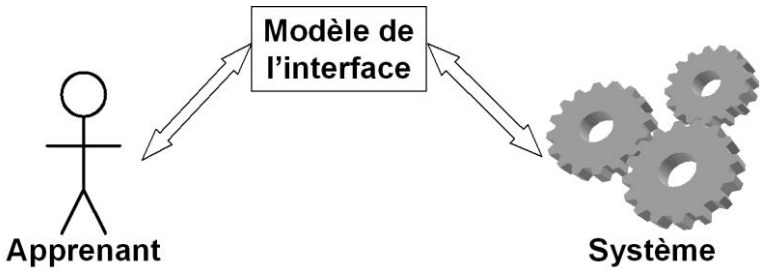


FIG. 2 – Communication bidirectionnelle apprenant/système

Étant dans un cadre d'apprentissage, il y a deux aspects à respecter :

– « L'esthétique et le degré de réalisme graphique ne constituent pas le point essentiel pour une application de réalité virtuelle pour l'apprentissage, [...]

les objectifs cruciaux concernent la fidélité des comportements d'apprentissage attendus de la part de l'utilisateur et l'atteinte des objectifs de formations »[Burkhardt et al. 03b] ;

– « L'interface ne doit pas imposer une charge d'apprentissage pour son utilisation qui obstruerait l'apprentissage réel » [Wenger 87]. L'interface doit utiliser l'avantage des conventions de communication existantes, comme le langage naturel, tout en en introduisant de nouvelles, tel qu'une souris [Vigano et al. 03].

A partir de ces informations, nous pouvons donc définir les deux sens de la traduction comme ceci :

– Du système vers l'apprenant : c'est l'ergonomie de l'interface, la manière de définir quel type de média utiliser pour traduire l'information du système. Un des problèmes qui se posent alors dans ce contexte est la difficulté d'adaptation aux utilisateurs. En effet, différents utilisateurs auront des comportements différents à divers types d'interface. Par exemple, certaines personnes seront plus réceptives à une interface purement visuelle, d'autres sonores, etc. Cependant, dans le cadre des ITS, le concepteur de l'interface pourra s'appuyer sur le modèle de l'apprenant pour tenter de découvrir à quel type de stimuli l'utilisateur sera le plus réceptif ;

– De l'apprenant vers le système : c'est la détection d'action/intention, la manière de capter les informations venant de l'utilisateur pour pouvoir les analyser dans le système. [Burkhardt 03] indique que la détection d'intention de l'utilisateur est un problème central pour le succès des interactions. Il est nécessaire et important de séparer la détection d'action de la détection d'intention, en effet :
  – L'apprenant peut avoir effectué la bonne action tout en ayant eu l'intention de l'effectuer ;
  – L'apprenant peut avoir effectué la bonne action sans en avoir eu l'intention ;
  – L'apprenant peut ne pas avoir effectué la bonne action et avoir eu l'intention d'en effectuer une bonne.

  Ce problème de détection d'action/intention est accru du fait de la difficulté d'adaptation au type d'interface.

2.1.2 Ergonomie pour la représentation

« Une image vaut mille mots. »
CONFUCIUS

Dans cette partie, nous définissons le modèle de l'interface d'un point de vue ergonomique. Nous partons des interfaces habituellement rencontrées dans la vie quotidienne, puis amenons la discussion dans le contexte de réalité virtuelle, en spécifiant ce que nous pourrions inclure dans le modèle de l'interface. Nous voyons alors qu'une formalisation s'impose et nous en citons un exemple.

Les interfaces en deux dimensions (2D) peuplent déjà notre environnement quotidien : dans les guichets de banques, dans les billetteries automatiques, etc. La plupart de ces interfaces sont basées sur le modèle WIMP (*Windows Icon Menu Pointer*).

Dans le cadre des environnements virtuels, l'interface ne s'arrête plus à l'écran. En effet, l'immersion dans un monde virtuel est une des définitions de la réalité virtuelle [Fuchs et al. 03a]. Cette immersion introduit déjà implicitement une interface. Mais dans la plupart des applications de réalité virtuelle conçues auparavant, les interfaces ont souvent été développées sans utiliser une méthodologie formalisée, préférant développer l'interface en fonction des besoins [Vigano et al. 03]. Pourtant, [Fuchs et al. 01] définit que « de l'ergonomie de l'interface dépend la qualité d'un système ». En effet, si la manipulation « naturelle », à priori simple, d'objets virtuels est impossible, le système sera totalement inutile pour le but qu'il doit rendre (c.-à-d. être un système pour l'apprentissage).

Pour la conception de l'interface graphique, il faut prendre en compte que des systèmes focalisés sur le réalisme (essentiellement visuel) sont souvent peu flexibles, avec une scène de base où la gestion se fait pour l'essentiel objet par objet, de façon inerte [Burkhardt et al. 03b]. Le réalisme même, indroduit un problème puisque celui-ci est basé sur l'intuition, et personne ne partage la même définition ni les mêmes critères [Burkhardt et al. 03a]. Un autre exemple concerne les primitives d'affichage ou de dialogue qui ne sont pas toujours adaptées aux besoins qu'engendre la gestion d'une interaction intelligente entre l'utilisateur et le système. Dans le même ordre d'idées, la lourdeur d'un bon nombre d'applications actuelles engendre souvent des modes d'interactions limités. Cette lourdeur est due en partie au poids du calcul graphique ou à la gestion des données associées aux interfaces de retour d'effort et à la simulation du mouvement.

Un des buts à atteindre est donc de la rendre totalement transparente à l'utilisateur, en lui permettant d'agir en se concentrant sur les actions qu'il a à faire et non pas sur la façon d'agir. Pour arriver à ce résultat, il peut sembler utile de jouer sur l'impression de « déjà vu » de l'utilisateur. Un problème est celui de l'humanisation de l'objet graphique pour lui donner la capacité de converser avec la personne qui doit l'utiliser [Vigano et al. 03]. Il faut tenter de développer l'interface dans le but de suggérer l'action correcte, en traduisant à l'utilisateur non expérimenté sa fonction immédiate. Le problème de communication surgit de manière plus importante lorsque différents types de médias sont mélangés, donnant ainsi lieu à une sorte

d'interface hybride mélangeant les diverses caractéristiques de chacun des médias. Cette idée d'interface hybride est utile aux VET. En effet, il faut profiter du fait que nous sommes dans un environnement virtuel pour superposer une information d'assistance aux scènes de l'environnement virtuel : flèches, sons, clignotement, etc.

Comme il n'existe pas ou peu de formalisation des interfaces graphiques en réalité virtuelle, (sous-entendu, les environnements 3D) tout comme les interfaces graphiques ont été formalisés pour les environnements 2D, [Vigano et al. 03] propose une classification de ce type d'interface en trois niveaux :

- Niveau 1 : le niveau où les objets graphiques sont leurs propres interfaces. Lorsqu'une forme à trois dimensions décrit elle-même la fonction, et l'action, l'objet lui-même peut finalement être classé comme un élément de l'interface (comme le bouton marche/arrêt sur le panneau de commande d'une machine à laver) ;

- Niveau 2 : le niveau où il y a une relation entre l'utilisateur et le reste de l'environnement virtuel. Dans ce niveau, il y a un ensemble d'outils et d'infrastructures 3D qui autorisent l'utilisateur à intéragir avec l'objet virtuel (par exemple, un menu contextuel sur le bouton) ;

- Niveau 3 : le niveau abstrait de l'interface. C'est le niveau d'interface pour les opérations externes à l'environnement virtuel. Le niveau 3 n'a pas obligatoirement besoin d'être développé en 3D et de présenter une immersion. C'est dans ce cas que les modèles 2D peuvent s'appliquer.

S'il est une dernière chose à retenir, c'est que :« le bénéfice de l'immersion n'a pas réellement de confirmation empirique » [Burkhardt et al. 03b] pour un VET. Le degré d'immersion dans l'environnement virtuel peut entraîner une distraction de l'attention des apprenants du contenu du moins dans les premières immersions [Burkhardt et al. 03a].

2.1.3 Détection d'actions

Dans les applications classiques de réalité virtuelle, la détection d'actions est souvent associée à l'utilisation de périphériques tel que des gants de données, souris à trois dimensions, capteurs magnétiques ou optiques, etc. Diverses améliorations dans le développement de tels périphériques ont permis d'obtenir une meilleure précision des données ainsi qu'un usage facilité. Cependant, l'utilisation de tels outils entrainent forcément un temps d'adaptation.

Dans le cadre des VET, l'apprentissage est au centre du système, [Fuchs et al. 01] nous dit que le but à atteindre n'est pas l'interaction ni l'immersion totale avec l'en-

vironnement. La précision des données des capteurs prend alors une importance moindre. Par exemple, ce n'est pas la position exacte de l'apprenant dans l'environnement qui est importante, mais plutôt de savoir si l'apprenant se trouve dans une zone qui correspondrait à une action importante pour la formation [Lourdeaux 01]. Idem lors d'interactions avec un objet de l'environnement, ce n'est pas le fait que nous sachions que l'apprenant ait l'objet dans sa main droite qui soit important, mais plutôt le fait de savoir que l'apprenant ait utilisé cet objet dans l'action qui correspond à la tâche qu'il avait à réaliser.

