

Une typologie d'actions humaines pour les Environnements Virtuels

Cyrille BAUDOIN, Pierre CHEVAILLIER, Ronan QUERREC

LISYC : Laboratoire des Systèmes Complexes

ENIB, Université Européenne de Bretagne

CERV, 25 rue Claude Chappe, 29280 Plouzané

19 mai 2008

MOTS-CLÉS : action, environnement virtuel d'apprentissage humain, réalité virtuelle, apprentissage, interface.

1 Introduction

Cet article se place dans le contexte du développement et de l'utilisation des Environnements Virtuels pour l'Apprentissage Humain (EVAH). Nous nous intéressons à l'utilisateur principal de ces systèmes de réalité virtuelle que nous qualifierons d'apprenant.

Nos recherches portent sur le suivi et l'analyse de l'activité de l'apprenant. Nous souhaitons pister, si possible automatiquement, ce que fait l'utilisateur dans le monde virtuel pour mieux faciliter l'apprentissage. L'objectif premier de notre analyse est d'évaluer l'activité d'un apprenant par rapport à des procédures ou des plans d'actions liés aux domaines d'apprentissage. Pour cela, Nous voulons pouvoir exprimer ces plans d'actions selon un référentiel explicite, puis fournir une méthode permettant de comparer les actions réalisées par l'apprenant à ce référentiel.

Pour pouvoir mener cette analyse, pendant ou à l'issue d'une séance d'apprentissage, il est nécessaire de réifier l'ensemble du modèle de l'interface, c'est-à-dire expliciter la chaîne d'interactions depuis le périphérique que l'utilisateur tient en main jusqu'à l'effet produit dans l'environnement. Une fois explicité, il est possible de surveiller l'activité du monde virtuel pour tracer les éléments observables de cette chaîne. De plus, il devient possible d'analyser cette trace pour détecter la réalisation d'actions humaines, matière première de notre analyse [Baudouin et al. 07a]. Pour atteindre ces objectifs, il est nécessaire de modéliser deux activités : celle du monde virtuel et celle de l'utilisateur humain.

Pour lier les deux, nous proposons de nous appuyer sur une typologie d'actions humaines permettant de modéliser l'activité observable des utilisateurs de façon explicite. Chaque action admet une formalisation évaluable. Il est ainsi possible de détecter la réalisation des actions dans les traces informatiques issues d'un environnement virtuel. Malheureusement, l'ensemble des actions humaines dans le monde réel est potentiellement infini et reste très grand dans nos mondes virtuels immersifs.

L'objectif de cet article est de proposer la formalisation d'un ensemble limité d'actions humaines, génériques et spécifiques, réalisables dans nos mondes virtuels.

Dans une première partie, nous présentons plus en détail le contexte d'application de cette étude, les besoins et les choix opérés pour réaliser notre modèle. Nous étudions ensuite des systèmes interactifs (2D et 3D) existants pour en extraire une liste d'actions. Puis nous décrivons une typologie d'actions humaines et leur formalisation. Nous montrons enfin l'utilisation de ce modèle à travers plusieurs exemples simples, puis leur possible application sur un cas d'étude existant.

2 Contexte de recherche et objectifs

Notre travail s'inscrit dans le développement et l'enrichissement de MASCARET¹ dont les contours ont été dessinés par [Chevaillier 06]. Il s'agit d'une plate-forme logicielle et d'un ensemble de méta-modèles, orientée EVAH, permettant à plusieurs utilisateurs, apprenants et formateurs, d'agir simultanément dans un environnement virtuel.

¹Multi-Agent System for Collaborative, Adaptive and Realistic Environment for Training, développé au LISYC (EA3883)

Nous proposons un modèle pour structurer les interactions humaines, afin de faciliter la mise en place d’une interface Humain–Machine plus souple, cohérente avec l’environnement virtuel et, dans le cas plus précis de la formation, en accord avec un domaine d’apprentissage et un scénario pédagogique.

Nous proposons dans cet article une typologie d’actions humaines, orientée vers deux objectifs. D’une part, nous souhaitons proposer un modèle d’interface explicite et exécutable, permettant à l’utilisateur d’interagir avec les objets du monde et avec d’autres agents. D’autre part, nous souhaitons définir une typologie exploitable pour l’expression et l’analyse de l’activité de l’utilisateur dans le monde, avec une visée principalement pédagogique.

Ces deux vues de l’interaction portent bien sur les mêmes actions humaines, mais elles ne sont pas toujours formellement liées. Elles doivent être mises en rapport de façon explicite pour permettre l’analyse. De plus, si l’on souhaite analyser les comportements en temps réel, ces liaisons doivent être réifiées dans nos environnements.

Notre proposition repose sur plusieurs postulats.

1. Le monde virtuel est un environnement discret. Le comportement des entités est déterminé par des machines à états explicites.
2. Le monde virtuel ne prend pas en compte la physique. Il est possible de détecter les collisions, mais nous ne calculons pas les forces imposées aux entités.
3. L’interface est continue, avec éventuellement un retour d’effort. Toutefois, nous ne nous intéressons pas aux gestes de l’utilisateur.
4. Les actions de l’apprenant devant son écran (écrire sur son cahier, poser une question orale à son voisin, etc.) ne sont pas médiées par l’environnement et par conséquent, on ne peut en rendre compte. Nous nous intéressons uniquement aux actions humaines détectables dans le monde virtuel.

2.1 Un monde structuré et informé : VEHA

Dans MASCARET, le monde virtuel est entièrement décrit en utilisant un méta-modèle qui est une extension de celui d’UML [Chevaillier et al. 08] : VEHA (*Virtual Environment supporting Human Activities*). Ce méta-modèle permet de modéliser les classes d’objets qui constituent l’environnement physique de l’apprenant. Il fournit une description de la composition

de cet environnement ainsi que des comportements des entités qui le composent. Il offre donc à la fois une vue statique et dynamique de l’environnement d’apprentissage. Ce monde est considéré comme discret : il est composé d’objets identifiables et leur évolution se fait par des changements d’états instantanés.

Le tableau 1 illustre la comparaison de VEHA avec UML : chacun des méta modèles (niveau M2) repose sur le MOF, Meta Object Facility (niveau M3) et peut être instancié en un modèle utilisateur (niveau M1) qui est lui même utilisé pour créer des applications spécifiques (M0).

M3	MOF (restriction d’UML)		
M2	modèle UML	modèle VEHA	
M1	<i>user model</i> UML	modèle EV_1	...
M0	<i>user object</i>	EV_{1a}	EV_{1b} ...

EV : environnement virtuel

TAB. 1 – Couches de modélisation (M_i) : positionnement de VEHA dans le référentiel MOF, parallèle avec UML.

L’utilisation de VEHA implique les deux premiers postulats : la non-continuité du monde et la non-prise en compte de la physique. Mais, en revanche, un modèle construit à partir de VEHA montre des propriétés intéressantes, en particulier pour concevoir des environnements à visée pédagogique :

1. Le monde est finement contrôlable : l’état du monde reste cohérent, même avec des comportements pseudo-aléatoires.
2. Le monde est informé et observable : il est possible de parler des objets, de leur comportement et de leurs propriétés.

2.2 Un modèle d’interface pour les environnements virtuels

La réalité virtuelle offre un grand nombre de possibilités pour réaliser l’interfaçage comportemental d’un utilisateur avec un monde virtuel [Fuchs 96]. La conception d’une interface de réalité virtuelle est soumise à trois types de contraintes : celles liées aux tâches et actions possibles dans le monde virtuel, celles liées aux utilisateurs de l’environnement et celles liées aux périphériques d’interactions disponibles. Nous nous intéressons dans cet article au premier point.

2.2.1 Interface et périphériques

Notre modèle d’interface permet une communication uniforme avec plusieurs périphériques de réalité virtuelle : gant de données, traqueur de tête, souris 3D, Wiimote². Chaque périphérique est décrit comme un ensemble d’actionneurs paramétrés. Chaque actionneur a une valeur : binaire pour les boutons, réelle pour les axes. Les paramètres peuvent être quantitatifs (bornes, stabilité, etc.) ou qualitatifs (prise en main, affordance, résistance, etc.). Ce modèle est implémenté sous la forme d’une bibliothèque assez proche de la plate-forme d’interface VRPN³ [Russell M. Taylor II et al. 01].

