Définir des individualités pour des Personnages Non Joueurs

Tony Dujardin — Jean-Christophe Routier

Laboratoire d'Informatique Fondamentale de Lille - Université de Lille 1
Cité Scientifique, 59655 Villeneuve d'Ascq cédex, France
{tony.dujardin, jean-christophe.routier}@lifl.fr

RÉSUMÉ. *Concevoir le comportement des personnages non joueurs (PNJ), dans un jeu vidéo de type jeu de rôles, est un problème difficile tant du point de vue de la réalisation informatique que de la modélisation du comportement lui-même. Nous nous dégageons de ces obstacles par à l'approche centrée interactions, promue par le projet IODA-CoCoA, qui permet de proposer un moteur générique adaptable dans différents contextes. A ce moteur nous associons un mécanisme de sélection d'action basé sur les motivations. Ce mécanisme permet la définition de différents comportements de PNJ en jouant sur les paramètres des motivations et sans nouvelle programmation de code. Notre proposition constitue un moteur comportemental pour la modélisation de comportements de personnages situés. Par sa généricité, ce moteur peut être utilisé dans différents environnements (jeux) et pour différents agents en s'adaptant aux spécificités et capacités de chacun tout en proposant une diversité dans les comportements réalisables.*

ABSTRACT. *Designing the behaviour of non player characters (NPC), in role-playing video games, is a hard problem both from the programming point of view and from behaviour modelling considerations. We tackle these hurdles with the interaction oriented approach, promoted by the IODA-CoCoA project. It enables us to propose a generic engine suitable to several contexts. This engine is combined with an action selection mechanism based on motivations. This mechanism provides means to define several NPC behaviours by tuning the motivation parameters without new programming code. Our proposition constitutes a behavioural engine dedicated to the modelling of situated character behaviour. Thanks to its genericity, this engine can be used in various environment (ie. games) and for various agents, since it adapts to individual specificities and abilities while proposing a diversity in designed behaviours.*

MOTS-CLÉS : *comportement, sélection d'action, agent situé, jeux vidéo*

KEYWORDS: *behaviour, action selection, situated agent, video games*

RSTI - RIA – 23/2009. Jeux : modélisation et décision, pages 253 à 292

1. Introduction

Les CRPG (*Computer Role Playing Games*) sont des jeux vidéo permettant aux joueurs de s'immerger dans un monde fantastique peuplé de nombreuses créatures imaginaires. Le but de ce type de jeu est de faire évoluer un personnage dans ce monde afin de résoudre des quêtes. Les joueurs sont assistés dans leurs quêtes (ou en concurrence) avec des personnages gérés par l'ordinateur appelés PNJ (Personnages Non Joueurs). Le rôle des PNJ est très important dans un CRPG. En effet, plus les PNJ sont variés plus l'immersion dans le jeu est importante et plus le CRPG captive le joueur. Une des tâches du concepteur de CRPG est donc de différencier les PNJ, pas seulement au niveau graphique mais également au niveau de leur comportement.

D'un jeu à l'autre, certains éléments semblent communs et donc potentiellement réutilisables, par exemple les actions des PNJ comme attaquer, voler, ouvrir une porte, etc. Il en est de même pour les traits de caractère comme agressif, opportuniste, fainéant, serviable qui définissent le comportement du PNJ. Réutiliser les définitions des comportements pour d'autres jeux, signifie qu'ils doivent être «génériques» et indépendants de leur contexte d'exécution.

Une définition de comportement explicitement dépendante de son contexte d'exécution (comme l'environnement, l'état et les capacités du PNJ) peut entraîner une dépendance dans la conception. Ainsi les comportements sont souvent réalisés en dernier ou du moins après les éléments dont ils dépendent.

Anticiper cette dépendance de conception n'est pas toujours possible, comme par exemple avec les MMORPG (*Massively Multiplayer Online Role Playing Game*) qui constituent une catégorie spécifique des CRPG. Un MMORPG est une application en perpétuelle évolution : fréquemment l'environnement est étendu, les capacités des PNJ et des joueurs changent, de nouveaux éléments sont ajoutés au jeu, etc. Il est donc impossible d'attendre la fin de la conception du contexte d'exécution de l'agent pour définir ses comportements. Ces derniers doivent être réalisés parallèlement aux autres éléments du jeu dont ils dépendent. Une solution à ce problème est que les PNJ interagissent avec le joueur dans une zone limitée à un territoire restreint, ce qui réduit la dépendance du comportement à son contexte d'exécution et minimise les modifications à effectuer sur le comportement suite aux évolutions du jeu. Néanmoins, cette solution limite également les capacités et l'autonomie du PNJ.

Il existe de nombreux travaux sur la conception de comportements qualifiés de «réalistes» ou «proches de l'humain». J. Laird (Laird *et al.*, 2000) définit les jeux vidéo comme *the killer application* pour la modélisation de comportements humains. Dans la pratique, construire des agents autonomes dont la définition du comportement serait à la fois générique et indépendante du contexte d'exécution reste une tâche difficile.

Pour mieux le comprendre, nous allons au préalable présenter une architecture assez générale d'agent cognitif adaptée à la représentation d'un PNJ. Le comportement de l'agent est déterminé par les actions qu'il va effectuer pendant son exécution. Ces

actions sont choisies par l'agent afin de résoudre ses buts dans un environnement spécifique selon ses connaissances et ses capacités (voir figure 1).

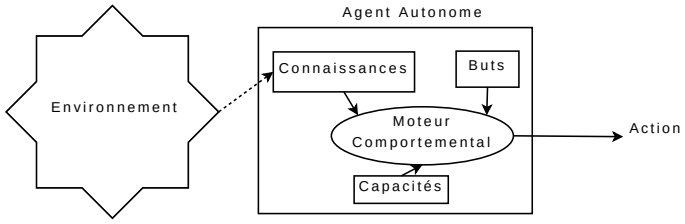

Figure 1. *Un agent autonome prend en compte ses connaissances (qui évoluent selon l'environnement), ses capacités et ses buts pour prendre ses décisions*

Dans un jeu, les personnages sont différents les uns des autres. Leurs buts et leurs connaissances évoluent au cours du temps. Néanmoins, ils utilisent le même moteur comportemental qui doit s'adapter aux évolutions du jeu, ce qui rend la conception des comportements difficile. Or, d'un jeu à l'autre, l'environnement, les capacités et les buts des agents sont différents, mais la problématique, c'est-à-dire la construction des comportements des PNJ, est la même. Nous proposons d'utiliser un seul moteur comportemental pour tous les jeux, capable de s'adapter aux changements des buts, des connaissances et des capacités des personnages.

Pour cela, nous proposons de ne pas gérer ces éléments fluctuants dans la définition du comportement. En effet, la définition du comportement ne dépend pas du contexte d'exécution, seuls ses effets en dépendent. C'est au moteur comportemental de se charger de l'appliquer dans un contexte particulier. Nous proposons d'utiliser un moteur comportemental pour les agents situés applicable à la représentation des PNJ. Ce moteur se décompose en deux parties distinctes que nous avons appelées **raisonnement** et **individualité** (voir 4).

Avant de présenter notre proposition, nous allons nous situer par rapport aux travaux et aux domaines connexes comme la planification, la mémorisation et la modélisation de comportement. Puis, nous définirons notre contexte de travail pour nous focaliser par la suite sur les définitions de *comportement*, *raisonnement* et *individualité* qui vont nous permettre d'introduire la construction des comportements à l'aide de l'approche centrée interaction et de l'approche basée sur les motivations. Avant de conclure, nous présenterons différentes simulations que nous avons réalisées afin d'illustrer les types de comportements que permet d'obtenir notre proposition.

2. Travaux connexes

La modélisation de comportement est un domaine très vaste, dans une approche cognitive, le comportement est construit à partir d'informations traitées par différents

modules comme la perception, la mémorisation, la planification et la sélection d'action. Notre proposition est une approche pour concevoir des comportements dirigés par les buts pour agents cognitifs. Nos agents réagissent aux événements qui correspondent à un ou plusieurs buts prédéfinis. Néanmoins, si nous considérons qu'un mécanisme de sélection d'action est un processus qui évalue et sélectionne la meilleure action parmi un ensemble d'actions exécutables, nous observons que ce mécanisme permet de définir les comportements pour les agents qu'ils soient réactifs ou cognitifs. En effet, seule la détermination des actions exécutables diffère, elle provient d'un processus de planification dans une approche cognitive et de stimuli internes ou externes à l'agent pour une approche réactive. Il en est de même pour les architectures mixtes comme InteRRaP (Müller *et al.*, 1993). Toutefois, bien que notre approche soit purement cognitive, notre proposition permet d'introduire des comportements qui semblent être réactifs comme l'opportunisme (nous détaillons ce comportement dans la section 5.4).

Nous reconnaissons que le module de planification est en soi une première étape de la construction du comportement puisqu'il calcule les résolutions possibles suivant les connaissances et les capacités de l'agent. Notre plan est construit à l'aide d'un simple chaînage arrière qui est suffisant pour présenter les résolutions au mécanisme de sélection d'action. D'autres planificateurs comme *SHOP* (Nau *et al.*, 1999), *SHOP-II* (Nau *et al.*, 2003) ou *Graphplan* (Blum *et al.*, 1995) sont plus évolués et proposent des propriétés intéressantes comme la détection d'actions mutuellement exclusives pour le Graphplan. Dans notre démarche, le module de planification n'est qu'un outil permettant de calculer les alternatives nécessaires à l'évaluation des actions exécutables. C'est pourquoi, nous distinguons bien le travail de conception de l'individualité de l'agent issu du mécanisme de sélection d'action de la conception du plan que nous ne détaillerons pas. Nous nous focalisons principalement sur la construction de l'individualité qui permet aux agents de faire des choix différents même s'ils ont calculé exactement les mêmes résolutions.

La phase de planification repose sur les connaissances de l'agent. Il est évident que la qualité des connaissances issues du module de mémorisation de l'agent et de leur représentation influence son comportement. Nous considérons que les faits mémorisés sont vrais tant qu'ils ne sont pas mis à jour par la perception. En effet, le processus de mémorisation n'est ici qu'un outil pour la phase de construction des résolutions. La prise en compte de l'incertitude n'est pas l'objectif de notre proposition. Cette considération permet d'obtenir une planification qui est certainement plus efficace. Néanmoins, nous ne nous intéressons pas uniquement à la résolution des buts, mais plus à la manière de les résoudre, c'est-à-dire la sélection d'action.

Nous nous distinguons sur ce point des solveurs comme SOAR (Laird *et al.*, 1987) ou ACT-R (Anderson *et al.*, 2004), qui cherchent à résoudre un problème selon une politique prédéfinie. Ces derniers permettent de résoudre une problématique de manière optimale en testant les solutions et en apprenant de leur erreurs. Ceci permet notamment d'obtenir des résolutions complexes comme la possibilité d'anticiper les actions des adversaires (Laird, 2001). Certains comme ACT-R ont un processus d'ap-

prentissage basé sur des informations partielles, ce qui permet de s'adapter aux environnements partiellement observables. ACT-R introduit également un facteur de bruit dans le choix des règles à exécuter ce qui lui permet de s'adapter aux changements de son contexte d'exécution. Enfin ACT-R gère de la connaissance déclarative et de la connaissance procédurale. A l'inverse de ce genre de solveurs, nous privilégions la modélisation d'individualité (aussi appelée personnalité) basée sur des définitions de comportements réutilisables par d'autres agents et d'autres applications et surtout des définitions qui sont applicables sans phase d'apprentissage au préalable.

Nous nous distinguons également de l'approche à base de scripts dont les défauts sont reconnus et ont déjà été soulignés (Tozour, 2002). Même si certaines propositions contournent les défauts de cette approche (Ponsen *et al.*, 2004; Spronck *et al.*, 2006), l'utilisation de scripts oblige à être très dépendant du contexte d'exécution, ce qui limite la réutilisation des comportements d'un jeu à l'autre. Notre approche permet de définir les comportements indépendamment du contexte d'exécution permettant ainsi leur réutilisabilité pour d'autres personnages et d'autres jeux.