Pour illustrer ce concept, nous pouvons nous inspirer de la communication non-verbale entre avatars, c.-à-d. la communication entre des représentations virtuelles d'êtres humains et/ou des représentations virtuelles d'agents de l'environnement. La manière habituellement utilisée est la logique floue [Gardie 01]. Nous pouvons prendre l'exemple des agents de guidage au sol des avions. Ces agents utilisent des signes pour dialoguer avec les pilotes. Dans les deux cas, c.-à-d. dans le cas des environnements réels et dans le cas des environnements virtuels, ces gestes n'ont pas à être d'une précision extrême. La transmission d'information se fait par la reconnaissance de geste (par exemple, le bras de l'agent se déplace du bas vers le haut avec une grande amplitude) et non pas par reconnaissance de position exacte. Pour arriver à un tel résultat, il serait intéressant de décomposer les mouvements.

Lorsque l'apprenant se trouve en environnement virtuel, il a une ou plusieurs activités à accomplir. Celles-ci peuvent se décomposer en activités élémentaires que [Fuchs et al. 03b] propose d'appeler Primitives Comportementales Virtuelles (PCV). Ces primitives peuvent être regroupées en quatre catégories décomposées elles-même en sous-catégories :

- Observer le monde virtuel ;
- Se déplacer dans le monde virtuel ;
- Agir sur le monde virtuel ;
- Communiquer avec autrui.

[Fuchs et al. 03b] introduit aussi le concept d'Aides Logicielles Comportementales (ALC). Les ALC, associées aux PCV, apportent des aides dans la tâche de l'utilisateur dans le monde virtuel, au niveau de la motricité et des perceptions programmées. Les ALC peuvent être déterminées en fonction des « affordances » des objets ou de l'environnement autour du sujet. D'après la théorie de [Gibson 79], « une affordance représente les interactions possibles entre l'objet et le sujet ». Ces interactions sont perçues par le sujet par rapport à l'idée qu'il se fait de la fonction de l'objet, plus qu'il ne perçoit les caractéristiques physiques, géométriques, etc., de l'objet ou de l'environnement. Il est donc plus utile de savoir par avance à quoi un élément de l'environnement va servir que d'avoir des notions précises de sa géométrie ou de ses caractéristiques physiques. Des chercheurs se basent sur cette approche pour analyser le comportement du sujet dans un environnement virtuel

[Morineau et al. 01].

Comme [Lourdeaux 01] l'indique et comme nous l'indiquions en introduction, afin de proposer des aides pour l'apprenant, il est souvent utile de détecter les intentions des utilisateurs. Par exemple, pour positionner automatiquement l'utilisateur à un endroit précis, il faut être sûr que celui-ci veuille bien s'y rendre. Nous détecterons, par exemple, non pas l'entrée dans une zone, mais l'arrêt avec une orientation donnée dans la zone.

En résumé, nous pouvons retenir les éléments suivants comme étant importants à inclure lors de la conception du modèle de l'interface :

- Une représentation graphique non pas centrée sur l'hyper-réaliste mais sur l'action d'apprentissage, pour ne pas perdre de vue que nous concevons un environnement d'apprentissage et non un jeu vidéo ;

- Une interface adaptée en fonction de chaque type d'objets (les 3 niveaux de [Vigano et al. 03]) et transparente pour l'apprenant afin de ne pas ajouter un apprentissage de l'interface pour le formé ;

- Une détection d'action/intention efficace qui ne s'embarasse pas des détails (position exacte d'un avatar) afin de savoir exactement ce que l'apprenant fait et envisage de faire, et de réagir en fonction.

2.2 Exemples de modèle d'interface

Dans cette partie, nous présentons quatre exemples de systèmes : un hypermédia, une application d'apprentissage et deux environnements de réalité virtuelle. Nous tâchons de voir comment ils ont tenté d'aborder le problème du modèle de l'interface et quel sont les avantages et les inconvénients de chaque solution.

2.2.1 MÉTADYNE (hyperMÉdia adapTAtif DYNamique pour l'Enseignement) [Delestre 00]

MÉTADYNE, est un hypermédia adaptatif dynamique. Un hypermédia regroupe les outils permettant la conception et la mise en page de présentations multimédias destinés à Internet, il est généralement constitué de pages et de liens. Par adaptatif, nous entendons le fait d'adapter le contenu des pages de l'hypermédia en fonction des caractéristiques, des volontés et des buts de l'utilisateur. Ainsi, si deux utilisateurs avec des profils différents accèdent à une même page, ils devront visualiser deux contenus différents. Le dernier terme, dynamique, ajoute l'idée d'adaptation instantanée. Le système n'est plus constitué de pages et de liens prédéfinis qui s'adaptent à l'utilisateur, mais les pages et les liens sont contruits dynamiquement. Dans les hypermédias, l'interface se présente sous la forme d'une page web.

MÉTADYNE se base sur le concept d'ITS. Nous considèrons que l'hypermédia s'appuie sur le modèle du domaine et le modèle de l'apprenant pour s'adapter à ce dernier. Le modèle du domaine permet de définir l'architecture globale du système. Le modèle de l'apprenant sert de filtre pour sélectionner les divers documents à présenter à l'utilisateur.

Dans MÉTADYNE, le modèle de l'interface est présenté sous le nom de générateur de cours. Il s'appuie principalement sur le modèle de l'apprenant. Lorsque l'apprenant clique sur un lien hypertexte, le générateur de cours va récupérer le profil de l'apprenant. En fonction de ce profil, le générateur de cours va sélectionner le meilleur média.

Pour cela, le concepteur de MÉTADYNE utilise un concept de briques élémentaires (c.f. figure 3). Pour choisir la meilleure brique élémentaire, le générateur de cours va appliquer trois filtres. Le premier permet d'extraire un sous-ensemble de briques pour un type cognitif déterminé. Le second permet d'effectuer l'opération mais pour un ensemble de niveau cognitif déterminé. Enfin, le troisième effectue le même genre d'extraction mais pour un type physique donné.

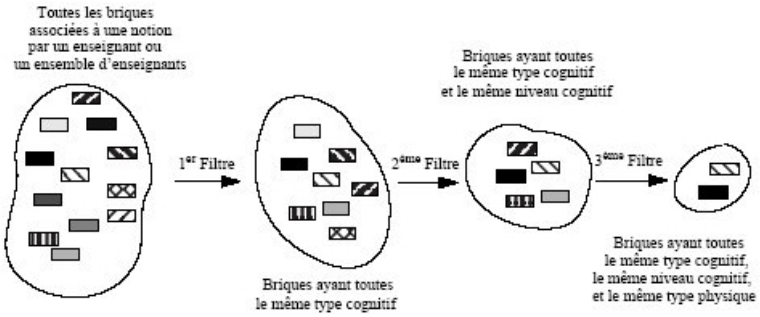


FIG. 3 – MÉTADYNE : les briques élémentaires

Une fois le média, c.-à-d. la page, construite, il faut que le système détermine les liens hypertextes permettant à l'utilisateur d'accéder à d'autres notions. Cette séléction se base principalement sur le modèle du domaine.

La représentation finale des liens prend alors la forme montrée dans le TAB. 1. Pour chaque type de relation du modèle du domaine correspond un type de lien hypertexte.

Finalement, nous pouvons voir que le générateur de cours de MÉTADYNE joue le rôle de modèle de l'interface tout en s'étendant à un des aspects du modèle pédagogique. En effet, c'est celui-ci s'occupe de faire la différence entre le modèle du

Relation du modèle du domaine	Type de liens hypertexte
Pré-requis	Page d'index
Analogie	Page d'index
Conjonction	Liens précédent et suivant
Disjonction forte	Page de choix

TAB. 1 – MÉTADYNE : représentation finale des liens

domaine et le modèle de l'apprenant et de fournir à l'apprenant des pages adaptées à son niveau. L'interface ici, n'est vue que du côté liens hypertextes.

2.2.2 INES (Intelligent Nursing Education Software) : un système d'enseignement générique à agents basé sur les ITS [Hospers et al. 03]

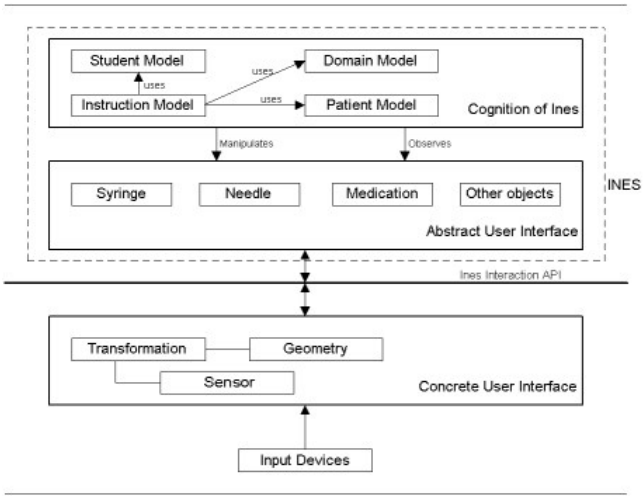


FIG. 4 – Architecture système d'INES

Le but principal du projet INES est de fournir un environnement d'enseignement efficace. Les auteurs considèrent que pour qu'un environnement soit efficace comme environnement d'enseignement, il doit remplir deux conditions : première condition, l'environnement doit fournir un système d'aide logique et correcte, c.-à-d. que le système d'enseignement doit être capable de fournir de la correction d'erreur (*feedback*), des démonstrations et des explications ; deuxième condition, l'interface utilisateur du système doit être optimale et accessible autant que cela puisse être possible. Dans ce but, un système pourrait, par exemple, utiliser une interface 2D mais l'interface pourrait aussi être un environnement 3D de très grande

qualité. Pour l'instant, le projet INES s'est plus concentré sur la première condition.