Du côté de l’environnement et des entités, les interactions possibles se ramènent soit à des activations d’état soit à des déclenchements d’opérations, qui peuvent être pilotées par des boutons, soit à la réalisation d’opérations paramétrées, qui nécessitent l’usage de valeurs non binaires, et donc d’axes.

2.2.2 L’agent Avatar

Cette description des périphériques est fournie parallèlement au modèle de l’environnement à un agent artificiel : l’avatar. Un avatar est associé à chaque utilisateur, pour qu’il puisse médier toutes ses interactions avec le monde. Cet agent est également le dépositaire du profil de l’utilisateur, qui définit ses préférences en terme d’interface.

Cet avatar a quatre responsabilités :

1. il représente l’utilisateur dans le monde, sa position, ses actions ;
2. il associe les opérations possibles à des actionneurs (périphériques) disponibles ;
3. il contraint les actions sur le monde virtuel (contraintes du contexte et du scénario pédagogique) ;
4. il trace l’activité de l’utilisateur, en renseignant sur les opérations réalisées et les périphériques associés.

2.3 Un modèle de l’activité humaine pour la RV : HAVE

Le méta-modèle HAVE (*Human Activities in a Virtual Environment*) de MASCARET, a pour objectif de formaliser les notions d’actions et de procédures comportementales dans la simulation et de décrire

²Manette de jeu de la console Wii (Nintendo), pourvue d’un dispositif de pointage par infrarouge et de trois accéléromètres

³Virtual Reality Peripheral Network

les activités d’un collectif d’agents [Chevaillier 06]. Comme dans le cas de VEHA, l’utilisation de ce méta modèle respecte les différents niveaux de modélisation (cf. tableau 1).

Toute entité active dans l’environnement est considérée comme un agent, c’est-à-dire une entité capable de percevoir son environnement local, de choisir une action adaptée, éventuellement guidée par un but explicite (principe de rationalité) et d’agir en conséquence par le biais d’interactions avec l’environnement ou d’autres agents. Les agents sont donc de diverses natures. Il s’agit tout d’abord des entités représentées dans le monde virtuel et exhibant un comportement autonome (non commandé par une autre entité). Les utilisateurs sont eux-mêmes considérés comme des agents ; ils exercent une activité orientée vers un but et interagissent entre eux, via l’environnement virtuel.

Notre approche est centrée sur les tâches effectives et ne contient pas de modèle de l’utilisateur. Nous ne faisons pas d’hypothèse forte sur les processus cognitifs des utilisateurs. Nous cherchons à détecter et qualifier ses actions, pas à déduire ses intentions. Des modèles cognitifs existants tels SOAR [Newell 90], ACT-R [Anderson 93] ou GOMS [John et al. 96], tous basés sur le modèle du processeur humain, proposé par Card, Newell et Moran, [Card et al. 83], peuvent être utilisés pour l’analyse des comportements cognitifs des utilisateurs.

De même, chaque agent artificiel se doit de respecter les contraintes d’interactions du monde, mais l’utilisation d’un modèle d’agent particulier n’est pas imposée.

Le modèle des activités est supporté par une modélisation de la structure organisationnelle de ce collectif. Le modèle organisationnel permet de définir le rôle des agents au sein du collectif. Cette notion d’organisation liée aux rôles est l’une des bases de MASCARET et a été proposée par [Querrec 02] dans sa thèse. Elle est proche de celle mise en place dans le modèle MOISE [Hannoun et al. 99]. Les activités sont décrites comme un ensemble d’actions à réaliser par une structure organisationnelle afin d’atteindre un objectif. Par exemple, une séance pédagogique est structurée en exercices dont la réalisation est décrite sous forme d’un modèle d’activités qui précise les actions de chacun des agents en fonction de leur rôle. Ceci permet de spécifier la tâche que doit réaliser l’apprenant au cours de chaque exercice. Les activités sont décrites sous forme d’enchaînements possibles d’actions ; on parle alors de description procédurale. Une action est définie par des conditions de faisabilité

et des effets attendus ; ces conditions sont formalisées sous forme d’expressions logiques sur les entités VEHA qui composent l’environnement des agents.

Du côté de l’environnement, la réalisation d’une action se traduit par une séquence d’opérations à réaliser, définies dans les entités VEHA. Ces dernières correspondent donc à l’atome de la description d’une activité, du point de vue de l’exécution. Par exemple, l’action placer sur l’entité NappeBlanche aura uniquement un effet sur sa position, alors que l’action Déplier sur cette entité aura un effet plus complexe et spécifique au domaine, qui modifiera l’état de l’entité et son apparence, par l’activation d’opérations et le déclenchement d’animations.

Du point de vue de l’analyse de l’activité humaine, l’atome de l’activité est l’action. Une suite de trois opérations peut avoir plusieurs sens, et peut traduire la réalisation d’actions très différentes (voir même opposées). Il n’est donc généralement pas possible de se baser uniquement sur l’exécution de ces opérations pour déterminer quelles actions ont été réalisées par un apprenant.

2.4 Une typologie d’actions pour décrire l’activité

Pour pouvoir exprimer l’activité des utilisateurs dans nos diagrammes d’activité HAVE, nous utilisons des verbes d’action génériques et des verbes tirés du domaine d’apprentissage. Un verbe générique tel que *se déplacer* a toujours le même sens pour tous nos environnements virtuels : modifier l’attribut position de son avatar. Bien que la réalisation de ces actions puisse varier en terme d’interface (déplacement en position ou en vitesse) et de résultat (liberté de mouvement dans le monde, chemin prédéfini), il est possible assez simplement de définir cette action en dehors du domaine d’application. Pour des actions spécifiques tel que *visser*, *plier* ou *enregistrer sous*, il est plus difficile de donner une définition valide pour plusieurs domaines. Il existe donc une ontologie d’actions pour chaque domaine d’application. Notre typologie doit donc se baser sur le langage naturel et permettre la description d’une activité humaine attendue avec une sémantique non ambiguë.

2.5 Une typologie d’actions pour analyser l’activité

Par ailleurs, nous souhaitons analyser de façon automatique et semi-automatique l’activité humaine observable dans un environnement virtuel. Pour pou-

voir détecter et analyser ce que fait un utilisateur, il est nécessaire que chaque action de notre typologie soit exprimé dans un langage informatique interprétable dont la sémantique est explicite.

3 Modèles et environnements existants

Avant d’établir notre liste d’actions génériques, nous avons cherché l’existence d’une telle typologie dans la littérature scientifique. S’il existe des typologies de verbes d’action en langage naturel ou des langages informatiques permettant de décrire une activité humaine attendue ou observable, nous n’avons pas trouvé de système qui fasse le lien entre nos deux besoins. Nous nous sommes donc tournés vers des systèmes informatiques existants, où l’activité est définie spécifiquement. Nous nous sommes plus particulièrement intéressés aux logiciels et environnements où l’utilisateur manipule l’interface par des verbes d’action explicites, ces verbes étant liés à une sémantique informatique. Ensuite, nous avons cherché à synthétiser les points communs de ces environnements existants comme base de notre typologie.

3.1 Définition du travail humain et méthode des temps prédéterminés

Les méthodes des temps prédéterminés, utilisées dans l’industrie pour servir de base au calcul du temps de travail peuvent servir de base pour décrire l’activité humaine. Elles définissent des actions basiques, liées à un temps nominal pour les accomplir. Elles permettent donc de découper une activité en de nombreuses actions minutées précisément. Ces méthodes existent depuis les débuts du Taylorisme et ont peu évolué dans le temps : révision des temps, proposition de nouvelles actions.