Dans notre proposition, les travaux que nous rejoignons le plus sont certainement ceux qui se focalisent sur le mécanisme de sélection d'action. Nous comparerons évidemment notre mécanisme de sélection d'action, et plus précisément une implémentation concrète de notre mécanisme, avec les critères définis par Tyrrell (1993) dans la section 5.5. Toutefois, parmi les nombreux travaux sur la modélisation de comportement et sur les mécanismes de sélection d'action, nous nous rapprochons des travaux de Sevin (2006) et de Andriamasinoro (2003). Tous deux proposent une architecture pour la conception de comportements d'humains virtuels basés sur la notion de motivation. Ils se basent notamment sur les travaux du psychologue Américain Maslow (1998) qui propose une hiérarchisation des besoins plus connue sous le nom de «pyramide de Maslow». Bien que ces propositions soient décrites comme génériques, leur utilisation n'en est pas pour autant facilitée et le paramétrage du système limite la réutilisation des comportements. De Sevin propose d'utiliser des algorithmes génétiques pour construire de nouvelles règles et d'utiliser des techniques d'apprentissage pour réduire l'espace de recherche du problème et améliorer le paramétrage de la hiérarchie des règles. Or ces deux techniques, pour être les plus efficaces possibles, doivent être liées au contexte d'exécution, ce qui réduit la réutilisation des comportements ainsi définis. Andriamasinoro reconnaît lui-même que le travail que doit fournir l'utilisateur n'est pas évident et certaines tâches comme la liaison entre les actions peuvent être facilitées par la simulation elle-même, ce qui rend sa proposition également dépendante du contexte d'exécution. Notre proposition nécessite uniquement de définir les actions que les personnages peuvent effectuer (dans la partie raisonnement) et son comportement (dans la partie individualité). Ces deux définitions sont effectuées indépendamment du contexte d'exécution. Le seul paramétrage que notre proposition nécessite est lié à la spécificité du comportement (que l'on appellera le profil comportemental) et non à son application dans un contexte particulier.

3. Contexte de travail

3.1. *De IODA à CoCoA*

Notre approche entre dans le cadre du projet CoCoA et permet la modélisation de comportements réutilisables. CoCoA (pour Cognitive Collaborative Agents) est une plateforme de simulation pour agents cognitifs situés. Ce projet repose sur la méthodologie centrée interaction nommée IODA (*Interaction-Oriented Design of Agent simulations*) (Mathieu *et al.*, 2001; Kubera *et al.*, 2008) pour la modélisation des capacités des agents. Dans cette approche, un agent peut effectuer et peut subir des interactions dans l'environnement. Le principe est de faire correspondre les interactions peut-subir avec les interactions peut-effectuer. Par exemple, un agent *arbre* peut subir l'interaction *couper*, *casser* et *brûler*. Un agent *bûcheron* peut effectuer les interactions *ouvrir* et *couper*. Donc l'agent bûcheron peut effectuer l'interaction couper sur un agent arbre. Dans IODA, certains agents (les actifs) peuvent effectuer les interactions et d'autres (peut être les mêmes) peuvent les subir (les cibles). Nous appellerons par la suite, une **capacité** d'un agent, une interaction qu'il **peut effectuer**.

3.1.1. *Interaction*

Une interaction définit comment un agent acteur peut agir sur un agent cible. La dynamique du monde peut s'exprimer ainsi :

$$a \in \text{Actif}, t \in \text{Agent}, \text{si } \exists\, i \mid a.\text{peut-effectuer}(i) \cap t.\text{peut-subir}(i), \text{alors } a.\text{exécuter}(i, t).$$

Les interactions décrivent les lois qui régissent le monde simulé et constituent une connaissance manipulable par les agents.

Définition 1 *Une **interaction** est définie par un nom et 3 parties :*

– la condition, *teste le contexte d'exécution de l'interaction, ce qui consiste à tester principalement les valeurs de la cible ou les propriétés de l'acteur ;*

– la garde, *vérifie les conditions d'application de l'interaction dans l'environnement, notamment pour l'aspect situé, le plus souvent cette garde sert à exprimer la distance minimale d'exécution d'une interaction. Cette condition est spécifique aux agents situés, c'est pourquoi nous la distinguons de la condition ;*

– une action, *décrit les conséquences de l'exécution de l'interaction et agit sur les propriétés de l'agent cible.*

Par exemple, l'interaction *ouvrir* (voir figure 2) un objet (porte, fenêtre, etc.) effectue le changement de l'état *fermé* à l'état *ouvert* de la cible de l'interaction. La nature de la cible n'a aucune importance ici (il suffit qu'elle puisse subir l'interaction), la connaissance peut ainsi être représentée d'une manière «universelle» pour l'interaction. Dans ce sens, les interactions sont des connaissances déclaratives : elles

décrivent une action mais pas comment la résoudre ou l'utiliser. Ainsi les interactions sont réutilisables pour différents agents ou différentes applications.

$$ouvrir : \begin{cases} \texttt{condition} & = & \texttt{target.opened} = \texttt{false} \\ \texttt{guard} & = & \texttt{distance}(\texttt{actor}, \texttt{target}) < 1 \\ \texttt{action} & = & \texttt{target.opened} = \texttt{true} \end{cases}$$

Figure 2. *L'interaction ouvrir est définie par une condition (que la cible soit fermée), une garde (ouvrir à une distance inférieure à 1) et une action (faire passer l'état de la cible à ouvert)*

Toutes les informations sont contenues dans l'interaction, ainsi en utilisant cette approche le moteur qui les manipule est le même pour toutes les simulations.

3.1.2. *Agents situés*

Pour représenter les PNJ, nous utilisons des agents situés dans leur environnement. C'est-à-dire qu'ils évoluent dans un espace euclidien, qu'ils perçoivent leur environnement et agissent en conséquence. L'environnement est composé de places et d'obstacles. Il ne comporte aucune information sur le comportement des PNJ. Les agents situés CoCoA sont également capables de déterminer la position d'autres agents ou d'obstacles de l'environnement et de raisonner sur les notions de distance ou de voisinage. Dans l'environnement, on peut trouver deux types d'agents : les inanimés et les animés (ne pas confondre avec les agents mobiles et non mobiles). Ces deux types d'agents possèdent des propriétés et peuvent subir des interactions, mais les agents animés peuvent également en effectuer et ont un moteur comportemental. Les agents animés sont cognitifs et proactifs. Les agents inanimés correspondent plus à des éléments du jeux.

3.1.3. *Agents cognitifs*

La structure des agents animés est présentée dans la figure 3. Cette structure correspond aux agents cognitifs du projet CoCoA (Devigne *et al.*, 2007). Les agents sont non omniscients, ils ne connaissent pas tout leur environnement et n'ont aucun *a priori* sur sa topologie, la présence d'autres agents, leur position ou leur état avant le début de l'exécution. Ils découvrent leur environnement dont ils ont une vision limitée par le module de perception. Les informations perçues par l'agent sont collectées et ajoutées dans sa mémoire (sa base de connaissances noté KB) qui présente une version dégradée de son environnement. En effet, les informations en dehors de la zone de perception de l'agent peuvent ne pas être à jour. Les informations en dehors du champ de perception sont donc potentiellement erronées. Les agents animés ont des buts. La satisfaction de ces buts les mène à planifier des résolutions. Le module de planification détermine les actions que l'agent peut effectuer dans l'environnement afin de résoudre ses buts. Le plan est produit par une chaînage arrière sur les interactions *peut-effectuer* à partir des buts de l'agent. Ce plan est construit en fonction des capacités de l'agent mais également des informations contenues dans sa mémoire (ses

connaissances). Cette phase de planification permet donc de calculer les différentes résolutions possibles en fonction de ses capacités et de ses connaissances courantes. Ces résolutions contiennent un ensemble d'actions exécutables qui représentent les interactions dont les conditions et la garde sont satisfaites. Le module de sélection d'action va déterminer la meilleure action à exécuter.

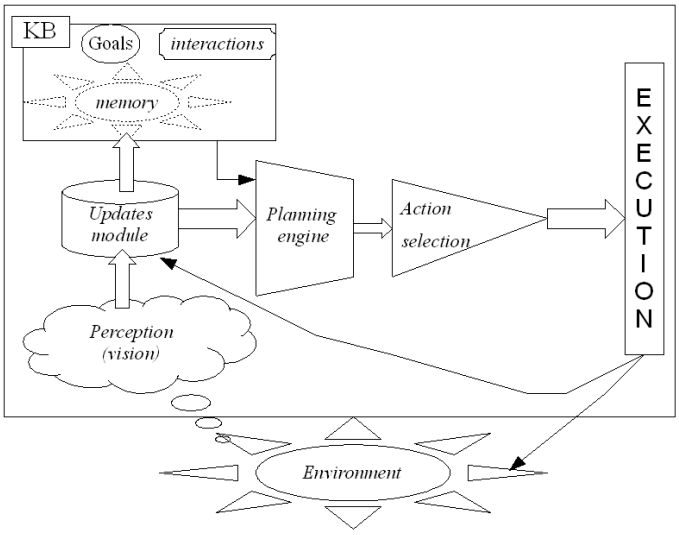

Figure 3. *Les éléments des agents cognitifs CoCoA : l'agent possède dans sa base de connaissance (KB) la description des interactions, ses buts et une mémoire qui est mise à jour par le module de perception. Le module de planification s'appuie sur ces connaissances pour établir le plan. Le module d'exécution tente d'exécuter une action choisie préalablement par le module de sélection d'action. Les modifications induites par l'exécution et les nouvelles informations détectées par le module de perception sont répercutées dans la mémoire par le module de mise à jour*

Le plan est construit selon les connaissances que l'agent possède dans sa mémoire. L'agent est donc capable de s'adapter aux changements de l'environnement. De plus, l'approche centrée interaction permet une séparation des connaissances déclaratives et du processus de planification procédural. Le plan construit dépend des capacités de l'agent. De ce fait, le même moteur peut être utilisé par plusieurs agents avec des capacités différentes et ce dernier n'est pas sensible aux évolutions de l'environnement ou des capacités de l'agent. Grâce à la méthodologie IODA, les capacités des agents sont réutilisables pour d'autres agents et d'autres contextes d'exécution.

3.1.4. *Planification : influence du situé*

Nos travaux ne s'inscrivent pas directement dans le thème de la planification dans la mesure où nous ne cherchons pas à établir une nouvelle proposition dans ce do-

maine. Cependant nous sommes utilisateurs dans la mesure où nos agents doivent planifier pour chercher à résoudre leurs buts.

Notre utilisation de la planification et des interactions est présentée dans (Devigne *et al.*, 2007). Il est cependant nécessaire de préciser un certain nombre de points concernant des contraintes imposées au planificateur, en particulier à cause du contexte situé de l'environnement et la nécessité d'exécution effective du plan. Nous ne cherchons pas à obtenir des comportements optimaux, que ce soit en termes de déplacements ou de nombre d'actions effectuées par exemple. Ainsi nous ne visons pas l'obtention du «meilleur» comportement (et encore faudrait-il définir le critère d'évaluation). En même temps, nous voulons obtenir des comportements plausibles, que doit donc nous proposer le planificateur. Il est probablement important de rappeler maintenant que dans le type de simulations que nous visons, les jeux, les comportements sont exécutés «pas à pas» et qu'il est donc possible de suivre en détail la démarche du personnage (l'agent).