INES est un système d'enseignement générique dans ce sens où le noyau dur est séparé des modules d'exercices et des interfaces utilisateur. Les exercices se présentent sous la forme de modules séparés. Ajouter une nouvelle tâche à INES signifie ajouter une nouvelle interface. La nouvelle interface peut être basée sur du texte ou de la 2D, mais peut aussi être un environnement 3D. Même des périphériques à retour de force ou un casque peuvent être utilisé sans faire aucun changement au système INES. Le système implémenté est une application à l'entraînement des infirmières.

La figure 4 montre les principales parties du système INES. Le modèle du patient (*patient model*) et les objets dans l'interface utlisateur abstraite (*abstract user interface*) sont spćialement faites pour les exercices. Pour d'autres exercices, d'autres modèles et objets peuvent être utilisés. L'architecture d'INES consiste en quatre parties : les entrées de périphériques (*input from devices*), une interface utilisateur concrète (*concrete user interface*), une interface utilisateur abstraite (*abstract user interface*) et la cognition d'INES (*cognition of INES*).

- La première partie (*input from devices*), s'occupe des périphériques du monde réel (gants de données, souris, clavier, périphériques haptiques, etc.). Le type de périphérique n'est pas important puisqu'INES attend juste les données nécessaires provenant de interface concrète (*concrete user interface*).

- L'interface utilisateur concrète s'occupe de la visualisation de l'exercice. Cette partie nécessite un développement par les concepteurs de l'exercice, ce qui permet une grande souplesse puisque n'importe quel type d'interface peut être développé (2D ou 3D).

- L'interface utilisateur abstraite (*abstract user interface*) est un modèle haut niveau de l'interface concrète. L'interface concrète dialogue directement avec l'interface abstraite.

- La quatrième partie concerne les autres modèles des ITS.

INES est donc le parfait exemple d'ITS n'utilisant pas le modèle de l'interface tel qu'il est décrit plus haut. En effet, les concepteurs d'INES laissent aux utilisateurs de ce système le soin de développer eux-mêmes leurs propres interfaces.

2.2.3 HAL (Help Agent for Learning) : un agent pédagogique intelligent [Lourdeaux 01]

HAL est un agent pédagogique intelligent basé sur les ITS qui permet de doter un environnement virtuel d'un moteur d'intelligence artificielle capable de s'adapter

aux styles d'apprentissage de chaque formé et de leur offrir des aides adaptées.

HAL est capable d'aider le formateur à détecter les erreurs ou les non-actions des formés et les causes supposées. Il analyse le comportement du formé (raisonnement, connaissance, utilisation des interfaces) en temps réel. HAL informe le formateur de l'historique du comportement du formé, de ses erreurs, des causes et des décisions pédagogiques qu'il prendrait. Cependant, HAL n'a pas pour but de remplacer le formateur.

Pour arriver à ce résultat, HAL s'appuie sur une approche multi-agents et utilise la technique de la planification (c.-à-d. la construction de plan) habituellement utilisée en Intelligence Artificielle (IA) classique. La planification permet au système de simuler dans un monde fictif le déclenchement des tâches nécessaires pour atteindre un but donné à partir d'un état initial. La planification est liée à la notion de décomposition. Dans le cadre de HAL, cette décomposition est hiérarchique.

Pour évaluer les actions du formé, HAL observe ses interactions avec le système (manipulation, déplacements, temps écoulé, utilisation des interfaces, événement du scénario, etc.). Ensuite, HAL analyse ces interactions et leur donne un sens dans l'environnement d'apprentissage, c.-à-d. une action correspondante.

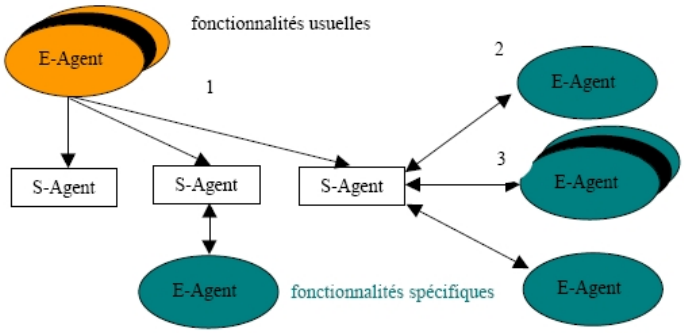


FIG. 5 – HAL : modèle

HAL possède deux types d'agents (c.f. figure 5). Les premiers gèrent l'exécution du scénario par le formé (S-Agents). Et les seconds gèrent la communication avec l'environnement virtuel (E-Agents).

Les S-agents sont des agents qui gèrent une partie de l'analyse de l'activité du formé, une partie de l'expertise pédagogique et une partie du diagnostic. Chaque S-Agent représente une tâche à réaliser par le formé. Ses tâches peuvent être décomposées en plusieurs sous-tâches qui, elles-mêmes peuvent être décomposées en

sous-tâches élémentaires. Ces sous-tâches élémentaires sont basées sur le concept de PCV (c.f. page 10).

Un S-agent sait comment une tâche doit être réalisée et quels sont les principaux problèmes qui peuvent être rencontrés. Les S-agent sont capables de coopérer avec d'autres S-agents pour parvenir à un but.

Enfin, les S-Agents peuvent détecter les intentions du formé. Ainsi, un S-Agent est déclenché quand le formé essaye de réaliser la tâche (par la détection des positions, déplacements, orientations dans des zones et non par la détection la position exacte ou de l'objet pris).

Pour recueillir des informations sur les actions de l'apprenant pendant l'exécution de la tâche, les agents d'environnement (E-Agents) observent les interactions avec l'environnement virtuel. Ils n'observent cependant que les interactions correspondant aux fonctionnalités ayant un sens et un intérêt pour les S-Agent, c.-à-d. :

- Les fonctionnalités concernant tous les S-Agents. Il s'agit de la position et de l'orientation du formé et de sa main ainsi que du mode déplacement/manipulation. Ces fonctionnalités sont observées continuellement. Un E-Agent est créé régulièrement et envoie ses observations à tous les S-Agents ;

- Les fonctionnalités spécifiques demandées par un S-Agent particulier pour identifier un comportement (quel objet se trouve dans la main du formé), une connaissance contextuelle (y a t'il du brouillard) ou une connaissance sensori-motrice particulière (e.g. la main du formé est-elle fermée). Ces fonctionnalités sont demandées ponctuellement et temporairement par des S-Agents. Un E-Agent est donc créé ponctuellement pour observer une fonctionnalité particulière et envoie ses observations au S-Agent demandeur.

HAL base donc la détection d'action sur un envoi de message entre des agents de l'environnement qui observent continuellement les actions de l'utilisateur. Cette approche fonctionne très bien puisque HAL « filtre » les informations reçues en fonction du scénario d'apprentissage.

2.2.4 STEVE (Soar Training Expert for Virtual Environments) [Rickel et al. 98] [Johnson et al. 00]

STEVE (c.f. figure 6) est un agent autonome animé implémenté en Soar [1] qui cohabite dans le monde virtuel avec les étudiants. STEVE contrôle continuellement l'état du monde virtuel dans lequel il évolue en le manipulant périodiquement via

[1] Architecture et langage d'intelligence artificielle développé pour donner un modèle général de la reconnaissance humaine [Laird et al. 87] [Newell 90]

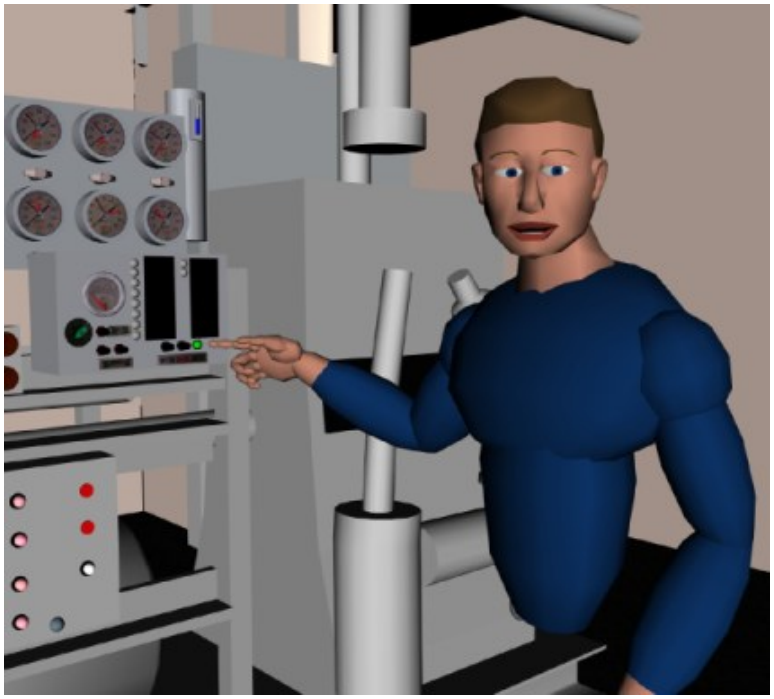


FIG. 6 – STEVE : STEVE montrant le bouton d'allumage d'une lampe [Rickel et al. 98]

un moteur virtuel d'action. L'objectif de STEVE est d'aider les étudiants à apprendre à exécuter des tâches physiques basées sur des procédures.