Nous nous sommes intéressés à l’une de ces méthodes, MTM⁴ [Maynard et al. 48], qui se dérive en plusieurs normes, proposant plusieurs niveaux de description : MTM-1, MTM-2, MTM-3. Le niveau MTM-1, qui constitue la norme historique est celui qui correspond le plus à la granularité de nos actions humaines atomiques. La granularité de MTM-1 est fine et pour analyser une minute de temps de travail sur une chaîne de production, il faut en moyenne une journée de travail à des experts de la

⁴Method Time Measurement

méthode. Le niveau supérieur, MTM2, conçu pour réduire ce temps d’analyse par dix, décrit des actions d’un niveau plus élevé, que nous considérons comme des activités. MTM-3 se veut plus efficace car plus spécifique au domaine. Cette norme ne décrit pas l’intégralité du travail à la chaîne, mais permet de définir, à partir de MTM-2, un ensemble d’actions spécifiques au métier. [Longo et al. 06] ont utilisé la norme MTM pour décrire temporellement l’activité de travailleurs simulée sur une chaîne de production. Une autre norme, se voulant plus simple que MTM, a été produite par le même cabinet d’experts, MOST⁵ [Zandin 03]. Elle a elle-même été dérivée et découpée en plusieurs autres normes : BasicMOST, MiniMOST, MaxiMOST et AdminMOST.

MTM1 se veut plus exhaustive que générique et décrit 350 actions de base, ce qui permet normalement de définir toutes les activités de travail possibles. A chaque action, la norme MTM associe un code et un temps standard. Le tableau 2 montre quelques exemples d’actions définies dans MTM-1. La granularité des actions est assez fine et décrit également les mouvements du corps de l’opérateur.

R	Atteindre
L	Mouvement de jambe
M	Mouvoir
F	Mouvement de pied
T	Tourner
SS	Pas de côté
AP	Appliquer Pression
TB	Rotation du corps
G	Saisir
W	Marcher
P	Placer
B	Pencher
RL	Relâcher
S	Relever
D	Désengager
ET	Déplacement yeux
EF	Focalisation yeux
KOK	S’agenouiller (1 genou)
KBK	S’agenouiller (2 genoux)

TAB. 2 – Quelques actions de la norme MTM-1

Pour la plupart des actions, MTM propose des déclinaisons, précisant l’action ou indiquant le niveau de difficulté rencontrée par l’exécutant. Par exemple, l’action Atteindre (symbole R - Reach) accepte ces

⁵Maynard Operation Sequence Technique

cinq déclinaisons :

1. RA : Objet toujours au même endroit, ou dans l’autre main ;
2. RB : L’emplacement de l’objet peut varier sensiblement d’un cycle à l’autre ;
3. RC : Atteindre un objet mêlé à d’autres ;
4. RD : Atteindre un objet petit ou à saisir avec précision ou précaution ;
5. RE : Vers une position indéfinie.

Cette typologie ne possède pas de formalisation, mais est une bonne indicatrice du nombre d’actions à définir si on veut pouvoir tout faire dans un monde virtuel. En ne traitant pas toutes les déclinaisons et en regroupant les actions par ressemblance, on peut réduire le nombre d’actions à moins de primitives. Nous pourrions aussi nous servir de MTM-1 pour évaluer notre modèle, en exprimant l’ensemble des actions définies par cette méthode grâce à notre formalisation, et en se basant sur nos actions méta.

3.2 Environnements existants et listes d’actions

Nous présentons ici un panel de systèmes interactifs utilisant des actions explicites, depuis les jeux des années 70 en mode texte jusqu’au simulateur de réalité virtuelle pour l’apprentissage. Pour chaque système, nous donnons une brève description du système (type d’environnement, contexte, objectif), une description de son interface (comment agit-on sur le monde?). Le tableau 7, donné en annexe, regroupe les actions principales des systèmes cités précédemment, classées par famille d’action.

3.2.1 Jeux vidéo textuels

Les systèmes informatiques les plus anciens utilisant une interface avec des actions explicites sont les jeux d’exploration de donjons (*Dungeon crawler* ou *Rogue-Like* en anglais) tel que *Rogue* ou les MUD (*Multi Users Dungeons*). Très simples visuellement, ils consistent à explorer des pièces reliées les unes aux autres, pour combattre des monstres et trouver des trésors. Affichés en mode texte, ils disposent d’une interface très simple, avec généralement peu d’actions possibles, généralement associées à une touche du clavier : Avancer, reculer, aller à droite, aller à gauche, ouvrir une porte, fouiller une pièce, combattre un monstre, fuir, etc.

Des actions spécifiques peuvent apparaître comme ouvrir un coffre, fracturer un cadenas, etc.

Certains de ces jeux acceptent une entrée clavier pour taper les verbes dans leur intégralité, permettant ainsi plus de souplesse dans l’interface, tout en ralentissant fortement le jeu.

3.2.2 Jeux vidéo pointer-cliquer

A la mode dans les années 1980, ce type de jeu met en scène un personnage contrôlable à la souris dans un espace en deux dimensions (parfois en 3D isométrique)⁶. Ce concept est particulièrement illustré par les jeux de Lucas Arts, basés sur le moteur SCUMM⁷, tels que *Maniac Mansion*, *Sam and Max*, *Indiana Jones* ou *Day Of The Tentacle*.



FIG. 1 – Capture d’écran du jeu Day of the Tentacle

L’interface est constituée d’une liste de verbes d’action, affichée en permanence à l’écran (cf. 1. Pour interagir avec un objet, il faut sélectionner l’action désirée sur l’interface, puis cliquer sur l’objet. Pour se déplacer, il suffit de cliquer sur le sol pour que le personnage s’y rende automatiquement grâce à un algorithme de *Path-finding* (l’action *aller à* étant sélectionnée par défaut). L’action *prendre*, sur un objet pointé qui peut être pris, ajoute cet objet dans un sac virtuel, non représenté dans le jeu, mais dont le contenu s’affiche à droite des verbes d’action. Certains objets sont des outils utiles pour résoudre certaines énigmes et il est possible d’assembler des objets deux à deux ensemble pour créer de nouveaux objets. L’utilisation d’un outil ou l’assemblage de deux objets se fait grâce à l’action générique *utiliser* un objet *sur* un autre objet.

3.2.3 Jeux de rôles

Les jeux de rôle classiques sont des jeux d’aventures et de récits, classiquement joués à plusieurs. Les règles définissent un ensemble de compétences

et d’actions possibles en langage naturel avec, en général, un moteur d’évaluation mathématique pour déterminer la réussite ou non d’une action : formules, tables, abaquages et dés pour insérer de l’aléatoire. Le jeu étant basé sur le récit, et le modèle d’évaluation étant souple, il est possible de dériver et d’ajouter des nouvelles actions par analogie.

Leurs transpositions dans les jeux vidéo, tel que *Fallout* ou *Knight of the Old Republic*, reprennent ces principes, mais cloisonnent les actions et les compétences à une liste fixe et prédéfinie. L’interface est généralement assez complexe et hérite des deux catégories de jeux citées précédemment. Il est possible de réaliser un certain nombre d’actions de base dans le monde (se déplacer, ouvrir les portes), plus des actions spécifiques sur certains objets. La valeur ajoutée ici est le profil du personnage qui octroie une liste particulière de compétences, et donc d’actions : modes de déplacements particuliers (courir, nager, se cacher, etc.), actions spéciales sur les objets (briser, cacher), utilisation d’outils (crocheter, réparer, etc.).

3.2.4 GVT

Generic Virtual Training⁸ est une plate-forme générique et modulaire pour le développement de sessions de formations utilisant la réalité virtuelle [Gerbaud et al. 08]. Développée en partenariat par le LISYC, l’IRISA et Nexter Systems⁹, cette plate-forme permet, entre autre, la formation à la maintenance mécanique de matériel industriel. L’apprenant évolue dans un environnement 3D où il déclenche des actions via un menu contextuel. Le système pédagogique de GVT supervise ces actions et les autorise ou pas selon leur validité et la stratégie pédagogique choisie. Le domaine d’application militaire impose des procédures prédéterminées avec peu de variantes possibles. De nombreuses opérations de maintenance nécessitent des outils précis (clé dynamométrique par exemple) et GVT utilise des actions génériques, et des actions liées aux objets.