Ces derniers points : plausibilité des comportements observés et exécution pas à pas, prennent une importance particulière du fait que nos environnements de simulation sont *situés*. Pour pouvoir exécuter les actions commandées par leur comportement et interagir avec les autres éléments de la simulation, les agents vont en effet être amenés à se déplacer dans l'environnement. Dans une exécution «pas à pas», ces déplacements vont constituer une part importante du comportement observé et représenteront un facteur majeur pour évaluer la rationalité du comportement, d'autant plus qu'ils sont particulièrement visibles pour l'observateur extérieur. Le caractère situé de nos simulations va donc avoir un impact sur les plans d'action des agents. A la séquence d'actions calculée pour résoudre les objectifs de l'agent s'ajoutent les déplacements nécessaires pour atteindre les positions de l'environnement concernées dans le cadre de la simulation. Évidemment, pour un même objectif et une même séquence d'actions, les déplacements nécessaires varieront pour des environnements différents. En outre, il est possible que, pour pouvoir être exécuté, un déplacement requière l'exécution d'autres actions qui devront être intégrées à la séquence initiale. Ainsi on peut distinguer ce que nous appelons le *plan abstrait* du *plan d'exécution*. Le premier correspond au plan d'actions dans lequel les positions relatives ou absolues des agents ne sont pas prises en compte, il est indépendant d'un environnement situé. Le second correspond au plan dû à l'exécution dans un environnement particulier.

Le plan abstrait fournit les actions permettant la résolution effective des objectifs. Le caractère situé intervient quand un agent exécute une action impliquant une interaction avec un autre agent. Le plus souvent, des contraintes de proximité entre les deux agents s'appliquent afin que l'interaction puisse effectivement avoir lieu. En conséquence, il est fort possible qu'avant de pouvoir exécuter son (inter)action, l'acteur doive se déplacer pour s'approcher de sa cible. Ce déplacement d devra donc être intégré au plan pour permettre le déroulement de la simulation. Le plan initial «*exécuter a*», devient donc *exécuter d puis a* (voir figure 4).

Mais cela soulève un nouveau problème lorsqu'un déplacement requiert à son tour une planification pour pouvoir être exécuté correctement. C'est par exemple le cas

lorsque ce déplacement amène l'agent à franchir une porte fermée. L'ouverture de cette porte et la ou les actions qu'elle nécessite devront être également intégrées au plan. Le déplacement *d* est donc décomposé en deux sous-déplacements d_1 et d_2 interrompus par l'ouverture de la porte. On en arrive donc au plan d'exécution *exécuter* d_1, *ouvrir la porte, exécuter* d_2 *puis a*. Les choses se complexifient encore un peu plus lorsque l'ouverture de la porte n'est pas atomique et que la résolution de cet objectif nécessite à son tour une planification. Planification qui entraîne la production d'un plan abstrait et de son plan d'exécution contenant ses propres déplacements qui ont un impact sur ceux déjà planifiés. C'est par exemple le cas lorsque la porte en question est cadenassée et que l'agent doit préalablement prendre la clé nécessaire et donc se déplacer jusque celle-ci avant d'aller à la porte. Le sous-plan préalable à l'ouverture de la porte est donc *exécuter le déplacement d'*$_1$, *prendre la clef, exécuter d'*$_2$, *décadenasser la porte*. Il remplace le déplacement d_1 qui n'a plus lieu d'être. Le schéma récursif de ce raisonnement est évident et l'on peut imaginer que la récupération de la clef engendre de nouvelles actions. On le voit sur ce petit exemple, un plan abstrait initial simple comme l'était *exécuter a* se voit fortement enrichi du fait du contexte situé. Ce point est également repris dans (Devigne *et al.*, 2004).

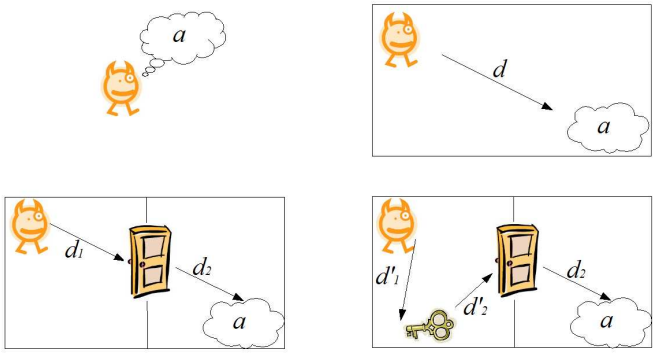

Figure 4. *Influence du situé sur la planification. En haut à gauche, le plan abstrait. En haut à droite, dans un contexte situé le déplacement doit être prévu pour atteindre l'endroit d'exécution de l'action. En bas à gauche, l'environnement impose une planification, ici ouverture d'une porte. En bas à droite, cas où l'agent sait - ou croit - que la porte est cadenassée et doit donc prendre la clef avant de s'y rendre. (les actions* ouvrir *et* décadenasser *n'apparaissent pas sur les figures)*

Il reste un dernier cas à évoquer qui est celui de l'*exploration* de l'environnement. L'agent évolue dans un environnement qu'il doit découvrir, dont il a une perception locale et qui, de plus, est soumis à des changements. Si, lors d'une planification, est identifiée la nécessité d'utiliser un objet dont l'agent ne connaît pas la position (comme la clef permettant d'ouvrir une porte), le chaînage va produire l'introduction dans le

plan d'une action d'*exploration* afin de permettre la découverte de l'objet manquant et donc la résolution du but.

Nous allons maintenant nous focaliser sur notre proposition qui permet de construire des comportements pour les PNJ. Le moteur comportemental que nous proposons s'adapte aux contextes évolutifs et permet de réutiliser la définition du comportement pour d'autres agents et d'autres contextes d'exécution.

4. La définition du comportement

Dans (Seth, 1998) le comportement est défini par trois entités que sont le personnage, l'environnement et un observateur. Cette définition basée sur l'éthologie présente le comportement comme le résultat de l'ensemble des actions qu'un agent peut effectuer dans un environnement. D'après cette définition, le comportement observable dépend des actions que l'agent peut effectuer, c'est-à-dire ses capacités. Par exemple, un agent qui ne pourrait que casser des portes pour les traverser pourrait être perçu comme plus brutal qu'un agent qui aurait la capacité de les ouvrir. Ensuite, le comportement résulte des actions qui sont exécutées dans l'environnement par l'agent. Une action exécutée correspond au résultat d'un processus de sélection parmi les actions exécutables par l'agent à chaque instant. Ces actions exécutables sont obtenues après un processus de **raisonnement** qui définit les solutions possibles à la résolution de ces buts. Le processus de sélection s'effectue en fonction de l'**individualité** de l'agent (voir figure 5). Nous distinguons bien le raisonnement de l'agent qui est souvent la tâche confiée à un planificateur, de son individualité qui est gérée par un mécanisme de sélection d'action.

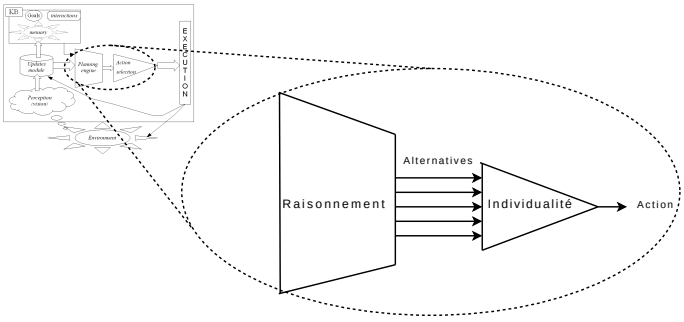

Figure 5. *Représentation du comportement : le comportement dépend de ce que l'agent peut effectuer à chaque instant et de son individualité*

4.1. *Le raisonnement des agents*

Le processus de planification fournit à partir des connaissances de l'agent, de ses capacités (interactions) et des buts qu'il doit résoudre, les résolutions possibles pour chaque but. Nous appelons **alternative** une telle résolution et **raisonnement** le processus de calcul des alternatives.

Définition 2 *Une **action exécutable** est une interaction dont la garde et les conditions d'exécution sont satisfaites.*

Définition 3 *Une **alternative** est une séquence d'actions allant d'une action exécutable jusqu'à la résolution d'un but.*

Pour un but donné, il existe zéro ou plusieurs alternatives possibles, correspondant aux différentes manières de résoudre ce but. Le mécanisme de sélection d'action évalue les alternatives fournies par le processus de planification.

Définition 4 *Un **planificateur** est un processus qui calcule les alternatives possibles pour résoudre un ensemble de buts à partir des capacités de l'agent et en fonction des informations contenues dans sa mémoire.*

$$\textit{Planificateur} : \quad Memoire \times Interaction^* \times But^* \rightarrow Alternative^*$$

4.1.1. *Le raisonnement avec un arbre-et/ou*

Nous présentons ici les différentes définitions ci-dessus avec le cas où le plan est représenté par un arbre-et/ou. Un *arbre-et/ou* est une structure arborescente alternant **nœud-et** et **nœud-ou** de père en fils. Les nœuds-ou représentent les actions permettant de satisfaire une condition (un but ou un nœud-et), dans les nœuds-et se trouvent les conditions nécessaires à l'exécution du nœud père. Nos arbres-et/ou présentent en racine le but à réaliser et en feuilles les actions exécutables.

Un *but* est un ensemble de conditions qui doivent être satisfaites. A chaque but b de l'agent a correspond l'arbre de planification associé (noté $Plan_a(b)$).

$$\forall a \in Agent, \forall b \in But_a,$$
$$\begin{array}{rccl} Plan_a : & But_a & \rightarrow & Arbre \\ & b & \mapsto & Plan_a(b) \end{array}$$

Dans un arbre-et/ou, une *action exécutable* est représentée par une feuille d'un arbre de planification.

$$\forall a \in Agent, \forall b \in But_a,$$
$$\begin{array}{rccl} ActionExecutable_a : & But_a & \rightarrow & Feuille \\ & b & \mapsto & x \end{array}$$
$$ActionExecutable_a(b) \in \{x \mid x = Feuille(Plan_a(b))\}$$

Dans un arbre-et/ou, une *alternative* est composée de toutes les actions issues d'un nœud-et et d'un choix parmi les actions issues d'un nœud-ou.

$$\begin{array}{lllll} Alternative_a : & (Arbre & \times & ActionExecutable_a) & \rightarrow & \{Action\}^* \\ & (Plan_a(b) & \times & feuille) & \mapsto & \{x\}^* \end{array}$$

Chaque arbre de planification contient les différentes alternatives (les différentes suites d'actions possibles) permettant de résoudre le même but.

4.1.2. *Collecte d'une alternative dans un arbre-et/ou*

Pour collecter les actions d'une alternative à partir de l'action exécutable, nous avons mis au point un algorithme (voir la figure 6). Cet algorithme fait appel à la fonction $N(x)$ qui prend en entrée une action exécutable, afin de collecter les actions issues d'un nœud-et et de faire un élagage de l'arbre en effectuant un choix parmi les branches issues d'un nœud-ou. Dans cet algorithme, nous utilisons trois fonctions externes qui sont $Pere$, $Fils$ et $Choice$ dont voici la définition :

– $Pere(y)$ est la fonction qui renvoie l'ancêtre du nœud y.

– $Fils(y)$ est la fonction qui renvoie l'ensemble des fils du nœud y.

– La fonction $Choice$ permet, en évaluant les actions présentes dans les sous-alternatives, d'élaguer l'arbre en effectuant un choix parmi les propositions fournies par les nœud-ou. Notons qu'il est possible de faire appel au mécanisme de sélection ou à certains de ses composants (ses évaluateurs voir la section 4.2.1.2) comme fonction $Choice$.

$N(x)$ est la fonction qui collecte les actions de l'alternative à partir de l'action exécutable x, en remontant jusqu'à atteindre un but.

$$N(x) = \begin{cases} \emptyset & \text{si } x = racine \\ \{x\} \cup (P(Pere(x))) & \text{sinon} \end{cases}$$

Où $P(x)$ est la fonction qui collecte les actions parmi les ancêtres du nœud courant en remontant parmi les pères et en descendant parmi les fils qui sont des nœuds-et

$P(x) = N(Pere(x)) \cup \bigcup_{s \in Fils(Pere(x)) \setminus x} E(s)$

Où $E(x)$ est la fonction qui sélectionne la meilleure alternative des nœuds-et afin d'élaguer l'arbre.