Il peut montrer comment réaliser une tâche en expliquant ses actions. Il peut aussi contrôler les étudiants en train d'exécuter des tâches, et fournir de l'aide lorsque les étudiants en ont besoin.

Les informations sur l'état du monde que recueillent STEVE sont maintenues par le simulateur. Le module de perception de STEVE qui représente ces états en une paire attributs-valeurs. Chaque attribut représente une variable d'état de l'environnement et la valeur associée représente la valeur de cette variable. Le module de perception regarde l'état du simulateur en écoutant les messages provenant du simulateur (*message dispatcher* figure 7). En fonction du scénario choisi, STEVE donnera une liste d'attributs intéressants. Lorsque l'état des attributs intéressants change, le *message dispatcher* diffuse alors les nouvelles valeurs associées.

Le module de perception (c.f. figure 7) utilise ces messages pour conserver un aperçu du système à un instant donné (*snapshots*). Le module cognitif peut alors recevoir ces *snapshots* dès lors que le module de perception reçoit des nouvelles informations de l'environnement sauf en cas d'actions simultanées. Prenons un

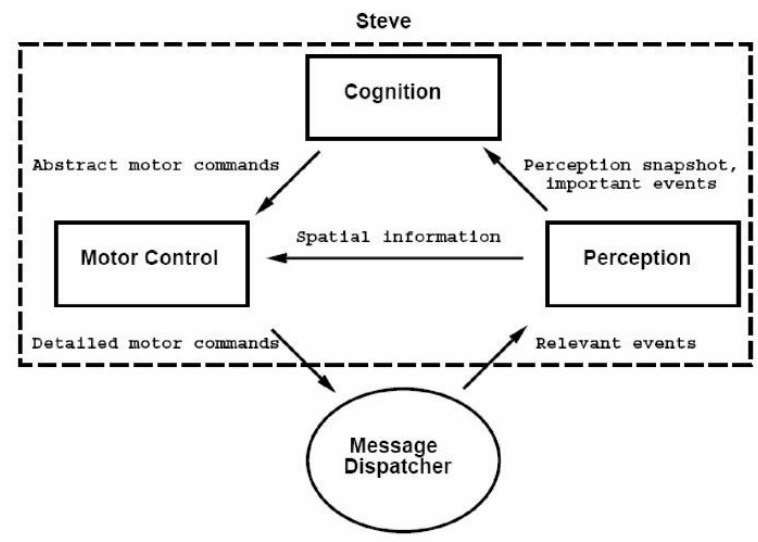


FIG. 7 – STEVE : modèle

exemple : supposons qu'une lampe s'allume lorsqu'un bouton est relâché. Lorsque l'utilisateur va appuyer sur le bouton, puis le relacher, le simulateur va alors envoyer simultanément deux messages, un spécifiant que le bouton est relâché, et un autre spécifiant que la lampe est allumée. Le module cognitif de STEVE ne recevra alors un *snapshot* que lorsque le module de perception aura reçu les deux messages, même si celui-ci demande une mise à jour durant un instant situé entre la réception des deux messages. Pour arriver à ce résultat, les concepteurs de STEVE ont mis en place un système de présentation de début et de fin de message. Ce comportement rappelle les transactions dans une base de données. Les états ne peuvent être mis à jour que de manière atomique.

STEVE utilise un peu la même approche que HAL dans le sens où il y a d'abord une phase de détection, puis une comparaison avec un objectif. Dans HAL, le scénario est déjà pré-découpé en sous-parties et en actions élémentaires. Cependant, dans STEVE, c'est le module de reconnaissance qui lui va créer le scénario pour arriver à l'objectif final. De plus, STEVE va plus loin que la détection d'actions. Il communique avec l'apprenant en utilisant la parole (grâce à deux modules de reconnaissance et de génération de parole). Il montre les objets. Il a été développé pour simuler le comportement qu'auraît un enseignant réel.

2.3 Conclusion

Dans cette étude bibliographique, nous nous sommes interessés au modèle de l'interface dans les ITS. Dans un premier temps nous avons décrit ce qu'étaient les VET et les ITS. Ensuite, nous avons détaillé ce que le modèle de l'interface pouvait

apporter de plus à un VET. Enfin, nous avons présenté des exemples de systèmes adaptant les ITS à leurs besoins.

Au travers des divers exemples proposés, nous avons pu voir que le modèle de l'interface tel que nous l'avons défini n'est pas réellement utilisé. Certains systèmes, tel qu'INES, n'incluent pas de modèle de l'interface du tout mais fixe les informations qu'il souhaite récupérer. D'autres, comme MÉTADYNE n'ont pas à se soucier d'un modèle de l'interface bien définit puisque qu'ils sont basés sur des hypermédias avec une interface déjà définie (pages web et liens).

Les approches de HAL et de STEVE emploient des agents utilisant les variables de l'environnement et des scénarios pour détecter les actions/intentions de l'utilisateur. Ces approches semblent intéressantes dans le cadre des environnements virtuels immersifs. Pour ce qui est de l'ergonomie, seul STEVE semble montrer un réel avantage au niveau pédagogique, avec son principe d'agent humanoïde attentif en permanence aux actions de l'apprenant.

L'objectif du stage de DEA est de formaliser le modèle de l'interface et de l'intégrer dans les travaux en cours sur les VET au sein du laboratoire et de l'appliquer aux VET déjà réalisés. Les PCV tels que nous les avons vu dans cette partie semble être intéressant pour la détection d'action et c'est donc à partir de cette base que nos travaux vont se tourner.

3 Modèle

Notre problème est la détection d'action dans les environnements virtuel d'un point de vu pédagogique. Comme nous l'avons vu dans la partie précédente, les Primitives Comportementales Virtuelles définies par [Fuchs et al. 03b] semblent intéressantes. En effet, celles-ci ont déjà été utilisées par [Lourdeaux 01] dans le développement d'un environnement virtuel de formation. Le modèle de l'interface que nous proposons est donc basé sur les PCV. Dans cette partie, nous commençons par décrire l'environnement dans lequel notre modèle va s'intégrer. Puis nous décrivons en détail chaque partie de notre modèle.

3.1 L'environnement virtuel de formation

Notre modèle à été développé à l'aide la plateforme ARéVi (Atelier de Réalité Virtuelle [Reignier et al. 98]). ARéVi est une bibliothèque de simulation et de rendu 3D écrite en C++ utilisant la bibliothèque OpenGL. Elle est développée sous la responsabilité du projet ARéVi au Centre Européen de Réalité Virtuelle (CERV). Le projet ARéVi a pour objectif de définir et de développer des méthodes et des outils de prototypage interactif, collaboratif et coopératif pour la mise en œuvre d'applications en environnements virtuels distribués. Elle permet l'intégration d'entités virtuelles pouvant être considérées comme des agents.

L'environnement dans lequel notre modèle va être développé est multi-agents. Les systèmes multi-agents (SMA) sont historiquement fondés sur l'intelligence artificielle classique dont la principale utilisation dans les outils de formation est la conception de tuteurs intelligents. La notion de distribution de l'intelligence, qui fonde les systèmes multi-agents, se retrouve également dans les outils de formation actuels. Les SMA sont composés d'agents, des entités autonomes qui sont douées de capacités de perception, de décision et d'action. Les agents sont déjà utilsés dans les VET du CERV (c.f. [Querrec 02]).

De plus, notre modèle s'inscrit dans le cadre des environnements virtuels de formation (VET : *Virtual Environment for Training*) du CERV. Dans ce sens, il communique avec d'autres modules déjà développés. Nous ne présentons ici que très rapidement les éléments avec lesquels nous communiquons. Ces modules sont responsables des autres ITS du système et sont des agents (l'organisation pédagogique est présentée dans [Buche et al. 04]) :

- le *Learner Manager* (figure 8 page 21) : ce module sert à mettre à jour le modèle de l'apprenant des ITS en fonction des diverses informations que lui transmettent les autres modules ;
- le *Pedacogical Agent* : il prend les décisions pédagogiques en fonction des informations qui lui sont transmises. Ce module s'appui sur le modèle pé-

dagogique et sur le modèle de l'apprenant des ITS afin de prendre une décision ;
– l'*Application Manager* : il effectue les requêtes d'action dans l'environnement virtuel.

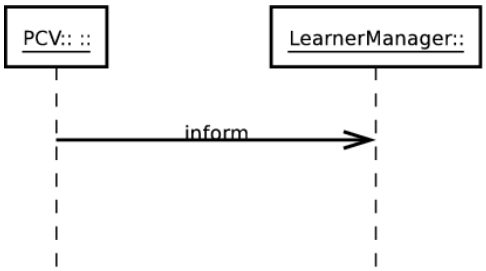


FIG. 8 – Le *Learner Manager* (notation UML)

3.2 Les Primitives Comportementales Virtuelles

Selon [Fuchs et al. 03b], « dans toutes les applications de réalité virtuelle, les activités de l'utilisateur sont toujours décomposables en quelques comportements de base appelés "Primitives Comportementales Virtuelles (PCV)" ». L'auteur les regroupe en quatre catégories :

– observer le monde virtuel ;
– se déplacer dans le monde virtuel ;
– agir sur le monde virtuel ;
– communiquer avec autrui ou avec l'application.

Nous avons donc choisi de découper notre modèle en quatre modules qui correspondent à chaque PCV. La figure 9 présente le *package* PCV tel que nous l'avons conçu.