3.2.5 Synthèse

L’étude de ces différents systèmes montre bien d’un côté l’existence d’actions communes, tels que les déplacements dans l’environnement ou les actes de communication, et de l’autre, le nombre important d’actions spécifiques à chaque domaine d’application. Les actions communes, bien que liées à

⁶Point and Click en anglais

⁷Script Creation Utility for Maniac Mansion

⁸anciennement GIAT Virtual Training.

⁹anciennement GIAT Industries

une interface différente (touche du clavier, souris, manette de jeu...), possèdent la même sémantique et peuvent donc être regroupées dans notre typologie comme des actions génériques, possédant la même définition quelque soit le domaine. C’est le cas d’actions telles que *prendre objet*, *parler* ou *aller à*. Les actions relatives aux différents systèmes doivent, au contraire, avoir une définition spécifique pour chaque domaine. L’action *appeler formateur* (GVT) n’a pas de sens dans un jeu d’aventure, de la même façon que l’action *secret door* (chercher une porte secrète) (Rogue) n’a pas lieu d’être dans un environnement de formation. Un seul verbe d’action, tel que *tirer*, a également un sens différent en fonction du domaine : tirer une manette, ou attirer un objet vers soi. Certaines actions sont proches bien que différentes mais peuvent être généralisées en partie dans notre typologie, sous une forme commune. Par exemple, *lire*, *regarder* ou *vérifier*, définissent des actions différentes, mais sont toutes des prises d’information, qui impliquent l’affichage de données particulières à l’écran.

Cela nous encourage à définir deux niveaux de description : d’une part un type d’action, générique, apportant une description partielle de l’action, puis l’action elle-même, dont la définition est spécifique au domaine.

4 Proposition : une typologie d’actions pour HAVE

Nous cherchons à décrire formellement un ensemble d’actions non spécifiques au domaine, et nous proposons une méthode d’écriture pour les actions spécifiques. Cette typologie a pour but d’enrichir l’actuel modèle HAVE de MASCARET, en proposant un ensemble d’actions méta supportées par le méta-modèle VEHA. Il est donc possible de définir chaque action méta en apportant une formalisation des conditions nécessaires à son exécution et ses effets sur le monde, en se basant sur les propriétés génériques de nos entités de réalité virtuelle : positions, orientations, machine à états, propriétés.

Nous classons les actions humaines et les activités en quatre familles, proposées par [Fuchs et al. 06] :

- la manipulation d’objets, qui regroupe toutes les interactions avec les entités du monde,
- la navigation, qui regroupe les actions visant à se déplacer dans le monde virtuel,
- l’observation, qui correspond aux changements de point de vue et aux prises d’information dans

le monde,

- la communication, qui regroupe les dialogues verbaux et non verbaux entre les agents humains et artificiels.

Lors de la création d’un modèle en utilisant VEHA et HAVE, nous pouvons instancier ces actions méta en actions spécifiques au domaine d’application (cf. figure 2). Ces actions spécifiques portent donc sur des entités du domaine et leur formalisation est dérivée du niveau M2. Il est également possible de dériver les actions pour rajouter des actions avec des contraintes spécifiques au domaine d’application. Ensuite, une action peut être instanciée dans l’environnement (niveau M0) pour des exécutions attendues de cette action, que l’on pourra comparer aux exécutions observées, disponibles dans les traces.

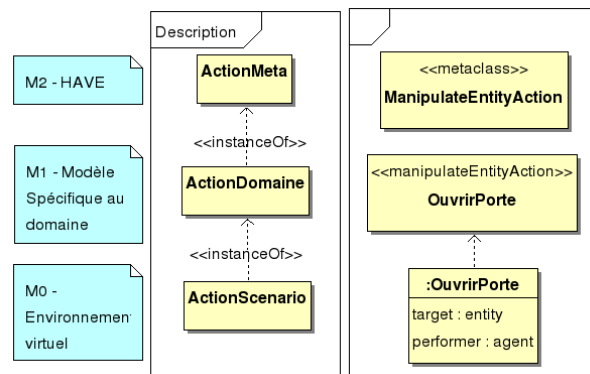


FIG. 2 – Vue des actions pour les trois niveaux de modélisation

En apportant cette formalisation, il devient possible de raisonner sur ces actions, pour pouvoir analyser l’activité de l’utilisateur.

4.1 Un langage pour la formalisation des actions

Nous cherchons donc un langage pour exprimer formellement l’exécution d’une de ces actions humaines dans notre monde. Ce langage doit permettre d’exprimer les contraintes nécessaires à l’exécution d’une action, et son effet sur le monde. Notre méta-modèle étant proche de celui d’UML, nous nous tournons naturellement vers le langage d’expression de contraintes qui lui est associé : OCL, Object Constraint Language [OMG 06].

La définition formelle de chaque action précise le contexte de l’action (contexte :) et les préconditions (pre :) et postconditions (post :). Ces conditions

portent sur le contexte d’exécution des actions, c’est-à-dire sur les propriétés des entités manipulées et de l’avatar. Lorsque cela est nécessaire, des méthodes des méta-classes du méta-modèle HAVE et VEHA (cf. figure 3) peuvent être utilisées pour évaluer les conditions (query OCL) et des méthodes peuvent être définies localement (*def* :).

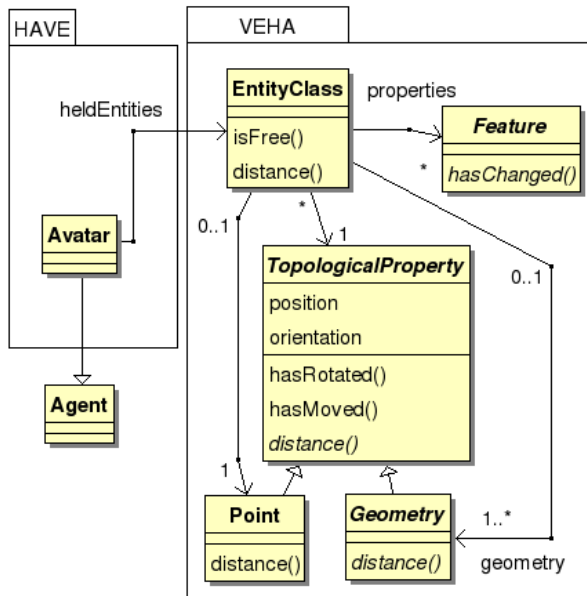


FIG. 3 – Diagramme de classe UML figurant les méta classes Avatar et EntityClass

4.2 Manipulation d’entité

Dans les environnements virtuels réalisés grâce à MASCARET, un utilisateur peut utiliser plusieurs périphériques en même temps : un dans chaque main, mais aussi des périphériques de capture (position de la tête, des bras, du regard). Il est donc à même de réaliser en parallèle des actions ou des activités de familles différentes, par exemple se déplacer (navigation) et retourner une entité (manipulation) ou déplacer une entité (manipulation) et changer son point de vue (observation).

L’avatar de l’utilisateur, tel que défini par le méta-modèle HAVE, n’est pas forcément un humanoïde animé. Il possède néanmoins la caractéristique minimale suivante : pouvoir prendre et porter une ou deux entités en même temps et pouvoir exécuter parallèlement une action de manipulation sur chaque entité.

D’un autre côté, chaque entité ne peut être tenue que par un et un seul utilisateur. Si l’on souhaite

permettre la prise à plusieurs avatars, il convient de modéliser l’objet avec plusieurs entités, en définissant explicitement les points ou les zones de prise.

4.3 Navigation

Le niveau de complexité de l’entité avatar dépend des choix de conception de l’environnement final. On pourra se contenter de déplacer la caméra dans le monde, animer un personnage qui ne pourra que se déplacer en marchant, ou permettre le contrôle fin d’un pantin articulé. L’interface pour une même action sera alors très différente : par exemple l’action *s’asseoir* pourra être liée à une entité chaise, et consister en un simple clic sur la surface de la chaise lorsque l’avatar est à proximité, ou utiliser une modification de la posture de l’avatar pour plier les jambes et lui donner une position assise, avec ou sans chaise. Quelque soit l’interface, le point commun de la navigation est le changement de position et d’orientation de l’avatar, qu’il soit représenté ou non dans le monde.