$E(x) = A(f) \mid f \in Fils(x)$ et $Choice(A(f)) = \max_{k \in Fils(x)}(Choice(A(k)))$

Où $A(x)$ est la fonction qui récolte les nœuds action parmi les fils issus d'un choix.

$A(x) = \{x\} \cup (\bigcup_{f \in Fils(x)} E(f))$.

Figure 6. *Algorithme de calcul des alternatives dans un arbre-et/ou pour une action exécutable*

Indépendamment de son contexte d'exécution (l'environnement d'exécution et les buts de l'agent), le comportement de l'agent dépend de ses capacités et de ce que nous appelons son individualité. La méthodologie centrée interaction et ses avantages pour la modélisation des capacités de l'agent ont été rapidement présentés, pour plus de détails se référer aux articles correspondants (Mathieu *et al.*, 2001; Kubera *et al.*, 2008; Devigne *et al.*, 2007). Nous nous focalisons dans la suite de cet article sur la construction de l'individualité des agents pour définir leur comportement.

4.2. *L'individualité des agents*

Nous avons défini préalablement que le comportement de l'agent dépend de ses capacités et de son individualité. Les capacités sont représentées par les interactions qui sont réutilisables. Nous définissons l'individualité des agents.

Définition 5 *L'**individualité** est l'ensemble des facteurs propres à l'agent qui l'influencent le choix des actions qu'il exécute. Deux agents dans le même environnement, qui ont les mêmes capacités, les mêmes buts à réaliser et les mêmes connaissances, ont deux individualités différentes s'ils choisissent d'exécuter des interactions différentes à un moment donné de la simulation.*

Le choix de l'action étant évidemment cohérent avec les propriétés de l'agent, nous ne considérons pas un mécanisme purement aléatoire comme permettant de définir l'individualité même s'il permet d'obtenir des choix d'actions différentes (puisqu'aléatoire).

À partir de cette définition de l'individualité, nous pouvons déduire deux propriétés de l'individualité. La première est que l'observation de l'individualité est liée aux choix d'actions effectuées par le personnage et donc à son mécanisme de sélection d'action, c'est donc dans ce mécanisme que l'individualité des personnages est construite, pour orienter le choix des actions à exécuter. La seconde est que l'individualité dépend de propriétés comportementales qu'il faut définir. Nous proposons d'utiliser l'approche basée sur la notion de motivations pour définir les propriétés comportementales et d'adapter cette approche dans un mécanisme de sélection.

4.2.1. *Motivations*

Le comportement résulte de l'ensemble des actions qu'un agent a effectuées dans un environnement. Ces actions sont sélectionnées parmi les capacités de l'agent. Nous pouvons remarquer que la sélection d'action est un processus dépendant de plusieurs contraintes. Prenons par exemple *manger* : un humain peut préférer aller au restaurant plutôt que de cuisiner, car il est fatigué. Ou alors, il peut préférer manger des plats surgelés, car il n'a pas les moyens d'aller au restaurant. Ainsi, le comportement est orienté (de façon positive ou négative) par des **tendances**. La notion de tendances repose sur une théorie du comportement psychologique développée par Albert Burloud (Burloud, 1936). En accord avec ce constat, nous proposons de définir notre vision du

comportement comme les actions effectuées qui résultent d'un ensemble de tendances subies par l'agent.

4.2.1.1. Approche par motivation

Dans (Ferber, 1995) les tendances sont définies par les cognitons (i.e. particules élémentaires permettant de définir l'état mental d'un agent selon Ferber) qui poussent ou contraignent un agent à agir ou qui l'empêchent d'agir. Elles sont issues de combinaisons de motivations qui sont des cognitons plus élémentaires. Le fonctionnement de l'approche par motivation est expliqué au travers d'un système conatif proposé par l'auteur, ce dernier est présenté dans la figure 7. Dans ce système, les motivations forment la base de l'élaboration des tendances qui contraignent la décision de l'agent. L'auteur propose une classification des motivations en quatre catégories suivant les origines des motivations :

motivations personnelles : ce sont les motivations qui procurent un certain plaisir à l'agent. Par exemple, avoir faim est une motivation personnelle qui va pousser l'agent à chercher de la nourriture.

motivations provenant de l'environnement : ce sont les motivations qui proviennent d'éléments de l'environnement de l'agent. Par exemple, percevoir de la nourriture ou être à proximité d'elle, va pousser l'agent à manger.

motivations sociales : ce sont les motivations liées à la société qui pousse l'agent à agir d'une certaine façon. Par exemple, il est préférable d'avoir un travail et de bien le faire.

motivations relationnelles : ce sont les motivations provenant des autres agents. Par exemple un agent va en aider un autre à porter une charge qui est trop lourde pour un seul agent.

Chaque motivation peut orienter le comportement de l'agent selon les tendances qu'elles fournissent. Nous proposons une adaptation de l'approche par motivation basée sur la notion d'alternative en utilisant un mécanisme de sélection d'action (voir 4.2.2.1). Comme dans l'approche de Ferber, chaque motivation va produire une tendance allant de l'*attraction* à la *répulsion* afin d'influencer le choix de l'ASM. Une attraction va pousser l'agent à réaliser une action alors qu'une répulsion va le freiner.

Dans notre proposition nous distinguons deux cas particuliers de tendances qui sont la *neutralité* et l'*inhibition*. La *neutralité* correspond à une tendance n'influençant pas le mécanisme de sélection d'action, l'*inhibition* correspond à une répulsion absolue qui amène le mécanisme de sélection d'action à écarter l'action considérée sans même tenir compte des autres motivations.

4.2.1.2. Adaptation des motivations pour la construction de l'ASM

Le module (ou mécanisme) de sélection d'action (noté ASM pour *Action Selection Mechanism*) a la charge de sélectionner une action parmi l'ensemble des actions exécutables $\mathcal{A}^R$ identifiées par le moteur de planification. Classiquement, un ASM

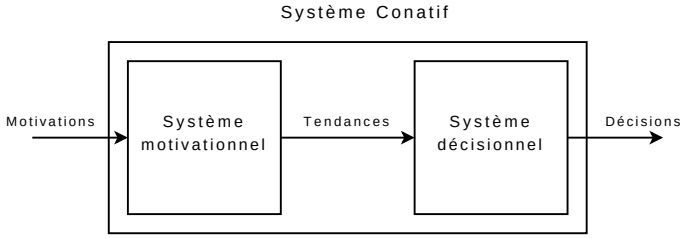

Figure 7. *Le système conatif proposé par Ferber est composé de deux sous-systèmes : le système motivationnel qui élabore les tendances et le système décisionnel qui décide des actions à effectuer en fonction des tendances*

affecte à chaque action exécutable appartenant à $\mathcal{A}^R$ une valeur numérique et sélectionne l'action possédant la plus grande valeur. La valeur est calculée comme une combinaison de *critères d'évaluations* ou *évaluateurs*. Chaque évaluateur e_i est défini par une fonction γ_{e_i} :

$$\begin{array}{rccc} \gamma_{e_i}: & \mathcal{A}^R & \rightarrow & \mathbb{R} \\ & a & \mapsto & valeur \end{array}$$

Pour chaque action a dans $\mathcal{A}^R$, la note finale ϕ utilise une fonction de combinaison $Comb$ pour agréger les n-évaluateurs :

$$\begin{array}{rccc} Comb: & \mathbb{R}^n & \rightarrow & \mathbb{R} \\ \phi: & \mathcal{A}^R & \rightarrow & \mathbb{R} \\ & a & \mapsto & Comb(\gamma_{e_1}(a), \cdots, \gamma_{e_n}(a)) \end{array}$$

Classiquement, l'ASM sélectionne l'action exécutable α ayant obtenu la meilleure (la plus grande) valeur :

$$ASM(\mathcal{A}^R) = \alpha \text{ où } \alpha \in \mathcal{A}^R \mid \phi(\alpha) = \max_{a \in \mathcal{A}^R}\{\phi(a)\}$$

Nous proposons d'adapter l'approche par motivation pour construire le mécanisme de sélection d'action. Ainsi, une fois identifiées les motivations de l'individualité de l'agent, chacune peut être représentée par un évaluateur spécifique. La fonction de combinaison des évaluateurs sert de système décisionnel afin de déterminer la meilleure action à exécuter. Dans notre approche, les tendances ne correspondent pas aux évaluations de chaque action exécutable (comme dans une approche classique présentée ci-dessus), mais à l'évaluation de son alternative.

Définition 6 *Un **ASM** est un processus qui calcule la meilleure action à exécuter en fonction de la définition de l'individualité de l'agent et à partir des alternatives identifiées par le système de raisonnement.*

4.2.2. *Des motivations aux évaluateurs*

Chaque motivation possède une description unique quel que soit le contexte d'application. Par ce principe, chaque motivation (tout comme les interactions) peut être réutilisée pour d'autres agents et d'autres applications. Il est ainsi aisé de définir l'individualité par les motivations qu'il subit. Par exemple, prendre en compte la caractéristique «être opportuniste» d'un agent, requiert «simplement» de définir un évaluateur pour exprimer cette motivation. Si on définit la motivation d'opportunisme comme *l'attirance à effectuer les interactions exécutables dont la cible est proche*, cette motivation est du type *provenant de l'environnement*. Le moteur comportemental étant le même quelles que soient l'application et les motivations de l'agent, il est possible d'ajouter et d'enlever une motivation sans perturber le processus de sélection d'action. De plus, chaque évaluateur est compréhensible puisqu'il correspond à une motivation spécifique.

4.2.2.1. Un mécanisme de sélection d'action basé sur la notion d'alternative

Nous redéfinissons le processus de sélection d'action, afin qu'il devienne un processus de sélection d'une ou plusieurs alternatives dont la première étape est l'action exécutable. L'utilisation de l'alternative permet d'obtenir deux propriétés de l'ASM. La première est un apport sur la qualité de la sélection d'action, la seconde permet la réutilisabilité des motivations.

Par définition, une alternative correspond à une résolution possible, elle contient donc la prévision (pas l'anticipation) des actions que l'agent devra exécuter. L'évaluation des alternatives permet de prendre en compte l'ensemble des actions à exécuter. Or, dans un processus de sélection d'action, les oscillations entre les différentes actions sont à proscrire dans la mesure du possible (Tyrrell, 1993). Ainsi, nous ne voulons pas qu'un personnage soit engagé dans la résolution d'un but et qu'il soit par la suite amené à remettre en cause cette résolution parce qu'une des actions de l'alternative s'avère en fin de compte incompatible avec la définition de son individualité. Considérons deux alternatives a_1 et a_2 qui permettent de résoudre un même but, mais chacune correspond à des actions exécutables différentes, α_1 et α_2, et à des chemins de résolution (au moins partiellement) différents. L'ASM doit tenir compte de l'ensemble des actions impliquées dans a_1, resp. a_2, pour promouvoir α_1, resp. α_2. Supposons, que l'ASM privilégie α_1 qui à court terme semble préférable, cela signifie que l'agent s'engage dans a_1 et en traite les actions. Il ne faut pas qu'après quelques pas de résolution, cet agent se trouve confronté à une action de a_1 que l'ASM rejetterait pour réorienter l'agent sur a_2, suite à une inhibition sur cette action par une motivation, par exemple. Dans une telle situation, l'ASM doit favoriser immédiatement α_2. L'utilisation de l'alternative dans l'évaluation des actions exécutables guide l'ASM vers a_2, et donc vers α_2, même si à court terme, α_1 semblait la meilleure action à exécuter. L'ASM doit prendre en compte la prévision sur les actions futures du personnage à moyen terme, il considère donc toutes les actions qui apparaissent dans l'alternative. Il en résulte que la sélection d'action ne doit pas se focaliser sur le choix de l'ac-

tion préférée à chaque pas, mais sur la meilleure résolution, c'est-à-dire la meilleure alternative.