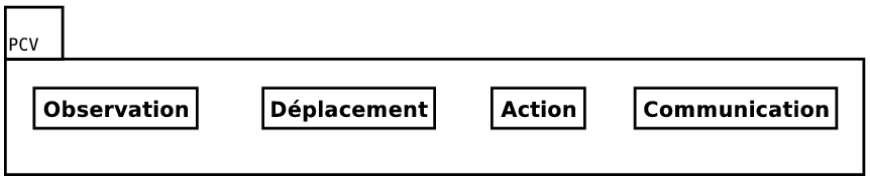


FIG. 9 – Le *package* PCV (notation UML)

Par la suite, nous présentons ces quatre *packages* et montrons quelles informations ils construisent, manipulent et communiquent aux autres agents du système.

3.3 La PCV d'observation

Dans le cas de l'observation du monde virtuel, [Fuchs et al. 03b] décrit plusieurs sous catégories de PCV :

- selon que l'observation soit visuelle, auditive, tactile ou réalisée par combinaison de ces sens ;
- selon qu'il s'agit d'interpréter l'environnement ou de s'orienter par rapport à celui-ci.

La PCV d'observation telle que nous l'avons définie va permettre d'obtenir une liste des objets qui apparaissent dans le champ de vision de l'utilisateur ainsi que leurs distances et leurs orientations par rapport à l'utilisateur. La liste des objets obtenue est en fait une liste de nom d'objets. En effet, étant donné qu'il est possible que le VET soit dans un environnement distribué, il semble judicieux de ne transmettre que des chaînes de caractères plutôt que des zones de mémoire. Ce modèle ne couvre cependant pas l'observation auditive ni tactile, mais peut être étendu en adaptant les périphériques d'entrée.

Les informations reçus sont utiles lorsqu'il est spécifié dans le scénario d'apprentissage que l'apprenant doit d'abord observer l'environnement avant de commencer à effectuer la tâche requise. Le fait de connaître l'orientation nous permet précisement de savoir si un objet est simplement apparu dans le champ de vision de l'utilisateur, ou si l'utilisateur en a vu la face principale (par exemple, d'un panneau d'affichage avec une face contenant des indications et une face arrière ne contenant aucune information).

Afin de pouvoir s'adapter à tous les utilisateurs, la PCV d'observation va s'appuyer sur le modèle de l'apprenant pour obtenir des données physiologiques nécessaires (profondeur et largeur du champs de vision de l'utilisateur). La figure 10 montre les services de la PCV d'observation.

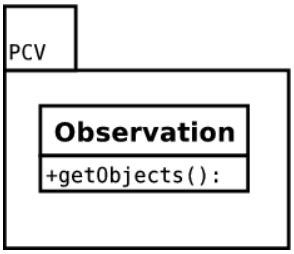
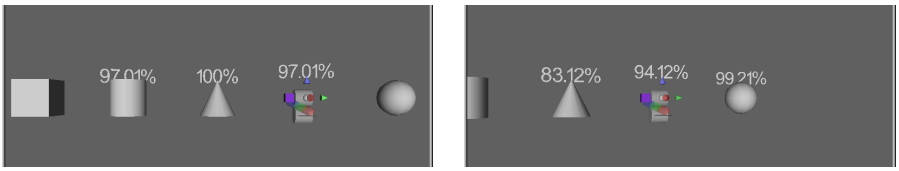


FIG. 10 – Diagramme de classe de la PCV d'observation (notation UML)

3.3.1 Illustration

Dans l'exemple illustré par la figure 11, nous avons disposé des objets dans un environnement. La PCV d'observation analyse en permanence l'environnement avec ici un champ de vision réduit à un angle de $\frac{\pi}{2}$ (choisit de manière arbitraire). Les informations sur la visibilité des objets sont envoyées directement à l'environnement et le pourcentage est affiché au dessus de chaque objet (11 (a)). Lorsque l'utilisateur se déplace, la visibilité des objets change et l'affichage du pourcentage est mis à jour (11 (b)).



(a) (b)

FIG. 11 – Exemple d'utilisation de la PCV d'observation

3.3.2 La PCV d'observation dans le reste du système

Dans les VET du CERV, c'est le *Learner Manager* qui à l'initiative de faire appel aux services de la PCV d'observation. En effet, un appel à la méthode `getObjects()` est très couteux en calcul puisqu'elle utilise un lancer de rayon. Avec un environnement contenant un faible nombre d'entités, cette méthode peut être utilisée de manière continue, cependant, plus le nombre d'entités augmente, plus l'observation sera lourde (c.f. figure 12).

Lorsque le *Learner Manager* á fait sa demande, la PCV observation lui renvoie la liste des noms d'objets, des distances et des orientations. Ces informations vont être communiquées au *Learner Manager* afin qu'il mette à jour le modèle de l'apprenant.

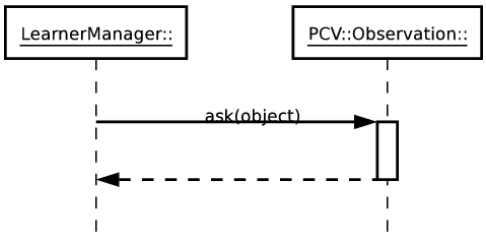


FIG. 12 – Fonctionnement de la PCV d'observation (notation UML)

Les figures 13 et 14 (page 25) montrent des implémentations du modèle d'interface dans un VET développé au CERV. On voit l'utilité de la fonction de filtrage des entités. En effet, l'environnement virtuel est composé entièrement d'objets. Nous pouvons donc isoler une certaine classe d'entité pour se focaliser sur les objets de la situation pédagogique.

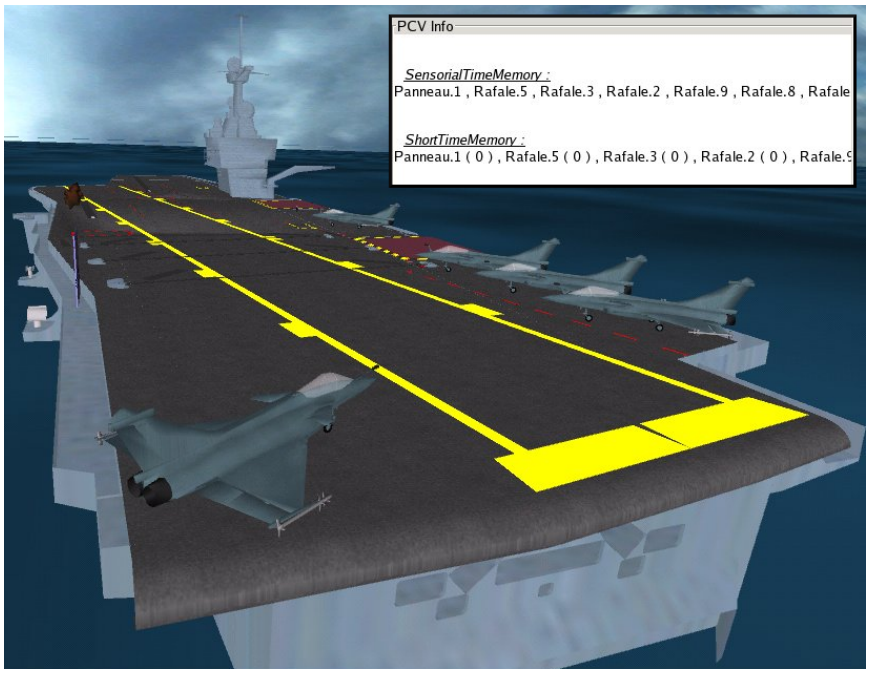
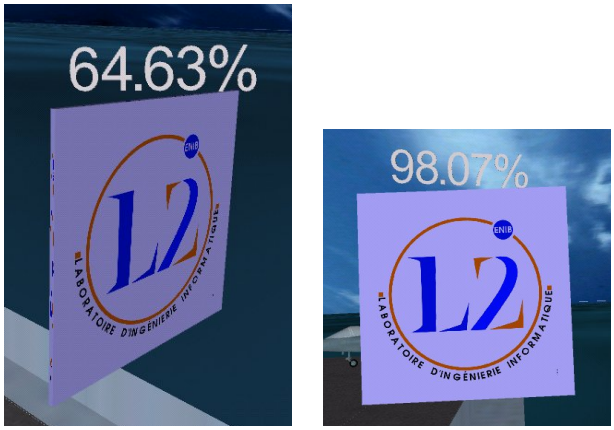


FIG. 13 – La PCV d'observation dans un VET (1)

Dans le cas du panneau d'affichage nous pouvons savoir si l'apprenant voit les informations de face ou s'il les entraperçoit.



(a) (b)

FIG. 14 – La PCV d'observation dans un VET (2)

3.4 La PCV de déplacement

Dans le cas d'un déplacement, [Fuchs et al. 03b] définit aussi plusieurs sous catégories de PCV, selon le type de déplacement : trajectoire à 1 dimension (sur une droite ou sur une courbe), déplacement sur une surface (plane ou non) ou dans l'espace. Le déplacement peut se faire avec ou sans changement d'orientation de l'utilisateur.

Lors de l'exécution d'un scénario, il n'est pas nécessaire de connaître en permanence la position exacte de l'apprenant dans l'environnement. En revanche, il est utile de savoir, lors d'une tâche précise, si l'apprenant se déplace vers ou en dehors du lieu où se déroulera l'action, si l'apprenant s'approche d'un objet précis et à quelle allure.