4.4 Communication

Les opérations VEHA pour communiquer verbalement entre agent humain et artificiel sont l’envoi et la réception de messages verbaux (textuels ou vocaux). Dans cet article, nous ne rentrons pas dans l’étude complexe des activités de communication. Nous nous appuyons sur la théorie des actes de langage pour la communication verbale [Searle 69].

A cela s’ajoute la communication non-verbale : posture, regard, gestes, expressions du visage, etc. Selon [Lewis 00], on peut distinguer neuf canaux de communication : les expressions du visage (sourires, signes de tête), les gestes (en particulier des mains et des bras), les mouvements du corps, la posture, la direction du regard (en particulier regarder dans les yeux), les contacts physiques (poignée de main), l’occupation de l’espace (proximité, distance, positions), l’apparence (incluant les habits portés), les vocalisations non verbales (siffler, crier, pleurer, rire).

Les actions de communication non verbale explicites sont spécifiques à l’interface et au domaine cible, et nous ne proposons pas d’action méta pour ce vecteur de communication.

4.5 Observation

Les actions d’observation regroupent toutes les perceptions actives, c’est-à-dire les actions mises en œuvre par l’apprenant pour percevoir et s’informer

de l’état du monde. Ces actions ne sont généralement que partiellement médiées par l’environnement. En effet, il est possible de savoir quand un objet ou une situation est visible par l’apprenant, mais on ne peut que supposer que l’apprenant effectue bien une prise d’information. L’action de niveau méta de cette famille est l’action *changer point de vue*. L’action spécifique la plus évidente est *regarder*, qui correspond à un changement du point de vue dans l’environnement, mais qui ne porte pas d’information sur la direction réelle du regard de l’utilisateur. D’autres actions spécifiques comme *lire l’énoncé* ou *afficher l’information* pourront être définies dans l’interface, avec une définition propre en fonction du domaine.

Pour une analyse plus précise des prises d’information, deux choix peuvent être faits au niveau de l’interface : expliciter les prises d’information ou tenter de les inférer. La première solution est utilisée dans GVT : lorsque l’apprenant doit lire une valeur sur un cadran ou regarder l’état d’un objet, on l’oblige à déclencher l’action *vérifier* sur cet objet. La seconde solution peut consister à déduire des autres actions (manipulation, navigation) et des temps passés. Par exemple, si l’apprenant a bien changé son point de vue pour faire apparaître la situation intéressante et qu’il n’agit pas pendant quelques secondes, on considérera qu’il a bien regardé la situation. Ces choix sont dépendants de l’analyse de l’activité que l’on souhaite mettre en place.

5 Typologie des actions et définitions formelles

Nous présentons maintenant notre typologie d’actions. Elle n’est pas exhaustive et ne constitue pas un modèle de l’activité en elle-même. Elle doit être suffisante pour exprimer les activités humaines médiées par nos interfaces dans nos mondes virtuels. Les activités prises en compte sont la navigation dans le monde, la manipulation d’objets virtuels, les actions explicites d’observations et la communication dans l’environnement (avec des agents, humains ou artificiels).

Pour chaque action, nous donnons ces quatre informations :

1. Sa dénomination, généralement un verbe d’action, parfois une expression ;
2. Sa définition en langue naturelle, tirée du TLFi¹⁰ [Bernard et al. 02] ;

¹⁰Trésors de la Langue Française Informatisé, dictionnaire

3. Notre définition, généralement équivalente à celle du TLFi, ou plus restrictive ;

4. Notre définition formelle, exprimée en OCL.

Pour les actions spécifiques, nous précisons le stéréotype (l’action méta correspondante), les éventuelles relations qu’elle partage avec les autres actions (*dérive de* ou *contraire de*, par exemple) et un ou plusieurs exemple(s) d’instruction portant sur cette action (sauf actions méta).

5.1 Les actions méta

5.1.1 Actions de manipulation

Prendre et lâcher sont les deux actions qui bornent toutes les manipulations d’entités.

Action : Prendre (entité)

Définition : Saisir quelque chose (ou quelqu’un), généralement avec une partie du corps ou avec un instrument, à des fins diverses.

Sémantique : Saisir l’objet sélectionné, dans le but de le manipuler ou de le modifier. L’objet est lié à l’avatar qui l’a saisi.

```
context: Avatar::PickUpEntity(in
ent:Entity)
pre: self~Selectionner(ent) and
ent.isFree()
post: not ent.oclInState(Free) and
~createLinkAction(self.heldEntity,
entity.holder) and
self.heldEntities→includes(ent)
```

Une entité est sélectionnée lorsque l’utilisateur désigne l’objet avec l’un des curseurs de sélection (2D ou 3D). C’est l’interface finale qui spécifie comment l’utilisateur peut sélectionner les entités, mais cette primitive est un pré-requis à tous nos environnements. L’action prendre ajoute l’entité sélectionnée dans l’ensemble des entités portées par l’avatar. Elle modifie aussi les valeurs de l’association entre entité et avatar ($\text{heldEntities} \leftrightarrow \text{holder}$).

Action : Lâcher (entité)

Définition : Rendre moins serré, tenir moins tendu

Sémantique : Détacher un objet pris par la main.

en ligne, <http://atilf.atilf.fr>, ATILF, CNRS, Nancy 2

```
context: Avatar::DropEntity(in ent:Entity)
pre: self.heldEntities→includes(ent)
post: ent.isFree() and
~destroyLinkAction(self.heldEntity,
entity.holder) and not
self.heldEntities→includes(ent)
```

Manipulation générique :

Action : Manipuler (entité)

Définition : Toucher, tenir, transporter avec les mains. Faire fonctionner ; utiliser avec les mains.

Sémantique : Manipuler une entité pour modifier l’une de ses propriétés, faire changer son état ou déclencher une opération.

```
context: Avatar::ManipulateEntity(in
ent:Entity)
pre: self~PickUpEntity(ent)
post: self~DropEntity(ent)
ent.features→exists(feat | hasChanged())
```

Manipulations géométriques :

Action : Bouger (entité)

Définition : Déplacer.

Sémantique : Modifier la position d’une entité.

```
context: Avatar::MoveEntity(in ent:Entity)
post: self~ManipulateEntity(ent) and
ent.geometry.hasMoved()
```

Action : Orienter (entité)

Définition : Disposer, tourner dans telle ou telle direction favorable.

Sémantique : Modifier l’orientation d’un objet.

```
context: Avatar::OrientEntity(in
ent:Entity)
post: self~ManipulateEntity(ent) and
ent.geometry.hasRotated()
```

Changement d’état :

Action : Faire changer l’état d’une (entité)

Définition : Changer : Faire autre, faire que quelque chose ou quelqu’un soit autre. État : Manière d’être (soit stable, soit sujette à des variations) d’une personne ou d’une chose.

Sémantique : Changer l’état interne d’une entité, c’est-à-dire tirer l’une des transitions de sa machine à état.

```
context: Avatar::ChangeEntityState(in
ent:Entity)
pre: ent.stateMachine.transitions
→exists(isActive())
post: self.stateMachines→exists(state |
hasChanged())
```

L’entité possède une machine à états VEHA avec au moins une transition tirable.

Les modifications de propriétés (*Feature*) non géométriques et non liées aux états sont des actions du type *Manipuler*.

5.1.2 Actions de navigation

Action : Bouger

Définition : Faire un mouvement (le plus souvent minime), remuer

Sémantique : Avoir parcouru une certaine distance.

```
context: Avatar::MoveSelf()
post: self.geometry.hasMoved()
```

Action : Se Tourner

Définition : Se mettre dans un autre sens, en sens inverse, dans une direction donnée.

Sémantique : Tourne le corps de l’avatar autour de son axe vertical (a priori identique à l’axe vertical du repère global).

```
context: Avatar::RotateSelf()
post: not self.geometry.orientation.z =
self.geometry.orientation.z@pre or
self.geometry.hasRotated()
```

Cette rotation du corps est distincte d’un changement du point de vue (lié à l’orientation de la tête et du regard). Néanmoins, il est possible de lier les deux actions dans l’interface (la tête ne tourne pas par rapport au tronc).