Le processus de planification est le même quel que soit le contexte d'exécution. En effet, toute l'information est contenue dans l'interaction et la construction des alternatives est indépendante du contexte d'exécution. De plus, les évaluations s'effectuent sur les alternatives qui contiennent des connaissances procédurales manipulées par le planificateur. Le processus d'évaluation peut donc être défini indépendamment du contexte d'exécution puisque l'alternative contient déjà les informations nécessaires. Notre construction des motivations et des évaluateurs correspondants est ainsi indépendante du contexte d'exécution. Chaque évaluateur peut être réutilisé pour d'autres agents et d'autres applications.

4.2.2.2. Évaluer une alternative en fonction de la motivation

Pour construire l'individualité de l'agent, il faut au préalable définir le comportement que l'on souhaite obtenir, c'est-à-dire définir quelles sont les motivations subies par l'agent dans le jeu et sur quelles parties de l'alternative la motivation doit focaliser son évaluation. En effet, dans une alternative, nous pouvons distinguer trois parties (voir la figure 8) :

La première correspond à **ce que l'agent doit faire** : c'est le but qui lui a été attribué et que l'agent doit résoudre (B_i).

La seconde correspond à **ce que l'agent prévoit de faire** : c'est la séquence d'actions que l'agent prévoit de faire pour résoudre son but après avoir effectué l'action exécutable ($a_{i_1}, a_{i_2}, a_{i_3}, ..., a_{i_n}$).

Enfin la troisième correspond à **ce que l'agent peut faire** : c'est l'action exécutable (α).

Pour un but et une action exécutable donnés, il existe plusieurs alternatives associées. Ces alternatives sont différenciées par la partie correspondant à **ce que l'agent prévoit de faire**, c'est-à-dire qu'elles sont différenciées selon la manière dont l'agent prévoit de résoudre son but.

$$B_i \longrightarrow a_{i_1} \longrightarrow a_{i_2} \longrightarrow a_{i_3} \longrightarrow ... \longrightarrow a_{i_n} \longrightarrow \alpha_i$$

Figure 8. *Une alternative est composé de trois parties : le but (B_i), les actions prévues par l'agent ($a_{i_1}, a_{i_2}, a_{i_3}, ..., a_{i_n}$) et l'action exécutable (α_i)*

REMARQUE. — sur l'ensemble des motivations, il faut que chaque partie de l'alternative soit évaluée. Néanmoins le choix des parties à évaluer dépend en premier lieu de la définition même de la motivation. Par exemple, parmi les motivations présentées en section 5.4, la motivation d'opportunisme est liée au voisinage de l'agent, ainsi par opportunisme, un agent qui est près d'une pomme va la prendre. Cette motivation dépend de la distance entre l'agent et la cible, cette information se retrouve à partir

de l'action exécutable, c'est-à-dire *ce que l'agent peut faire*. Alors que la motivation de l'influence du but dépend du but (*ce que l'agent doit faire*) et la motivation des préférences de l'agent dépend à la fois de *ce que l'agent prévoit de faire* et de *ce que l'agent peut faire*.

Chaque évaluateur représente une motivation de l'agent. Chaque motivation dépend de paramètres propres à l'agent. Pour un contexte donné, l'influence qu'exerce une motivation sur le comportement de l'agent peut varier d'un agent à l'autre. Par exemple, la motivation d'opportunisme est liée à la notion de voisinage, c'est-à-dire être *«proche de l'agent»*. Cette notion dépend d'une distance à partir de laquelle l'agent considère qu'un autre agent est proche de lui ou non. Il faut donc définir l'*expression* de chaque motivation. Ainsi les mêmes évaluateurs sont utilisés par tous les agents et la variation du comportement est obtenue grâce aux choix de paramètres et non par une différence de pondération des évaluateurs. Nous appelons cette phase de spécialisation des motivations la construction du profil comportemental.

Définition 7 *Un **profil comportemental** est l'ensemble de paramètres qui s'appliquent sur les motivations de l'agent.*

REMARQUE. — Le profil comportemental est le seul paramétrage nécessaire dans notre approche. Celui-ci est lié à la spécificité de la définition du comportement et non à son application dans un contexte particulier. Ainsi, malgré cette spécialisation du profil comportemental, les évaluateurs restent indépendants du contexte d'exécution : le même profil comportemental peut être réutilisé pour d'autres agents et d'autres jeux.

4.2.2.3. Combinaison des évaluations

Nous avons vu précédemment que les comportements sont dépendants des capacités et de l'individualité de l'agent. Nous avons également défini l'individualité comme étant la définition et l'expression des motivations. Dans notre cas, la prise de décision est faite en choisissant la meilleure combinaison des évaluations de chaque alternative. La fonction de combinaison $Comb$ permet de définir l'évaluation des résolutions possibles en fonction l'individualité de l'agent (voir la figure 9).

4.2.2.4. Fonction de combinaison

Chaque motivation donne son «avis» (évaluation) sur une alternative. Le rôle de la fonction de combinaison est d'agréger ces évaluations afin d'obtenir une évaluation globale. La fonction de combinaison joue un rôle similaire à un «arbitrator» de DAMN (Rosenblatt, 1997) qui «votent» pour l'action à exécuter. Il existe plusieurs méthodes pour combiner les motivations, telles que la prise en compte d'un choix social (Arrow, 1951) pour représenter les «avis» des motivations. Il existe également plusieurs fonctions mathématiques utilisables comme fonction de combinaison, chacune ayant des propriétés pouvant influencer la sélection d'action. Toutefois, parmi

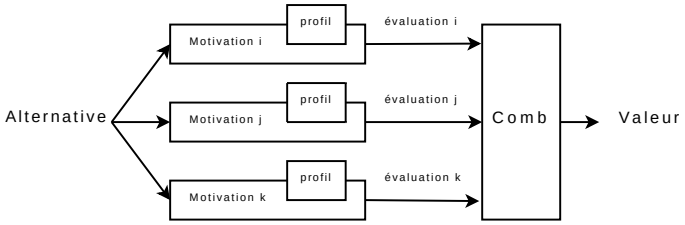

Figure 9. *Chaque alternative est évaluée par les motivations de l'agent. Ses motivations sont paramétrées par son profil comportemental. Les évaluations (ou tendances) sont combinées pour obtenir l'appréciation d'une action par l'individualité de l'agent*

l'ensemble des propriétés de ces fonctions mathématiques, nous en avons identifié deux essentielles. La fonction de combinaison doit :

– permettre aux motivations d'exprimer la neutralité, la répulsion, l'attraction et l'inhibition,

– permettre l'ajout et la suppression de motivations en gardant la cohérence de l'agrégation par rapport à l'individualité désirée.

Du choix de la fonction de combinaison et de l'intervalle de valeurs utilisés dépend la construction des évaluateurs. C'est pourquoi cette fonction doit être définie avant la construction des évaluateurs.

5. Réalisation du comportement

Après avoir présenté nos principes de modélisation d'individualités comportementales à travers le mécanisme de sélection d'action, nous présentons dans cette section une concrétisation de cette proposition adaptée au contexte applicatif que nous nous sommes fixé : celui des personnages non joueurs pour jeux vidéo de type jeu de rôles. Notre objectif n'est pas de pouvoir dérouler un scénario prédéfini mais d'avoir des agents autonomes capables d'évoluer en accord avec leur propre définition comportementale, dans l'environnement. Ainsi le paramétrage de l'ASM n'est pas conçu dans le but de résoudre un problème précis (la résolution étant assurée par le planificateur), mais de fournir à l'agent un guide de comportement qui oriente ses choix d'action. Nous commencerons donc par présenter l'ASM utilisé dans CoCoA qui concrétise les principes de modélisation d'individualités comportementales présentés ci-dessus puis nous l'illustrerons à travers des petits exemples. Chaque expérience présentée dans ce document a été mise en place et testée sur la plateforme de simulation CoCoA (voir section 6).

5.1. *Concrétisation de l'ASM : choix des motivations*

La mise en place de notre ASM se décompose en trois étapes :

1) identifier toutes les motivations désirées et pertinentes pour le contexte applicatif,

2) définir la fonction de combinaison $Comb$ qui sera utilisée,

3) en tenant compte de la fonction de combinaison choisie, définir pour chaque motivation un évaluateur et sa fonction d'évaluation γ.

Les étapes 1 et 2 étant indépendantes et afin d'éviter des redondances, nous aborderons le choix de la fonction de combinaison en premier, puis nous présenterons les différentes motivations et la réalisation des évaluateurs correspondants.

5.2. *Le contexte applicatif*

Prenons le temps de considérer le contexte applicatif. Qu'attendons-nous d'un personnage non joueur ? Celui-ci a, le plus souvent, plusieurs buts à gérer en concurrence. Ceux-ci vont de la résolution de «missions» à la gestion de ses éléments «vitaux». La granularité de ces buts et leur importance sont variables. Ils nécessitent la mise en place de comportements qui pourront être perçus tantôt comme cognitifs, lorsqu'il s'agira de construire un plan pour résoudre un but à moyen ou long terme comme c'est le cas des «missions», tantôt comme réactifs, lorsqu'il s'agira de prendre immédiatement un élément de l'environnement comme fuir un ennemi puissant qui vient d'apparaître. On comprend bien qu'une réorganisation des priorités relatives des buts doit donc être possible.

De plus, les agents évoluent dans un environnement géographique dont l'impact sur le comportement est nécessairement important. Ainsi la situation (ou position) de l'agent dans cet environnement a une influence sur ses choix, ne serait-ce que parce qu'il doit se déplacer dans cet environnement pour exécuter telle ou telle action à un endroit précis de cet environnement. La crédibilité du comportement perçu du PNJ sera en partie jugée sur ces déplacements, une gestion qui paraîtrait «déraisonnable» des déplacements pénaliserait le jugement. Ainsi, lorsque deux actions équivalentes peuvent être exécutées, mais la cible de l'une est très éloignée alors que celle de l'autre est proche, on peut s'attendre à ce qu'un personnage choisisse celle dont la cible est la plus proche. Le mécanisme de sélection d'action doit donc prendre en compte l'environnement.

Enfin, nous souhaitons pouvoir exprimer des *individualités* différentes pour les différents personnages. L'individualité s'exprime à travers de nombreuses facettes et, d'une certaine manière, on peut considérer que tout facteur influençant l'ASM est un élément constituant cette individualité. Cependant, pour des PNJ de jeux vidéo, ce que l'on veut à travers l'individualité c'est également exprimer des *traits de caractères*. Il faut considérer donc que les choix de l'action exécutée doivent exprimer ces traits de

caractères. Ou, inversement, que l'on peut vouloir transcrire des traits de caractères en favorisant l'exécution de telle ou telle action.

Voici donc un ensemble de caractéristiques qui doivent être prises en compte par l'ASM, celles-ci vont se traduire par un ensemble de motivations que nous allons présenter dans la section 5.4.

Les préoccupations de notre mécanisme de sélection d'action pour PNJ rejoignent celles exprimées dans les travaux de Tyrrell (1993) ou Schmidt (2005) qui concernaient les *animats*. Les motivations que nous retenons doivent permettre de satisfaire les critères exprimés dans ces travaux. Nous comparerons notre proposition avec ces travaux une fois celle-ci exposée. Nous allons commencer par préciser la fonction de combinaison que nous avons retenue avant de présenter les motivations mises en œuvre.

5.3. *La fonction de combinaison*

Comme nous l'avons précédemment dit, chaque évaluateur «note» chaque alternative et les valeurs obtenues sont agrégées à l'aide de la fonction de combinaison $Comb$. Nous avons choisi une fonction de combinaison permettant aux évaluateurs d'exprimer la neutralité, la répulsion, l'attraction et l'inhibition. Nous considérons des valeurs choisies dans $\mathbb{R}^+$ et que la fonction de combinaison $Comb$ est l'opérateur de multiplication. Ainsi, pour un évaluateur, la valeur 1 est interprétée comme neutre, une valeur au dessus de 1 est une attraction, une valeur comprise entre 0 et 1 est une répulsion et la valeur 0 implique une inhibition puisqu'elle annule même les effets des autres évaluations. Cette fonction possède les propriétés mentionnées en section 4.2.2.4.