Afin de répondre à ces besoins, la PCV de déplacement (diagramme de classe figure 15 page 26) est mise en œuvre. Les diverses méthodes contenues dans la PCV de déplacement permettent :

- d'obtenir l'orientation moyenne `getOrientation()` (c.-à-d. l'orientation qu'a l'utilisateur depuis un point de départ donné `setStartingPoint()`) ;
- de savoir quel est l'objet vers lequel l'utilisateur se dirige globalement (cette méthode utilise l'orientation moyenne) `findObject()` ;
- de connaître la vitesse instantanée de l'utilisateur `getSpeed()` ;
- de définir des zones `addZone()` et de savoir à quelles distances (sous forme de variables floues : au dehors, loin, près, au dedans) de ces zones l'utilisateur se trouve `checkZones()` ;

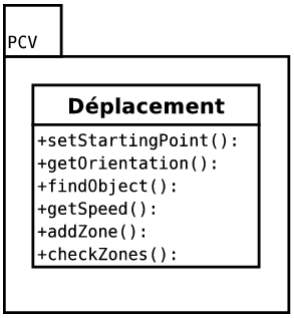
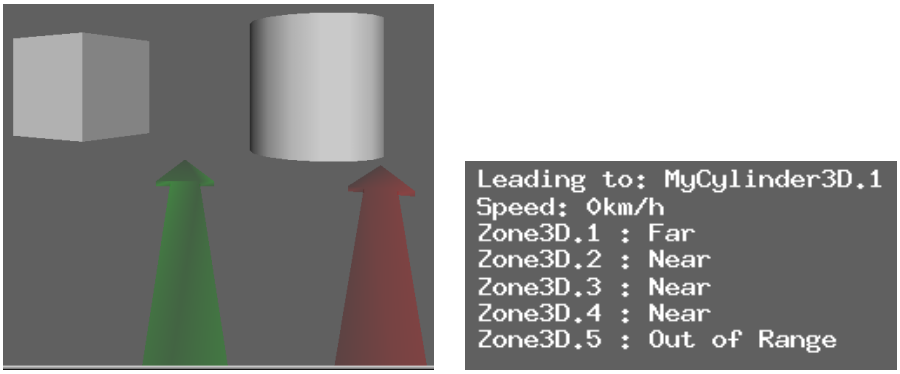


FIG. 15 – Diagramme de classe de la PCV de déplacement (notation UML)

3.4.1 Illustration

Dans l'exemple suivant (figure 11), nous avons disposé des objets dans un environnement et défini des zones autour de chaque objet. Lors de l'approche d'une zone, l'information de proximité (figure 11 (b)) se met à jour. La direction instantanée de l'utilisateur est représentée par une flêche verte et la direction moyenne (c.-à-d. la direction qu'a pris l'apprenant depuis un point de départ donné) par une flêche rouge (figure 11 (a)).



(a) (b)

FIG. 16 – Exemple d'utilisation de la PCV de déplacement

3.4.2 La PCV de déplacement dans le reste du système

Le *Learner Manager* va tout d'abord définir un point de départ pour commencer le calcul du déplacement moyen et va le remettre à jour lorsque l'apprenant aura atteint un lieu important dans le scénario et ainsi réinitialiser le calcul. La PCV de déplacement va envoyer les informations sur les zones, la direction et la vitesse en permanence au *Learner Manager* (figure 17 page 27).

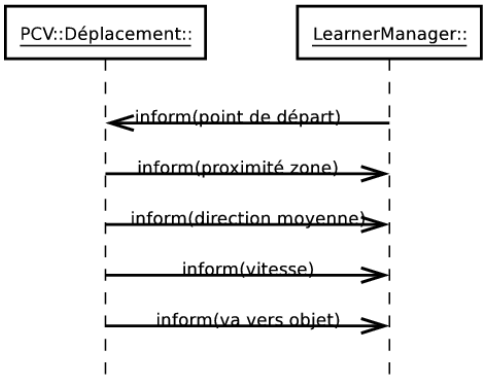


FIG. 17 – Fonctionnement de la PCV de déplacement (notation UML)

3.5 La PCV d'action

Dans le cas d'action sur le monde virtuel, [Fuchs et al. 03b] donne encore plusieurs sous catégories de PCV :

- manipuler un object en translation ;
- manipuler un object en rotation ;
- les deux actions associées ;
- déformer un objet ou assembler des objets.

Les actions sur les objets sont variées (rotation, translation, déformation, etc). Cependant, en fonction de l'avancement de l'apprenant, il est nécessaire de présenter l'action à effectuer de différentes façons. Mais le cadre de notre travail n'est pas la formation au geste technique ce qui simplifie le modèle. L'apprenant à une liste de d'action et doit faire le bon choix d'action.

Si l'apprenant n'est pas expérimenté, il est plus judicieux de lui faciliter la tâche en lui masquant les actions impossibles et les actions non requises. Dans le cas contraire, si l'apprenant est expérimenté, il est judicieux de lui montrer tous les types d'actions possibles. Lorsque l'utilisateur aura choisit son action, le *Pedagogical Agent* pourra alors vérifier s'il a acquis une connaissance.

La figure 18 (page 28) présente un scénario d'utilisation de la PCV d'action. Les *GeoEntities* sont des objets qui possèdent des connaissances sur les actions qu'ils peuvent effectuer. De plus, ils sont régis par des graphes d'état. Lorsqu'une action sur une de ces *GeoEntities*, elle va envoyer à la PCV d'action les diverses actions intrinsèques lui appartenant que l'état dans lequel elle se trouve.

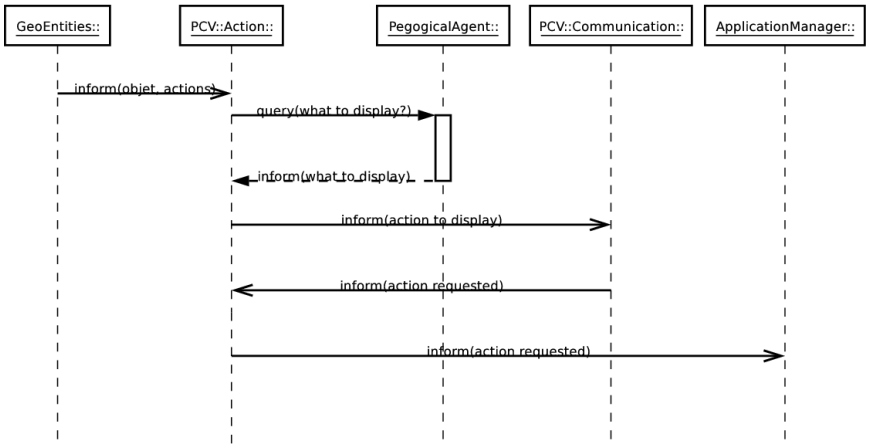


FIG. 18 – Fonctionnement de la PCV d'action (notation UML)

La PCV va ensuite demander au *Pedagogical Agent* quelles sont les actions à afficher en spécifiant pour chaque action si elle peut être activée ou pas. La PCV d'action informe ensuite la PCV de communication sur le fait d'afficher les différentes ces actions. Lorsque l'apprenant à choisit l'action à effectuer, la PCV communication relais cette action à la PCV d'action qui la relais elle-même à l'*Application Manager* qui va lors appliquer l'action d'en l'environnement virtuel.

3.6 La PCV de communication

Dans le cas de la communication, [Fuchs et al. 03b] donne plusieurs sous catégories de PCV :

- communiquer avec d'autres utilisateurs ;
- communiquer avec des personnages virtuels (avatars ou clones virtuels) ;
- communiquer avec l'application.

Dans le cas de la communication avec l'application, le sujet peut communiquer avec l'application pour modifier sa configuration, pour donner des ordres au système informatique, etc. Mais dans ce cas, l'utilisateur n'est plus en immersion pour utiliser l'application de réalité virtuelle. Nous nous situons dans le domaine classique de dialogue homme-machine des IHM (Interfaces Homme-Machine), en étant dans un environnement virtuel 3D (3 dimensions).

3.6.1 Reconnaissance vocale

Nous avons tout d'abord créé un module de reconnaissance vocale. En effet, nous souhaitons, en plus de détecter les actions, détecter aussi les intentions. Cependant, la détection d'intentions est un domaine très difficile. Donc nous avons choisi une solution simpliste mais qui nous permet de connaître de manière précise l'intention de l'utilisateur : la verbalisation d'intentions. L'utilisateur doit en effet verbaliser son intention avant d'effectuer une action. Ce module est composé de deux partie (la figure 19 page 30 montre un schéma décrivant le fonctionnement de ce module) :

- une interface avec le logiciel *Dragon Natural Speaking*[2], un logiciel de reconnaissance vocale qui envoie les phrases reconnues parmi une liste de phrases spécifiée ;
- un serveur TCP qui reçoit les phrases reconnues (dans une liste prééfinie) et les transmet à l'*Application Manager*.

En étendant les fonctionnalités de ce module, il est possible d'imaginer commander l'application sans passer par un périphérique tel qu'une souris ou un clavier, et communiquer avec les autres utilisateurs en utilisant un vocabulaire simplifié.