5.1.3 Actions d’observation

Action : Changer son point de vue

Définition : Changer : Faire autre, faire que quelque chose ou quelqu’un soit autre. Point de vue : lieu d’où l’on regarde.

Sémantique : Changer le point de vue sur la scène. Modifier l’orientation de la caméra qui filme la scène

3D.

```
context: Avatar::ChangeViewPoint()  
post: not (self.viewPoint =  
self.viewPoint@pre)
```

L’attribut *viewpoint* correspond à l’orientation de la caméra qui filme la scène, assimilée à la direction du regard de l’avatar, et donc à ce que voit l’utilisateur.

5.1.4 Actions de communication

Action : Informer (agent) de (connaissance)

Définition : Faire savoir quelque chose à quelqu’un, porter quelque chose à la connaissance de quelqu’un.

Sémantique : Donner une connaissance à un autre agent.

```
context: Avatar::Inform(in klg:Knowledge,  
in agt:Agent)  
pre: self→believe(klg)  
post: agent→believe(klg)
```

Action : Demander (connaissance) à (agent)

Définition : Faire savoir que l’on souhaite connaître quelque chose.

Sémantique : Demander une information à un agent et attendre une réponse de l’agent.

```
context: Avatar::Request(in klg:Knowledge,  
in agt:Agent)  
pre: self→believe(agent→believe(klg))  
or self→believe((agent→believe(not  
klg)) )  
post: self→waitAnswerFromAgent(agent)
```

5.2 Quelques actions spécifiques

Le domaine d’application étant un monde virtuel discrétisé et informé, il est possible de définir un ensemble d’actions de base, qui sera valable pour la plupart des environnements virtuels d’apprentissage humain.

Ces quelques actions spécifiques sont données à titre d’exemple. Elle n’ont du sens que relativement à un domaine, mais peuvent être facilement étendues ou modifiées. Ici, l’utilisateur peut se déplacer librement dans un monde dont tous les éléments sont manipulables : ils peuvent être pris, déplacés et il est

possible d’altérer leur état.

5.2.1 Actions de manipulation

Modifier la position d’un objet :

Action : Déplacer (entité)

Définition : Ôter quelque chose de la place qu’il occupait pour le mettre à une autre place ou pour mettre autre chose à sa place.

Exemple : Tu dois déplacer le cube vert

Type et relations: <<MoveEntityAction>>

Sémantique : Déplacer une entité, modifier sa position (location) dans le monde.

```
context: Avatar::MoveEntity(in ent:Entity)  
post: not (ent.location =  
ent.location@pre)
```

Action : Placer (entité) (relation) (entité)

Définition : Mettre à une place déterminée ; disposer d’une certaine façon.

Exemple : Tu dois placer le cube rouge sur la table. Tu dois placer les miroirs derrière le cube

Type et relations: <<MoveEntityAction>>, contraire de *enlever (entité) de (entité)*

Sémantique : Déplacer une entité vers une région de l’espace relativement à une autre entité (qui peut être une zone, i.e. une entité sans forme).

```
context: Avatar::PlaceEntity(inout  
ent:Entity, in area:Entity, in  
tpR:TopologicalRelationship)  
def: tpRelation(in ent:Entity):  
Set(TopologicalRelationship)  
post: self^MoveEntity(ent) and  
tpRelation(ent,ar)→includes(tpR)
```

La méthode *tpRelation()* retourne la relation entre les deux entités : dans, sur, à droite, à gauche... Ces relations sont spécifiques au domaine et nous ne proposons pas de méthode d’évaluation de ses relations dans le présent article.

Action : Positionner (entité) en ...

Définition : Mettre à une place déterminée ; disposer d’une certaine façon.

Exemple : Tu dois positionner le cube rouge en (30,0,80)

Type et relations: <<MoveEntityAction>>

Sémantique : Déplacer une entité pour que sa position (location) soit égale au paramètre.

```
context: Avatar::PlaceAt(in ent:Entity, in
loc:Location)
post: self~MoveEntity(ent) and
(ent.location = loc)
```

Action : Approcher (entité) de (entité)

Définition : Approcher qqc. (de qqn ou qqc.). Le placer, le mettre près de ..., l’avancer près de ...

Exemple : Tu dois approcher la chaise de la table.

Type et relations: <<MoveEntityAction>>, contraire de *éloigner de*

Sémantique : Déplacer une entité en réduisant la distance minimum entre cette entité et quelque chose (zone ou autre entité).

```
context: Avatar::MoveEntityNear(in
ent:entity, in entCible:Entity)
post: self~MoveEntity(ent) and
ent→distance(entCible) <
ent@pre→distance(entCible@pre)
```

Déplacer d’une certaine façon :

Action : Tirer (entité)

Définition : 1-Amener après soi; traîner, derrière soi, en se déplaçant dans le même mouvement.

Exemple : Tu dois tirer le cube.

Type et relations: <<MoveEntityAction>>

Sémantique : Déplacer une entité en la tirant, i.e. en le tenant par le côté qui est ‘devant’ lors du déplacement.

```
context: Avatar::Tirer(in ent:Entity, in
handler:Handler)
def: sensDeplacement(In ent:Entity, In
handler:Handler, in locStart:Location, in
locEnd:Location): Real rapport entre le
vesteur de déplacement et le vecteur
‘‘prise’’ vers ‘‘centre entité’’
post: self~MoveEntity(ent) and
sensDeplacement(ent, handler,
ent.location@pre, ent.location) > 0
```

Action : Pousser (entité)

Définition : Exercer une pression avec sa force musculaire sur un corps pour le faire rouler, glisser, basculer, pivoter.

Exemple : Tu dois pousser le cube.

Type et relations: <<MoveEntityAction>>

Sémantique : Déplacer une entité en la poussant, i.e. en le tenant par le côté qui est ‘derrière’ lors du déplacement.

```
context: Avatar::Tirer(in ent:Entity, in
handler :Handler)
def: moveDirection(in ent:Entity, in
handler:Handler, in locStart:Location, in
locEnd:Location): Real
post: self~MoveEntity(ent) and
sensDeplacement(ent, handler,
ent.location@pre, ent.location) < 0
```

La méthode *moveDirection()* détermine le rapport entre le vecteur de déplacement et le vecteur ‘prise’ vers ‘centre entité’ (points informés de l’entité).

Action : Tirer (entité) vers soi

Définition : Amener vers soi. Déplacer dans une/sa direction en étant soi-même immobile.

Exemple : Tu dois tirer le cube vers toi.

Type et relations: <<MoveEntityAction>>, équivaut à *approcher* l’entité *de* soi

Sémantique : Déplacer une entité pour réduire la distance entre l’entité et soi.

```
context: Avatar::TirerVersSoi(in
ent:Entity, in handler:Handler)
post: self~MoveEntityNear(self,ent)
```

Modifier l’orientation d’un objet :

Action : Incliner (entité)

Définition : Placer, disposer avec un certain angle par rapport à un plan, en particulier celui de l’horizon. Mettre en position oblique ce qui est normalement ou naturellement vertical; courber vers le sol, faire pencher.

Exemple : Tu dois incliner le verre.

Type et relations: <<RotateEntityAction>>, inverse de *Redresser*

Sémantique : Modifier l’orientation de l’objet en le tournant autour de l’un des axes du plan horizontal.

```
context: Avatar::TiltEntity(in ent:Entity)
def: tilt(in ent:Entity): real
post: tilt(ent) > tilt(ent@pre)
```

La méthode *tilt()* calcule l’angle entre la verticale d’une entité et la verticale du monde (angle entre deux vecteurs).

Action : Redresser (entité)

Définition : Remettre quelqu’un ou quelque chose dans la position verticale, ou proche de la verticale, qui était antérieurement la sienne.

Exemple : Tu dois redresser l’armoire

Type et relations: <<RotateEntityAction>>, inverse de *incliner*

Sémantique : Modifier l’orientation de l’objet en le tournant autour de l’un des axes du plan horizontal.

```
context: Avatar::StraightenUpEntity(in
ent:Entity)
def: tilt(in ent:Entity): real
post: tilt(ent) < tilt(ent@pre)
```

Orienter un objet par rapport à un autre :

Action : Orienter (entité) vers (entité)

Définition : Déterminer la position de (quelque chose, plus rarement quelqu’un) par rapport aux points cardinaux, ou par rapport au soleil, ou à une direction, à un objet déterminé.