REMARQUE. — précisons ici qu'il s'agit de *notre* choix de fonction de combinaison. Celui-ci, s'il peut sembler trivial, n'en satisfait pas moins les propriétés qui nous intéressent pour cette fonction. Il doit cependant être clair que notre modèle de mécanisme de sélection d'action n'empêche nullement d'envisager une autre fonction de combinaison.

5.4. *Les motivations*

Nous présentons ici à la fois les motivations que nous avons retenues pour la mise en place du comportement de nos PNJ et la définition de leur évaluateur. Nous présentons plusieurs motivations qui se répartissent dans les catégories présentées en section 4.2.1.1. Il s'agit de motivations *provenant de l'environnement* et de motivations *personnelles*. Pour chaque motivation, nous précisons le travail dévolu au concepteur, c'est-à-dire la définition du profil comportemental. Cependant, soulignons que ce travail n'a pas nécessairement à être répété systématiquement pour chaque agent. Il est évident qu'un même profil peut être partagé par plusieurs agents. Le concepteur peut

par exemple décider que tous les agents partagent le même profil d'opportunisme (i.e. la même valeur de θ_{opp}).

5.4.1. *Les motivations personnelles*

Dans cette catégorie, nous retenons cinq motivations, l'influence des buts, les préférences de l'agent, l'achèvement en temps, la revalorisation multibut et l'inertie.

L'influence des buts. Un agent possède différents buts, l'importance d'un but est intrinsèquement variable, cette importance peut varier pendant la simulation et peut également dépendre d'une propriété de l'agent. Pour mettre en œuvre cette importance, nous associons à chaque but une fonction dont la valeur indique la priorité du but. Prenons l'exemple où le but «survivre» dépend du niveau d'énergie de l'agent. Plus ce niveau est faible, plus l'agent est en «mauvais état» et donc plus le but de «survivre» devient important et plus cette motivation influence la sélection d'action (voir figure 10). Pour cette motivation, le travail à la charge du concepteur est de définir la fonction de priorité pour chaque but.

$$f(x) = \begin{cases} 0 & \text{si } x \geq 20 \\ MAX & \text{si } x \leq 0 \\ \frac{MAX}{1+\exp\frac{x-m}{\lambda}} & sinon \end{cases}$$

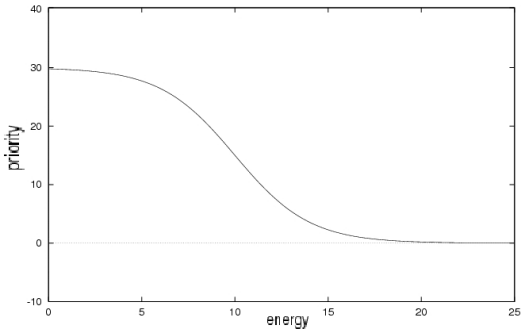

Figure 10. *Exemple de fonction représentant l'importance du but «garder son niveau d'énergie au dessus de 20». La fonction f utilisée est inspirée d'une sigmoïde. Avec MAX la valeur maximale que nous avons fixé à 30, m permet de régler le centre de symétrie que nous avons fixé à 10 et λ qui contrôle la pente (λ = 2 dans cet exemple)*

Les préférences de l'agent. Nous voulons attribuer des traits de personnalité à notre agent, il peut par exemple être brutal, pacifique, gourmand ou non. Chaque agent a des préférences différentes sur les actions qu'il peut effectuer. Ces préférences influencent la sélection d'action et donc le comportement de l'agent. Ainsi, suivant ces préférences, l'observateur pourra identifier différents traits de caractère. Par exemple, en attribuant une préférence plus forte pour *casser* que pour *ouvrir*, nous poussons l'agent à faire le choix de casser une porte plutôt que de l'ouvrir. Ainsi, l'observateur peut interpréter ce comportement, comme étant une conséquence du trait de caractère «brutal» de l'agent. Une préférence exprime la prédisposition de l'agent à exécuter une action a, elle correspond à un nombre $\pi(a)$ réel positif ou nul. Une préférence permet

d'exprimer l'attirance ($\pi(a) > 1$), la répulsion ($\pi(a) < 1$), l'inhibition ($\pi(a) = 0$) ou la neutralité ($\pi(a) = 1$) de l'agent par rapport à une action.

Cette motivation prend en compte les préférences de l'agent sur les actions qu'il peut exécuter. L'évaluation par cette motivation se base sur toutes les actions contenues dans l'alternative associée à l'action exécutable évaluée. Les préférences de toutes les actions de l'alternative (action exécutable incluse) sont collectées et combinées pour produire la valeur. L'évaluateur, γ_{pref}, est défini ainsi :

$$\text{Soit } preference(\alpha) = \{a_i \in Alternative(\alpha) \mid \pi(a_i) \neq 1\}$$
$$\forall \alpha \in \mathcal{A}^R, \gamma_{pref}(\alpha) = \begin{cases} 0 \text{ si } \exists a_i \in Alternative(\alpha) \mid \pi(a_i) = 0 \\ Pref(preference(\alpha)) \end{cases}$$

où $Alternative$ (voir l'exemple dans un arbre-et/ou en section 4.1.2) est une fonction qui collecte les actions présentes dans l'alternative de l'action exécutable α, $Pref$ est la fonction qui combine les préférences π de l'agent sur les actions collectées. $preference_{agent}$ ne contient que les actions dont la préférence π_{agent} est différente de 1, c'est-à-dire les préférences non neutres.

Nous avons testé plusieurs fonctions de combinaison des préférences ($Pref(X)$) différentes et retenu une solution basée sur la *moyenne harmonique* (Moy_{harm}) pour combiner les préférences de l'agent.

$$Pref(X) = Moy_{harm}(X) = \frac{\|X\|}{\sum_i^{\|X\|}(\frac{1}{\pi(x_i)})}, x_i \in X$$

Pour cette motivation, le travail à la charge du concepteur est de définir les préférences (π) de l'agent sur chaque interaction qu'il est capable d'effectuer.

L'achèvement en temps. Chaque action peut prendre plus ou moins de temps à être exécutée. Cela peut dépendre par exemple du niveau de l'agent dans une compétence particulière.

Par cette motivation, le coût en temps de chaque action est pris en compte afin de favoriser les résolutions prenant le moins de temps.

L'achèvement en temps, $achT$, est défini ainsi :

$$\forall \alpha \in \mathcal{A}^R, achT(\alpha) = 1 + \log_{achT}(\frac{\theta_{achT}}{cost(Alternative(\alpha))})$$

où $cost$ est le coût pour exécuter l'alternative $alternative$ du but auquel mène l'action α. Le paramètre θ_{achT} est le seuil de l'achèvement en temps. En dessous de ce seuil, le coût est considéré comme faible et l'action est favorisée. Au-delà, le coût est trop important et l'action est pénalisée. Pour cette motivation, le travail à la charge du concepteur est de définir le seuil de l'achèvement en temps (θ_{achT}) de l'agent.

La revalorisation multibut. Une action exécutable qui apparaît dans la résolution de plusieurs buts fait progresser simultanément, lorsqu'elle est exécutée, plusieurs buts de l'agent. En ce sens, elle est favorisée. C'est le rôle de cette motivation qui s'exprime ainsi :

$$\forall \alpha \in \mathcal{A}^R, multi(\alpha) = 1 + log(nbmulti(\alpha))$$

où $nbmulti(\alpha)$ est le nombre de fois où l'action α apparaît en tant qu'action exécutable.

Pour cette motivation, le concepteur n'a besoin que d'activer ou non cette motivation qui n'est pas paramétrable.

L'inertie. Quand un agent s'engage dans une résolution (une alternative), la motivation *inertie* exprime la tendance de l'agent à poursuivre dans la même résolution. Elle est définie ainsi :

$$\forall \alpha \in \mathcal{A}^R,$$
$$inertia(\alpha) = \begin{cases} 1 + \theta_{inertia} & if \alpha \in das \\ 1 & else \end{cases}$$

où das est la dernière alternative sélectionnée. Le paramètre $\theta_{inertia}$ correspond au bonus d'inertie. Pour cette motivation, le travail à la charge du concepteur est de définir le bonus d'inertie ($\theta_{inertia}$) de l'agent.

5.4.2. *Les motivations provenant de l'environnement*

Dans cette catégorie, nous retenons deux motivations, l'une basée sur la réactivité de l'agent face à son environnement et la seconde basée sur la prise en compte des déplacements dans la résolution des buts.

L'opportunisme. L'agent doit tenir compte de son environnement immédiat. Les agents sont situés, ils peuvent évaluer les distances entre eux et d'autres entités, il est donc possible de savoir si la cible d'une action exécutable est proche ou non. Cette motivation introduit un comportement réactif, en favorisant les actions dont les cibles sont proches, permettant à un agent d'être temporairement détourné d'une résolution à cause de la proximité d'une cible.

Dans le but de favoriser par opportunisme les actions dont la cible est proche de l'agent, nous définissons un seuil θ_{opp}, qui correspond à la portée de la motivation de l'opportunisme. En dehors de cette portée, la cible n'a pas de tendance et les actions exécutables reçoivent un évaluateur neutre. A l'intérieur, plus la cible est proche, plus l'évaluation est forte. L'évaluateur, γ_{opp}, de la motivation de l'opportunisme est défini ainsi :

$$\forall \alpha \in \mathcal{A}^R, \gamma_{opp}(\alpha) = \max\left(1, 1 + \log_{\theta_{opp}}\left(\frac{\theta_{opp}}{dist(a, cible(\alpha))}\right)\right)$$

où a est notre acteur et $cible$ la fonction retourne la cible de l'action α et $dist$ calcule la distance entre 2 positions (à l'aide d'un A^*).

Pour cette motivation, le travail à la charge du concepteur est de définir le seuil de l'opportunisme (θ_{opp}) de l'agent.

L'achèvement en espace. Cette motivation valorise les résolutions courtes en termes de déplacements, cette évaluation est tributaire des connaissances de l'agent

sur son environnement. Ainsi cette motivation poussera l'agent à favoriser les actions qui demandent «peu» de déplacements aux dépens de celles dont la cible est plus éloignée. Cette motivation s'exprime ainsi :

$$\forall \alpha \in \mathcal{A}^R, achS(\alpha) = 1 + \log_{achS}(\frac{\theta_{achS}}{nbspace(Alternative(\alpha))})$$

où $nbspace$ est le nombre de pas de déplacements requis pour exécuter l'alternative $alternative$ du but dont dépend l'action α. Le paramètre θ_{achS} est le seuil de cette motivation, en-dessous le déplacement est considéré faible et donc α sera favorisée, au-delà l'action est pénalisée. Pour cette motivation, le travail à la charge du concepteur est de définir le seuil de l'opportunisme (θ_{achS}) de l'agent.

5.4.3. *Commentaires*

Nous ne commenterons pas l'ensemble des évaluateurs mais attirons l'attention du lecteur sur certains points à l'aide de deux évaluateurs en particulier :γ_{pref} et γ_{opp}.

– γ_{pref} peut exprimer de l'inhibition à l'attraction alors que γ_{opp} exprime seulement l'attraction ou la neutralité ;

– γ_{pref} est calculée sur l'ensemble des actions contenues dans l'alternative correspondant à une action exécutable donnée, ce mode de calcul prend en compte les préférences de l'agent sur les actions qu'il pourrait être amené à effectuer (quelle que soit la simulation), il n'est donc pas perturbé par l'ajout ou le retrait de capacités de l'agent ;

– γ_{opp} est calculée sur l'action exécutable, l'opportunisme se base sur la notion de distance entre la cible de l'action exécutable et notre agent, ce calcul est effectué sans *a priori* sur la topologie de l'environnement, les modifications dans l'environnement ou le changement d'environnement ne perturbe pas son fonctionnement ;

– Enfin la motivation sur la *personnalité de l'agent* prend en compte les actions à moyen terme, cette motivation amène l'agent vers un comportement cognitif, alors que *l'opportunisme* révèle un comportement réactif.