3.6.2 Le menu 3D

Le deuxième module permet de faciliter la sélection des diverses tâche à effectuer. Nous mettons en place une « substitution sensori-motrice » [Lourdeaux 01] sous la forme d'un menu en 3D (figure 21 page 31). En combinaison avec la PCV

[2] http://www.scansoft.fr/naturallyspeaking/

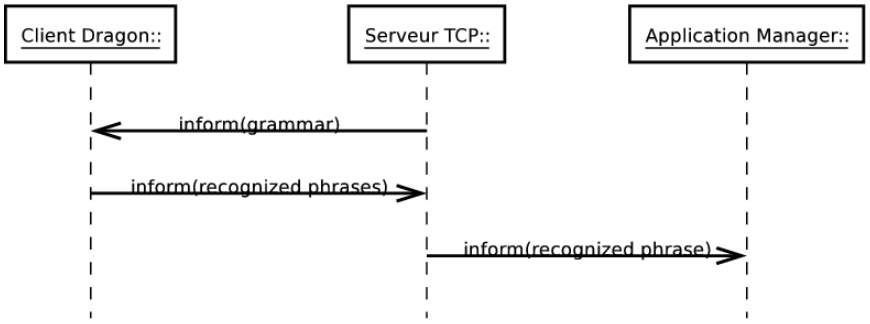


FIG. 19 – Fonctionnement du module de reconnaissance vocale (notation UML)

d'action, la PCV de communication va afficher les diverses actions que peut effectuer l'utilisateur. Une fois que l'utilisateur aura choisit l'action à effectuer, la PCV de communication va la retransmettre à la PCV d'action.

La figure 20 montre les diverses classes composant le menu 3D. Un menu est en fait un *container* d'élément 3D (Menu3DItem). Ceux-ci se disposent autour de l'objet sélectionné. Le *Menu3DInteractor* sert à détecter la sélection d'une des actions proposées.

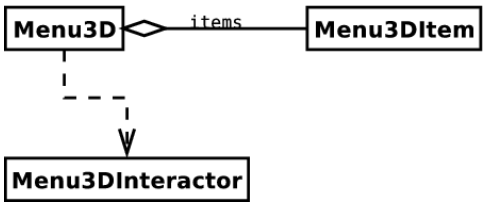


FIG. 20 – Les classes composant le menu 3D (notation UML)

La figure 22 page 32 présente l'utilisation de ce menu dans un VET.

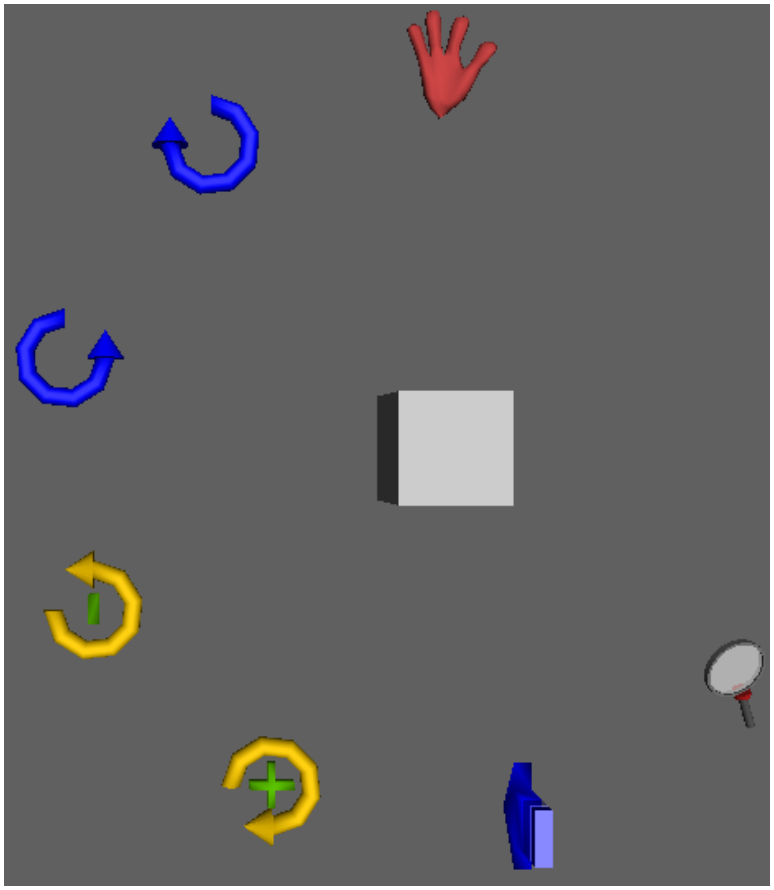


FIG. 21 – Exemple d'utilisation du menu 3D dans l'environnement

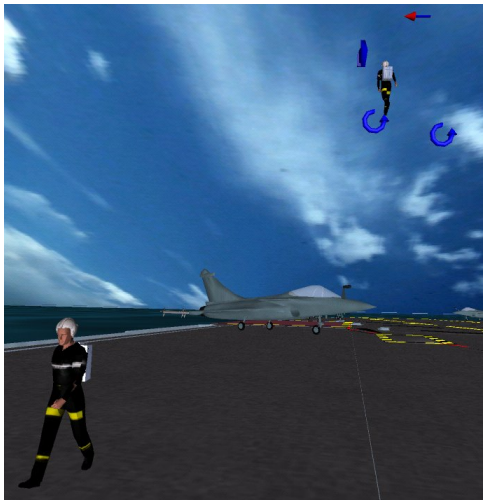


FIG. 22 – Le menu 3D dans un VET

3.7 Conclusion

Dans cette partie, nous avons présenté un modèle de d'interface en réalisant une implémentation des PCV de [Fuchs et al. 03b]. Nous avons détaillé chaque composant et avons montré l'application avec un exemple.

La PCV d'observation permet d'obtenir une liste des objets qui apparaissent dans le champ de vision de l'utilisateur ainsi que leurs distances et leurs orientations par rapport à l'utilisateur afin de les transmettres au *Learner Manager*.

La PCV de déplacement permet d'obtenir l'orientation moyenne, de savoir quel est l'objet vers lequel l'utilisateur se dirige globalement, de connaître la vitesse instantanée de l'utilisateur, de définir des zones et de savoir á quelles distances de ces zones l'utilisateur se trouve.

La PCV d'action permet d'intercepter les actions sur les objets de l'environnement et de les communiquer à l'*Application Manager* après avoir interroger les *Pedagogical Agent* sur la forme pour les présenter à l'apprenant.

La PCV de communication permet de « détecter l'intention » et de présenter l'information à l'apprenant.

4 Conclusion et perspectives

Dans ce rapport, nous nous sommes intéressés au modèle de l'interface dans les environnements virtuels de formation. Notre objectif était de donner une solution d'implémentation des PCV de [Fuchs et al. 03b].

Nous avons introduit le sujet par une étude bibliographique. Ensuite nous avons proposé notre modéle et avons détaillé chaque composant avec un exemple et une implémentation dans un VET que développe le CERV.

Le modèle que nous proposons permet de rendre des services au niveau pédagogique lorsqu'il est associé à un VET utilisant les ITS. Toutefois, il reste certains détails à mettre en œuvre. En effet, nous avons analysé la plupart des actions en entré indépendement, et il faudrait une meilleur gestion de ces entrées. Peut-être serait-il possible de regrouper tous les modules dans un agents interface, qu'il suffirait d'interroger, qui analyserait les besoins de la demande et qui nous renverrait les informations ad-hoc.

Il pourrait être également intéressant de se pencher sur le côter ergonomique comme il est spécifié dans la bibliographie. En effet, vu le temps impartit pour réaliser ce modèle, nous n'avons pas eu le temps de réflêchir à ce problème.

Références

[Anderson 88] Anderson, J. R. (1988). The expert module. In Polson, M. C. et Richardson, J. J., editors, *Foundations of Intelligent Tutoring Systems*, pages 21–53. Erlbaum, Hillsdale, NJ.

[Buche et al. 04] Buche, C., Querrec, R., Loor, P. D., et Chevaillier, P. (2004). Mascaret : A pedagogical multi-agent system for virtual environment for training. *Special issue 'Cyberworlds and education' of the international Journal of Distance Education Technologies JDET*, 2(4) :41–61.

[Burkhardt 03] Burkhardt, J.-M. (2003). Réalité virtuelle et ergonomie : quelques apports réciproques. *Le Travail Humain*, 66(1) :65–91.

[Burkhardt et al. 03a] Burkhardt, J.-M., Bardy, B., et Lourdeaux, D. (2003a). Les notions d'immersion, de réalisme et de présence dans la conception et l'évaluation d'environnements virtuels.

[Burkhardt et al. 03b] Burkhardt, J.-M., Lourdeaux, D., et d'Huart, D. M. (2003b). *Le traité de la réalité virtuelle*, volume Second, chapter La conception des environnements virtuels pour l'apprentissage, pages 207–296. Les Presses de l'École des Mines, second edition.

[Delestre 00] Delestre, N. (2000). *MÉTADYNE, un hypermédia adaptatif dynamique pour l'enseignement*. PhD thesis, Université de Rouen.

[Fuchs et al. 01] Fuchs, P., Arnaldi, B., et Tisseau, J. (2001). *Le traité de la réalité virtuelle*. Les Presses de l'École des Mines, first edition.

[Fuchs et al. 03a] Fuchs, P., Arnaldi, B., et Tisseau, J. (2003a). *Le traité de la réalité virtuelle*, volume First, chapter La réalité virtuelle et ses applications, pages 3–52. Les Presses de l'École des Mines, second edition.