Exemple : Tu dois orienter la chaise vers la table.

Type et relations: <<RotateEntityAction>>

Sémantique : Orienter une entité de façon à ce que son vecteur direction (ou l’un de ses vecteurs informés) pointe dans la direction de l’entité cible.

```
context: Avatar::TurnToEntity(in
ent:Entity)
def: Entity :relativeAngle(in ent:Entity):
real post: self~TurnEntity() and
self.relativeAngle(ent) = 0
```

La méthode *relativeAngle()* calcule l’écart d’angle entre l’orientation de la première entité et l’angle du vecteur de la première entité à la deuxième entité.

5.2.2 Actions de navigation

Orienter le corps de l’avatar :

Action : Se Tourner vers (entité)

Définition : Décrire une courbe, un cercle ; effectuer un mouvement de rotation sur soi-même.

Exemple : Tu dois te tourner vers le mur.

Type et relations: <<RotateSelfAction>>

Sémantique : Tourner le corps de l’avatar autour de son axe vertical de façon à ce qu’il soit orienté vers une entité.

```
context: Avatar::TurnSelf(in ent:Entity)
def: Entity :relativeAngle(in ent:Entity):
real post: self~TurnSelf() and
self.relativeAngle(ent) = 0
```

Déplacer le corps de l’avatar :

Action : Aller (relation) (entité)

Définition : Le terme du mouvement est indiqué par un mot (subst. précédé d’une prép., ou adv.) désignant un lieu

Exemple : Tu dois aller dans le bureau 203.

Type et relations: <<MoveSelfAction>>

Sémantique : Se déplacer jusqu’à se trouver dans une zone qui valide la relation.

```
context: Avatar::MoveSelfTo(in
area:Entity, in
tpR:TopologicalRelationship)
def: tpRelation(in ent:Entity):
Set(TopologicalRelationship)
post: tpRelation(ent,ar)→includes(tpR)
```

Action : S’approcher de (entité)

Définition : Venir près de ..., aller se placer près de quelqu’un ou de quelque chose

Exemple : Tu dois t’approcher de l’oscilloscope

Type et relations: <<MoveSelfAction>>

Sémantique : Se déplacer en réduisant la distance minimum entre soi et une entité.

```
context: Avatar::MoveSelfNear(in
ent:Entity)
post: ent→distance(entCible) <
ent@pre→distance(entCible@pre)
```

5.2.3 Actions d’observation

Action : Regarder (entité)

Définition : Chercher à percevoir, à connaître par le sens de la vue. Chercher à connaître, à se rendre compte de quelque chose d’après certaines indications, certains indices. Diriger, fixer les yeux sur quelque chose, sur quelqu’un, sur un spectacle.

Exemple : Tu dois regarder le mur.

Type et relations: <<ChangeViewPoint>>

Sémantique : Modifier le point de vue pour qu’une entité soit affichée, au moins partiellement

```
context: Avatar::View(In ent:Entity)
def: isVisible(In
pdv:TopologicalProperty, In ent:Entity):
Boolean
post: isVisible(self,ent)
```

Action : Regarder vers (entité)

Définition : Diriger, fixer les yeux sur quelque chose, sur quelqu’un, sur un spectacle.

Exemple : tu dois regarder vers la clef

Type et relations: <<ChangeViewPoint>>

Sémantique : Modifier le point de vue pour que le regard (centre de la scène) soit dirigé vers l’entité

```
context: Avatar::LookAt(In ent:Entity)
def: isOnSight(in ent:Entity): Boolean
post: self.isOnSight(ent)
```

5.3 Exemples d’utilisation

Nous présentons ici quelques exemples de procédures métier (niveau M0).

5.3.1 Se rendre d’une pièce à une autre

Dans un monde permettant de naviguer dans un bâtiment (le CERV), l’apprenant doit se rendre d’un bureau à un autre, les portes étant fermées. Cette activité mêle des actions de navigation et de manipulation et plusieurs descriptions peuvent en être données. Le bâtiment est construit grâce à VEHA et comprend des entités visuelles **Pièces** et **portes** et des entités non représentées, qui définissent des zones de placement ou des zones d’interaction.

Une description grossière peut être essentiellement basée sur le résultat :

1. **Aller dans** le bureau 204

Une description plus fine montrera comment l’apprenant doit ouvrir les portes (il peut exister d’autres alternatives) :

1. **Aller dans** la sortie du bureau 201 (sortie = zone devant porte)
2. **Ouvrir** la porte
3. **Aller dans** le couloir (zone)
4. **Regarder vers** le bureau 204
5. **Regarder** l’entrée du bureau 204 (entrée = zone devant porte)
6. **Aller dans** l’entrée du bureau 204
7. **Ouvrir** la porte
8. **Aller dans** le bureau 204

5.3.2 Déplacer avec précision un objet sur une table

Dans un monde sans navigation, l’avatar de l’apprenant est debout devant une **table** où sont posés des **blocs** de plusieurs couleurs. Dans un exercice, l’apprenant doit placer précisément un bloc rouge pour que le centre de sa base (un point informé visible

par transparence dans la scène) soit à une position pA (identifiée par un point informé sur la table) avec seuil de position de 0.1 cm.

Si le monde permet une manipulation très discrétisée, selon une grille de déplacement ou avec des points informés magnétiques, il est facile d’atteindre des positions particulières. La description pourra être la suivante.

1. **Aller dans** zone autour table (zone dynamique autour de la table)
2. **Regarder vers** le bloc rouge
3. **Placer** le bloc rouge à pA

Dans un monde où la manipulation est plus continue, et la position pA n’est pas un point magnétique.

1. **Aller dans** zone autour table (zone dynamique autour de la table)
2. **Regarder vers** le bloc rouge
3. Jouer le cycle suivant, tant que $distance(bloc.location,pA) > seuil$
 - (a) **Approcher** le bloc rouge de pA
 - (b) **Regarder vers** le bloc rouge

6 Conclusion

Dans cet article, nous avons affirmé notre besoin d’une typologie d’actions pour décrire des activités. Nous avons défini le contexte de nos environnements – finis, structurés et informés – et de notre analyse, qui porte sur l’activité observable des utilisateurs, c’est-à-dire l’enchaînement d’actions médiées par l’interface Humain–Machine.

Après avoir présenté quelques systèmes montrant la particularité d’une interface explicite, nous avons proposé une typologie à plusieurs niveaux, s’appuyant sur des actions méta, portant sur des entités génériques, et réalisées par un avatar générique. Nous avons montré comment exprimer des actions spécifiques, dépendantes du domaine à partir de ce niveau méta.

Notre modèle n’a pas été validé et doit l’être sur les trois points suivants.

1. Le modèle et la typologie d’actions permettent-ils l’expression de plans d’actions attendues? Sont-ils accessibles à un expert du domaine? A un formateur?
2. Ce modèle permet-il de reconnaître l’activité? Suffit-il à un agent pour détecter l’exécution de plans attendus dans une trace d’activité?

3. Ce modèle peut-il servir de base pour qu’un agent informatique cognitif construise des plans d’actions et les exécute dans un monde virtuel ?

La perspective à court terme de ces travaux est l’utilisation de cette typologie pour exprimer et reconnaître des activités, d’abord dans un environnement très simple, puis sur un cas d’étude tel que celui du TP de Physique, un environnement VEHA qui reproduit une paillasse de travaux pratiques [Baudouin et al. 07b] [Baudouin et al. 07a].

Afin d’évaluer la pertinence de la typologie d’action, deux expérimentations sont prévues. Nous exposerons des tâches simples grâce à la typologie d’action (langage naturel), puis nous demanderons à des sujets de réaliser ces tâches en manipulant un environnement virtuel. Nous obtiendrons un corpus de traces de manipulations, reflétant la réalisation de ces tâches de plusieurs manières, ainsi qu’un enregistrement vidéo de ces passations.