5.5. *Confrontation aux propositions existantes*

Nos comportements sont définis à l'aide de motivations, en ce sens nous nous rapprochons des travaux de Schmidt (2005) sur le modèle PECS (*Physical conditions Emotional state Cognitive capabilities Social status*) qui fonctionne en quatre étapes (voir le tableau 1). Nous proposons néanmoins des motivations telles que la personnalité qui permet de définir les «traits de caractère» de nos personnages.

Notre proposition est également conforme aux critères de la proposition de Blumberg (1994) qui considère qu'un ASM doit à la fois permettre l'opportunisme tout en conservant une certaine persistance afin d'éviter des oscillations entre les actions, ce que nous prenons en compte grâce à la motivation *inertie*. Tyrrell (1993) décrit les contraintes qui doivent être prises en compte pour évaluer un bon mécanisme de sélec-

Etape	Approche PECS	Notre proposition
1	Calculer les valeurs des variables d'état interne	Calculer les priorités des motivations des buts
2	Calculer l'intensité de chaque motivation	Calculer les évaluations
3	Sélectionner la motivation la plus importante	Sélectionner l'alternative ayant eu la meilleure note ϕ
4	Effectuer l'action sélectionnée	Effectuer l'action sélectionnée

Tableau 1. *Comparaison de l'approche PECS avec notre proposition*

tion d'action. Nous proposons un mécanisme de sélection d'action concret permettant de prendre en compte ces contraintes dans une approche cognitive pour la modélisation de PNJ dans un environnement virtuel. Le tableau 2 présente en quoi notre ASM satisfait ces critères.

Critères de Tyrrell	Notre proposition
Persister jusqu'à l'achèvement d'une action	L'inertie influence l'agent afin qu'il finisse l'alternative qu'il est en train de résoudre
Avoir des activations proportionnelles à l'état courant.	La motivation des buts peut être utilisée comme variable homéostatique
Avoir une concurrence équilibrée entre les actions	Pas de discrimination des actions achevant un seul but, mais revalorisation multibut
Avoir la continuité des séquences d'actions	L'inertie est transmise au père de l'action exécutée, l'évaluation des actions exécutables se fait sur l'ensemble des actions de l'alternative
Permettre l'interruptibilité si nécessaire	Les motivations de l'environnement peuvent valoriser des actions, permettant d'interrompre une résolution en cours
Profiter de l'avantage qu'offrent des opportunités.	La motivation de l'opportunisme
Combiner des préférences	Prise en compte de toutes les actions de l'alternative
Avoir une combinaison flexible des stimuli	Plusieurs fonctions de combinaison sont utilisables

Tableau 2. *Les critères de Tyrrell que nous satisfaisons*

6. Expérimentations

6.1. *Expérimentations élémentaires*

Nous allons présenter ici des exemples simples, exécutés avec CoCoA (voir l'interface de simulation en figure 11), permettant d'illustrer tour à tour un certain nombre de motivations.

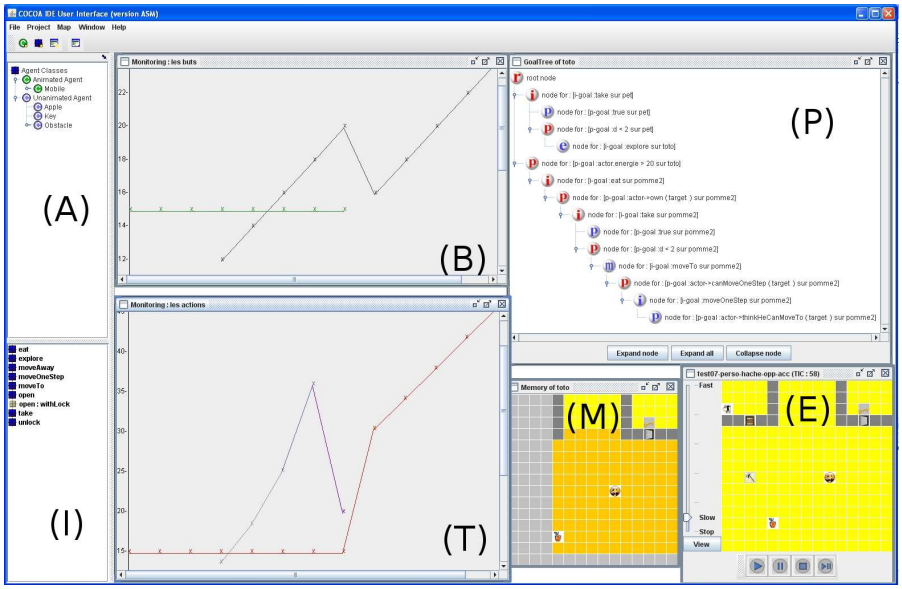

Figure 11. *L'interface de simulation de CoCoA permet de définir des interactions (I), des agents (A), des environnements (E) et de construire des simulations à l'aide de ces éléments. On peut alors observer l'environnement (E), la mémoire (M), le plan (P) et les évaluations des actions (T) et des buts (B) de chaque agent*

D'une expérience à l'autre nous n'utiliserons que l'une des motivations, celle que nous voulons mettre en évidence.

6.1.1. *Influence des buts et opportunisme*

Nous commençons par illustrer la motivation qui paraît certainement la plus naturelle : l'influence des buts. Pour des raisons qui apparaîtront immédiatement évidentes, nous couplons cette présentation avec celle de la motivation d'opportunisme. Dans les deux expériences ci-dessous seules ces deux motivations sont activées.

Dans cette expérience notre agent doit récupérer quatre objets : la pomme 1, la pomme 2, la clef et la hache, ayant chacun leur propre priorité de but : *prendre pomme1* $= 10$, *prendre pomme2* $= 9$, *prendre clef* $= 8$ et *prendre hache* $= 5$. La

hache n'apparait pas dans le champ de vision de l'agent au début de l'expérience, il doit donc *explorer* l'environnement pour la trouver. L'image de gauche de la figure 12 présente comment le plan est exécuté par le personnage si l'on ne prend en compte que l'influence des buts. On constate que les différents buts sont exécutés, et donc sélectionnés, dans l'ordre décroissant de leur importance pour l'agent. Cependant, la seule prise en compte de l'influence des buts se révèle clairement beaucoup trop naïve dans notre contexte situé dans lequel les déplacements doivent être pris en compte. Dans le cadre de simulation de comportements que l'on veut crédibles, la réalisation des déplacements est sans conteste un élément important de perception de cette crédibilité. C'est le cas en particulier pour les personnages de jeux vidéo. La séquence de déplacements générée ici peut amener l'observateur à désapprouver l'enchaînement des actions exécutées malgré la prise en compte des importances relatives des buts. C'est cet effet que la motivation d'opportunisme doit atténuer en favorisant les actions dont la cible est proche.

C'est ce que l'on observe dans l'image de droite de la figure 12. Il s'agit du même contexte mais nous avons ajouté à notre ASM la prise en compte de la motivation d'opportunisme. Par l'influence de l'opportunisme, l'exécution du plan (voir figure 14) change. Bien que le but *prendre pomme1* soit le plus important, la proximité de la *pomme2* (au pas de temps 0) amène l'agent à choisir ce but en premier. Ce choix est expliqué par la figure 13 qui présente les valeurs attribuées par l'ASM aux actions exécutables à chaque pas de temps. L'action *se déplacer vers pomme2* a une évaluation supérieure à l'action *se déplacer vers pomme1* grâce à l'opportunisme. Ensuite, alors qu'il se dirige vers la *pomme1*, l'agent est à nouveau temporairement amené à se détourner de son but principal (au pas de temps 5) pour prendre la *clef* dont il passe à proximité. Cette influence de l'opportunisme, qui oriente le comportement de l'agent malgré la priorité des buts, est à nouveau observable dans la figure 13 où, au pas de temps 5, la courbe de l'action *se déplacer vers clef* «passe au-dessus» de la courbe *se déplacer vers pomme1*, ce qui explique le choix d'action. Le compromis entre influence des buts et opportunisme a ainsi amené un enchaînement des actions plus «raisonnables». Notons que l'influence du but *prendre hache* n'était pas assez important pour que, même renforcé par l'opportunisme, cette action soit sélectionnée en début de simulation.

Signalons rapidement que le genre d'effet indésirable obtenu sans la prise en compte de l'opportunisme, est bien connu dans les jeux vidéo. Dans le fameux jeu *les Sims*, un personnage devant prendre le courrier dans la boîte aux lettres pour le déposer sur une table dans la maison et ramasser le journal au pied la boîte aux lettres pour le déposer à son tour sur la table fera des allers-retours de la boîte à la table plutôt que de ramasser le journal immédiatement lorsqu'il en est proche.

6.1.2. *Motivation des préférences*

Pour illustrer l'influence exercée par la motivation des préférences, nous allons mener deux expériences dans lesquelles nous ne changerons que les préférences attribuées aux actions *déverrouiller* et *casser*. Celles-ci sont présentées à la figure 15.

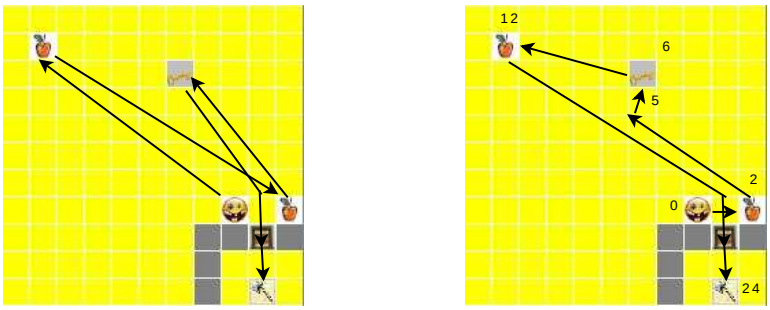

Figure 12. *L'agent a 4 buts, ayant chacun leur propre priorité :* prendre pomme1= 10, prendre pomme2= 9, prendre clef= 8 *et* prendre hache= 5. *La* pomme1 *est celle en haut à gauche. La version de droite possède les annotations de pas de temps que l'on retrouve dans la figure 13. A gauche, l'ASM prend en compte uniquement la priorité des buts, à droite, il prend en compte la priorité des buts et la motivation de l'opportunisme*

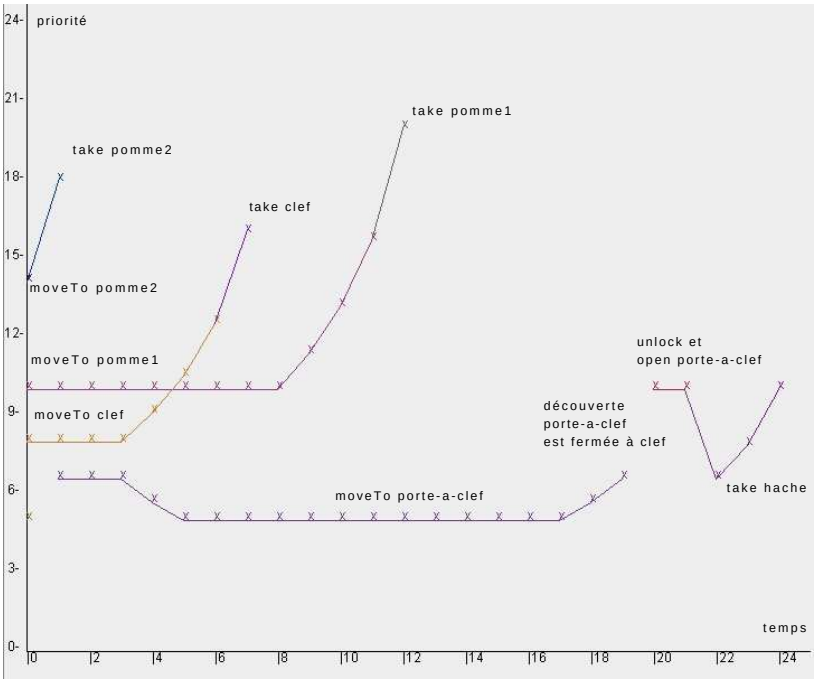

Figure 13. *Valeurs des actions exécutables calculées par l'ASM en fonction du temps, correspondant au déroulement de l'expérience de droite de la figure 12. Les valeurs prennent en compte la priorité du but associée et la valeur de l'opportunisme. Le pas de temps 0 correspond au temps avant de faire la première action. Les courbes correspondent aux évaluations accordées aux actions successives d'une même alternative*