[Fuchs et al. 03b] Fuchs, P. et Burkhardt, J.-M. (2003b). *Le traité de la réalité virtuelle*, volume First, chapter Approche théorique et pragmatique de la réalité virtuelle, pages 53–104. Les Presses de l'École des Mines, second edition.

[Gardie 01] Gardie, R. (2001). Communication non verbale entre avatars. Master's thesis, École Nationale des Ingénieurs de Brest.

[Gibson 79] Gibson, J. J. (1979). *The Ecological Approach to Visual perception*. Lawrence Erlbaum Associates, London.

[Halff 88] Halff, H. M. (1988). Curriculum and instruction in automated tutors. In Polson, M. C. et Richardson, J. J., editors, *Foundations of Intelligent Tutoring Systems*, pages 79–108. Erlbaum, Hillsdale, NJ.

[Hospers et al. 03] Hospers, M., Kroezen, E., Nijholt, A., Akker, R., et Heylen, D. (2003). Developing a generic agent-based intelligent tutoring system and applying it to nurse education. In Devedzic, V., Spector, J., Sampson, D., et Kinshuk, editors, *The 3rd IEEE International Conference on Advanced Learning Technologies (ICALT'03), Athens, Greece, July 9-12*, page 443, Los Alamitos. IEEE Computer Society.

[Johnson et al. 00] Johnson, W. L., Rickel, J., et Lester, J. C. (2000). Animated pedagogical agents : Face-to-face interaction in interactive learning environments. In *International Journal of Artificial Intelligence in Education*, volume 11, pages 47–78.

[Laird et al. 87] Laird, J. E., Newell, A., et Rosenbloom, P. S. (1987). Soar : An architecture for general intelligence. *Artificial Intelligence*, 33(1) :1–64.

[Lourdeaux 01] Lourdeaux, D. (2001). *Réalité virtuelle et formation : conception d'environnements virtuels pédagogiques*. PhD thesis, École des mines de Paris.

[Morineau et al. 01] Morineau, T., Chedmail, P., et Parenthoën, M. (2001). An affordance-based model to support simulation in virtual environment. In *Actes du congrès VRIC, Laval Virtual 2001*, pages 19–25.

[Newell 90] Newell, A. (1990). *Unified Theories of Cognition*. Harvard University Press, Cambridge.

[Querrec 02] Querrec, R. (2002). *Les systèmes multi-agents pour les environnements virtuels de formation. Application à la sécurité civile*. PhD thesis, Université de Bretagne Occidentale.

[Reignier et al. 98] Reignier, P., Harrouet, F., Morvan, S., et Tisseau, J. (1998). AReVi : A virtual reality multiagent platform. In *Virtual Worlds 98*, pages 229–240.

[Rickel et al. 98] Rickel, J. et Johnson, W. L. (1998). STEVE : A pedagogical agent for virtual reality. In Sycara, K. P. et Wooldridge, M., editors, *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pages 332–333, New York. ACM Press.

[VanLehn 88] VanLehn, K. (1988). Student modeling. In Polson, M. C. et Richardson, J. J., editors, *Foundations of Intelligent Tutoring Systems*, pages 55–78. Erlbaum, Hillsdale, NJ.

[Vigano et al. 03] Vigano, G., Mottura, S., Calabi, D., et Sacco, M. (2003). The virtual reality design tool : Case studies and interfacing open topics. In Coutellier, D., Fischer, X., Marquis, D., et Thouvenin, I., editors, *Proceedings of Virtual Concept 2003*, pages 364–371. École Supérieure des Technologies Industrielles Avancées.

[Wenger 87] Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann, San Francisco.

[Woolf 92] Woolf, B. P. (1992). Building knowledge based tutors. In Tomek, I., editor, *Computer Assisted Learning, Proceeding of the 4th International Conference, ICCAL '92*, Lecture Notes in Computer Science, pages 46–60. Springer.

Table des figures

Annexes

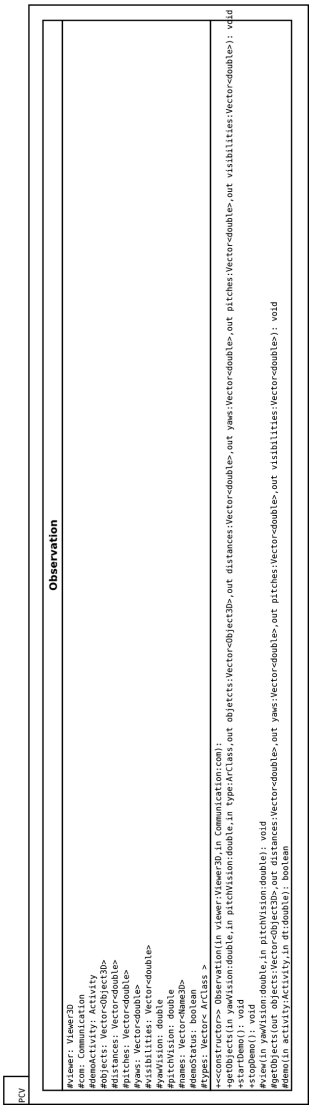


FIG. 23 – Diagramme de classe complet de la PCV d'observation (notation UML)

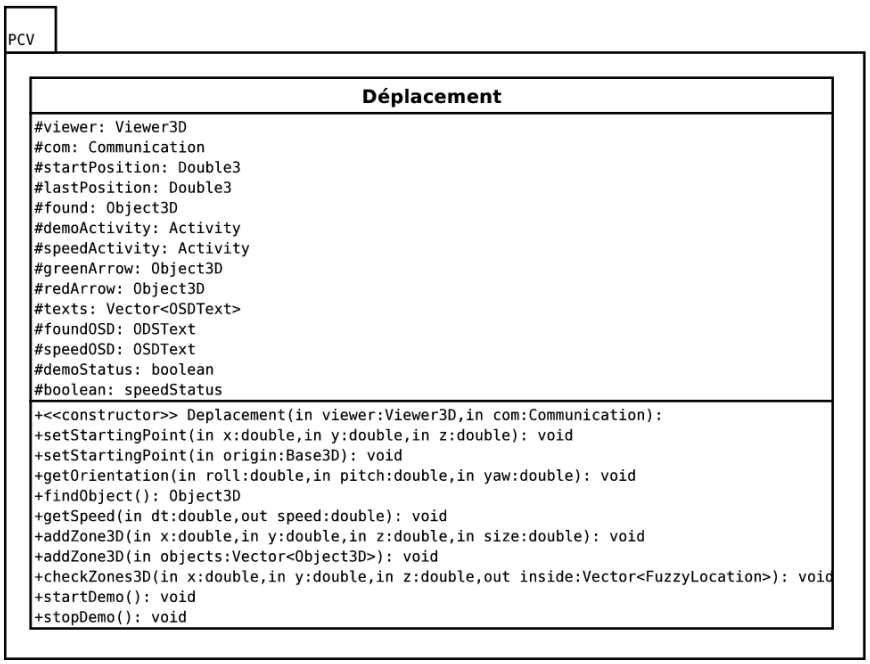


FIG. 24 – Diagramme de classe complet de la PCV de déplacement (notation UML)

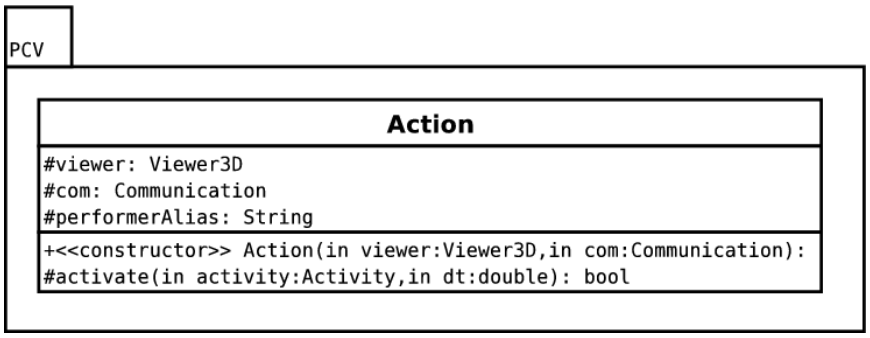


FIG. 25 – Diagramme de classe complet de la PCV d'action (notation UML)

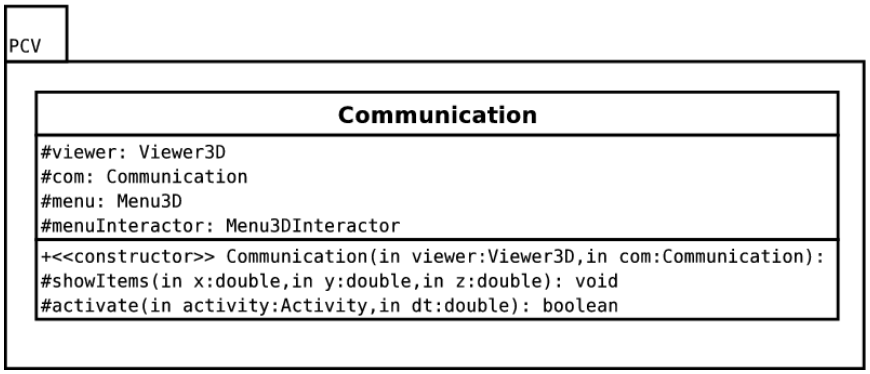


FIG. 26 – Diagramme de classe complet de la PCV de communication (notation UML)