La première expérience a pour objectif principal de valider la typologie du point de vue informatique, en fournissant la définition formelle des actions à un agent observateur pour qu’il détecte l’occurrence des actions dans les traces. Nous comparerons les résultats de l’agent aux instructions données aux sujets.

La seconde expérience consistera à présenter un enregistrement vidéo des passations des sujets à des juges, en leur demandant de déterminer, parmi la typologie en langage naturel, quelles actions ont été réalisées par les sujets. Nous comparerons ces notes manuelles aux occurrences détectées par l’agent afin de valider la cohérence de la typologie.

Par ailleurs, l’expression des contraintes liées aux actions se limite à des concepts géométriques très simples. Il n’est actuellement pas possible, ou assez difficile de dire qu’un objet se trouve sur ou à l’intérieur d’un autre objet [Aurnague et al. 97], [Morales et al. 07]. Nous envisageons d’étendre le méta modèle pour définir formellement ces relations pour pouvoir les appliquer dans des modèles VEHA et dans la description d’actions spécifiques.

Il est aussi possible d’améliorer l’expression de nos actions. Par exemple, il serait intéressant de définir formellement les relations entre les actions : parallélisme possible, occupation des mains, etc. Pour améliorer l’analyse de l’activité, nous voudrions aussi pouvoir évaluer qualitativement l’exécution des actions. Une action est-elle bien exécutée, à la bonne vitesse, présente-t-elle un danger ? Ces précisions nécessitent des informations supplémentaires sur l’action, mais également dans l’interface.

Références

- [Anderson 93] Anderson, J. (1993). *Rules of the Mind*. Lawrence Erlbaum Associates.
- [Aurnague et al. 97] Aurnague, M., Vieu, L., et Borillo, A. (1997). *Représentation formelle des concepts spatiaux dans la langue*, pages 69–102. Michel Denis.
- [Baudouin et al. 07a] Baudouin, C., Beney, M., Chevaillier, P., et Le Pallec, A. (2007a). Recueil de traces pour le suivi de l’activité d’apprenants en travaux pratiques dans un environnement de réalité virtuelle. *Revue Sticf, Sciences et Technologies de l’Information et de la Communication pour l’Éducation et la Formation*, 14. ISSN : 1764-7223.
- [Baudouin et al. 07b] Baudouin, C., Beney, M., Chevaillier, P., et Pallec, A. L. (2007b). Analysis of virtual helps usage in a virtual environment for learning. In *Intuition2007*.
- [Bernard et al. 02] Bernard, P., Dendien, J., Lecomte, J., et Pierrel, J. (2002). Un ensemble de ressources informatisées et intégrées pour l’étude du français : FRANTEXT, TLFi, Dictionnaires de l’Académie et logiciel Stella, présentation et apprentissage de leurs exploitations. *Actes de TALN*, 2 :3–36.
- [Card et al. 83] Card, S., Moran, T., et Newell, A. (1983). *The Psychology of Human-computer Interaction*. Erlbaum.
- [Chevaillier 06] Chevaillier, P. (2006). *Les systèmes multi-agents pour les environnements virtuels de formation*. Habilitation à diriger des recherches, Université de Bretagne Occidentale, Brest.
- [Chevaillier et al. 08] Chevaillier, P., Querrec, R., et Sept-seault, C. (2008). Veba : un méta-modèle d’environnement virtuel informé et structuré. *Revue TSI : Technique et Science Informatiques*, article soumis.
- [Fuchs 96] Fuchs, P. (1996). *Les interfaces de la réalité virtuelle*. Les Presses de l’Ecole des Mines de Paris.
- [Fuchs et al. 06] Fuchs, P., Moreau, G., Burkhardt, J., et Coquillart, S. (2006). L’interfaçage, l’immersion et l’interaction en environnement virtuel. In *Le traité de la réalité virtuelle (3e édition)*, volume 2. Les presses de l’Ecole des Mines, Paris.
- [Gerbaud et al. 08] Gerbaud, S., Ganier, F., Mollet, N., Arnaldi, B., et Tisseau, J. (2008). Gvt : A platform to create virtual environments for procedural training. In *in proceedings of IEEE VR*, Reno, Nevada.
- [Hannoun et al. 99] Hannoun, M., Boissier, O., Sichman, J. S., et Sayettat, C. (1999). MOISE : un modèle organisationnel pour la conception de systèmes multi-agents. In Gleizes, M.-P. et Marcenac, P., editors, *Ingénierie des systèmes multi-agents*, pages 105–118, Saint-Gilles, Ile de la Réunion, France. Hermes Science.
- [John et al. 96] John, B. et Kieras, D. (1996). The GOMS Family of User Interface Analysis Techniques : Comparison and Contrast. *ACM Transactions on Computer-Human Interaction*, 3(4) :320–351.
- [Lewis 00] Lewis, H. (2000). *Body Language : A Guide for Professionals*. Sage Publications.

- [Longo et al. 06] Longo, F., Mirabelli, G., et Papoff, E. (2006). Effective design of an assembly line using modeling and simulation. In *Proceedings of the 2006 Winter Simulation Conference*.
- [Maynard et al. 48] Maynard, H. B., Stegemerten, G. J., et Schwab, J. L. (1948). *Methods-time measurement*. McGraw Hill book company.
- [Morales et al. 07] Morales, A., Navarrete, I., et Sciacicco, G. (2007). A new modal logic for reasoning about space : spatial propositional neighborhood logic. *Ann Math Artif Intell*, 51 :1–25.
- [Newell 90] Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, Massachusetts : Harvard University Press.
- [OMG 06] OMG, editor (2006). *Object Constraint Language (OCL) specification*. Object Management Group. formal 06-05-01.
- [Querrec 02] Querrec, R. (2002). *Les systèmes multi-agents pour les environnements virtuels de formation. Application à la sécurité civile*. PhD thesis, UBO-ENIB.
- [Russell M. Taylor II et al. 01] Russell M. Taylor II, Hudson, T. C., Seeger, A., Weber, H., Juliano, J., et Helsen, A. T. (2001). Vrpn : A device-independent, network-transparent vr peripheral system. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology 2001, VRST 2001*.
- [Searle 69] Searle, J. (1969). *Speech Acts*. Cambridge University Press.
- [Zandin 03] Zandin, K. B. (2003). *MOST Work Measurement Systems*. Marcel Dekker.

7 Annexe : tableau de synthèse des systèmes à actions explicites

Nom du système	Rogue (Hexatron)	Maniac Mansion	Day of the Tentacle	Sam and Max hit the road	Fallout	GVT
Navigation	go left	walk to / aller	walk to / aller à	walk	Walking	aller à
	go right				Running	
	go up				rotate (self)	
	go down					
	go up stairs (>)					
	go down stairs (<)					
Manipulation	throw something (t)	pick up / prendre	pick up / prendre	Pick up	get item	prendre
	take object (automatic)	push / pousser	push / pousser	use	Use	poser
	drop object (d)	pull / tirer	pull / tirer		Use skill on (...)	tourner
	wield a weapon (w)	give / donner	give / donner		Use (...) on (...)	ouvrir
	wear armor (W)	open / ouvrir	open / ouvrir		drop item	fermer
	take armor off (T)	close / fermer	close / fermer			appuyer
	eat food (e)	use / utiliser	use / utiliser		Moving item (give to another)	augmenter
	call object (c)	fix / réparer	use (...) with (...) /		Unload ammo (on a weapon in inventory)	diminuer
	put on ring (P)	turn on / allumer	utiliser (...) avec (...)			allumer
	remove ring (R)	turn off / éteindre				éteindre
	(...)	unlock / Ouvrir Serrure				remettre à zéro
						serrer (tourner avec outil) (...)
Observation	read scroll (r)	read / Lire	Look at / regarder	Look at	Examine	Vérifier (j'affirme que P1 est vrai)
	identify object (i)	what is / Qu'est-ce				
	search for trap or secret door (s)					
Communication			talk to / Parler	Talk to (initiate dialog)	Talk	appeler formateur
				question (ask)	Push	demander aide
				declaration (say)		répéter
				non-sequitur		
				stop (stop talking)		
Autres	print help (?)	new kid/ changer de personnage		Use Max	Attack	
	fight (f)					