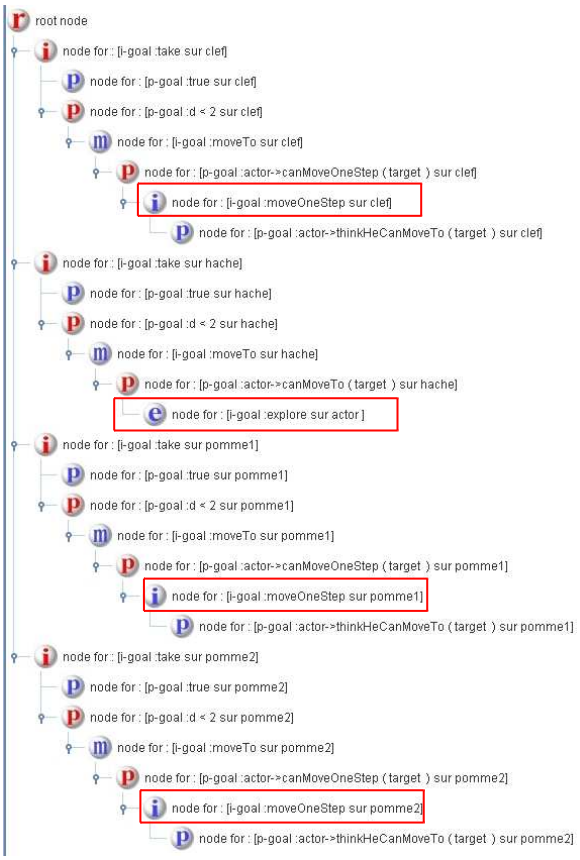

Figure 14. *Plan construit par le raisonnement, à l'aide d'un arbre et-ou, au début des deux simulations de la figure 12. Les actions exécutables sont encadrées en rouge. L'action explore s'explique par le fait que initialement l'agent ne sait pas où se trouve la hache*

A nouveau, un agent est situé dans un environnement (voir figure 15) dans lequel se trouve une porte, une pomme, une clef et une hache. Le but de l'agent est d'atteindre la pomme (pour la manger). La porte est fermée et verrouillée et il existe deux moyens de la franchir, soit en la déverrouillant avec la clef (puis en l'ouvrant) soit en la cassant avec la hache.

Dans la simulation de gauche, les valeurs de préférences attribuées aux capacités de l'agent sont : $\pi(\textit{déverrouiller}) = 2, \pi(casser) = 0,8$ et $\pi = 1,1$ pour les autres. Dans la simulation présentée dans l'image de droite, seules les préférences de *déverrouiller* et *casser* sont échangées : $\pi(\textit{déverrouiller}) = 0,8, \pi(casser) = 2$ et $\pi = 1,1$ pour les autres. On peut constater qu'à gauche l'agent privilégie l'action

prendre la clef pour ensuite déverrouiller la porte alors qu'à droite il privilégie l'action prendre la hache pour casser la porte.

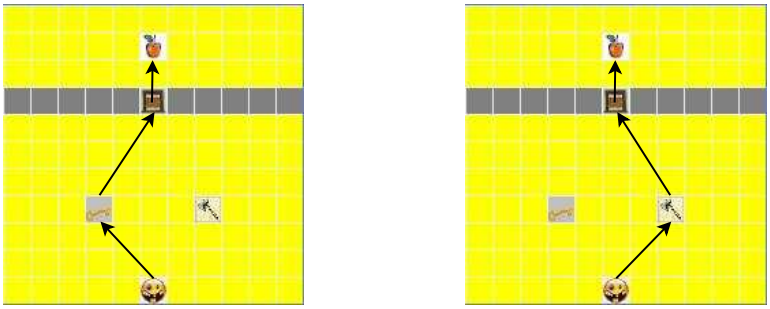

Figure 15. *Selon les préférences accordées à ses capacités, le personnage choisit de* déverrouiller *(à gauche) ou de* casser *(à droite) la porte.*

Expliquons rapidement ce qui justifie ces différences de sélection d'action. Le but de l'agent est *prendre(pomme)*, en fonction des capacités de l'agent, le chaînage arrière du planificateur produit pour ce but deux alternatives concurrentes, calculées à partir de l'arbre est/ou produit :

1) *prendre(pomme)* $\longrightarrow$ *ouvrir(porte)* $\longrightarrow$ *déverrouiller(porte)* $\longrightarrow$ *prendre(clef)*,

2) *prendre(pomme)* $\longrightarrow$ *casser(porte)* $\longrightarrow$ *prendre(hache)*.

La construction des plans prend en compte les déplacements en gérant les sous-plans qui en découlent et donc calcule que le franchissement de la porte est nécessaire à la résolution des buts. L'agent a deux manières de franchir cet obstacle : en le déverrouillant ou en le cassant (ce qui le supprime). Pour chacune la valeur de γ_{pref} s'exprime ainsi :

1) $\gamma_{pref}(prendre(clef)) = Moy_{harm}(\pi(prendre); \pi(ouvrir); \pi(\textit{déverrouiller}); \pi(prendre))$,

2) $\gamma_{pref}(prendre(hache)) = Moy_{harm}(\pi(prendre); \pi(casser); \pi(prendre))$.

Pour chacune des simulations on a donc :

à gauche $\gamma_{pref}(prendre(clef)) = Moy_{harm}(1,1;1,1;2;1,1) = 1,24$ et $\gamma_{pref}(prendre(hache)) = Moy_{harm}(1,1;0,8;1,1) = 0,98$, l'action retenue est donc *prendre(clef)*,

à droite $\gamma_{pref}(prendre(clef)) = Moy_{harm}(1,1;1,1;0,8;1,1) = 1,01$ et $\gamma_{pref}(prendre(hache)) = Moy_{harm}(1,1;2;1,1) = 1,29$, l'action retenue est donc *prendre(hache)*.

Dans les deux simulations, l'environnement, les capacités et les connaissances (l'état de sa mémoire) de l'agent étant rigoureusement les mêmes, les plans construits, et donc les alternatives proposées par le planificateur, sont identiques. La différence de déroulement des simulations est uniquement due à des choix différents réalisés par le mécanisme de sélection d'action. Dans ce cas, on peut percevoir l'influence de la

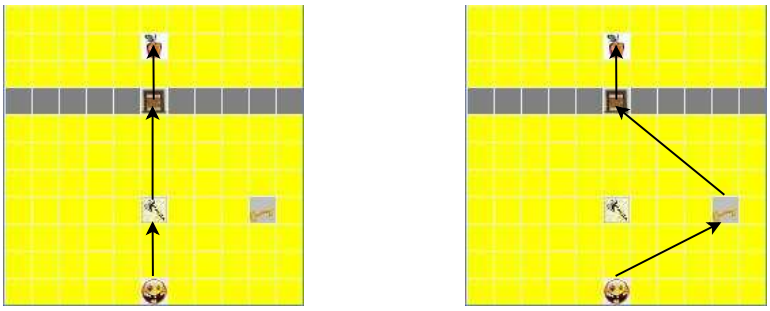

Figure 16. A gauche : *Influence de la motivation en espace : l'alternative avec le trajet le plus court est privilégiée.* A droite *: Influence de la motivation en temps, l'alternative de moindre coût est privilégiée*

motivation des préférences. Un observateur extérieur pourrait considérer l'agent de la simulation de droite comme «plus brutal» ou «moins discret» que celui de la simulation de gauche. Ainsi, en jouant sur les valeurs attribuées aux préférences des différentes capacités (c'est-à-dire son profil comportemental pour cette motivation) d'un agent, on influe sur la perception que l'on a de sa personnalité.

6.1.3. *L'achèvement en espace*

Cette fois, seule la motivation appelée *achèvement en espace* est utilisée dans un contexte de simulation similaire à celui utilisé dans la section 6.1.2. Cette motivation prend en compte les déplacements nécessaires à la réalisation d'une alternative. La simulation de gauche de la figure 16 illustre ce point : toutes les autres motivations sont écartées (notamment l'opportunisme), on constate que l'agent privilégie l'alternative nécessitant le moins de déplacement et donc ici l'action *prendre(hache)*.

6.1.4. *L'achèvement en temps*

La motivation de l'achèvement en temps a pour objectif de favoriser les alternatives dont les actions prennent le moins de temps (cumulé). Elle prend donc en compte les actions nécessaires à la réalisation d'une alternative. Nous expérimentons cette motivation dans le même contexte que l'achèvement en espace, mais seule cette motivation influence l'ASM (voir la figure 16 à droite). Nous avons, arbitrairement, attribué un coût de 5 à l'action *casser* et de 1 à toutes les autres actions. Le seuil θ_{achT} est fixé à 5. Si l'on reprend les alternatives présentées en section 6.1.2, on obtient :

1) $cost(resolution(prendre(clef))) = 1 + 1 + 1 + 1 = 4$ et donc $achS(prendre(clef)) = 1 + \log_5(\frac{5}{4}) = 1,14$

2) $cost(resolution(prendre(hache))) = 1 + 5 + 1 = 7$ et donc $achS(prendre(hache)) = 1 + \log_5(\frac{5}{7}) = 0,79$

L'action *prendre(clef)* est favorisée alors que *prendre(hache)* est pénalisée ce qui justifie le choix de prendre la clef retenu par l'ASM (à nouveau les autres motivations ont été désactivées, en particulier l'opportunisme et l'achèvement en espace).

Nous avons illustré l'influence exercée par la plupart des motivations que nous avons proposées. Dans ces expériences, le paramétrage est présenté de façon à illustrer l'influence des motivations sur la sélection d'action. Cependant il convient d'insister sur le fait que notre proposition n'a pas pour but de paramétrer les motivations afin de réaliser un scénario prédéfini, mais de permettre l'expression d'une individualité pour chaque agent dans tous les contextes. Une expérience de plus grande ampleur et exploitant la combinaison des motivations est présentée dans la section 6.2.

6.2. *Une expérience plus complète*

Dans cette expérience, nous considérons un agent-PNJ a caractérisé par un champ de vision et un attribut représentant son *énergie*. La valeur de son énergie décroît à chaque pas de temps. Les capacités de a sont de pouvoir se *déplacer*, *prendre* les objets, *manger*, *casser*, *déverrouiller*, *ouvrir* et *explorer*[1] l'environnement (voir tableau 3).

Pour définir la personnalité de l'agent, nous avons attribué arbitrairement pour chaque capacité une préférence afin de modéliser un agent qui manifeste un caractère brutal. Pour cela, nous avons donc affecté une valeur de $\pi(casser) = 1,5$ à l'action *casser* et pris $\pi(\textit{déverrouiller}) = 0,5$ (répulsion), les autres actions reçoivent une préférence neutre (ie. $\pi = 1$)[2]. Ces choix devraient mener a à préférer casser les portes plutôt que de les déverrouiller.

Une fois l'ASM construit, a est situé dans l'environnement, présenté à la figure 17. Les buts sont alors donnés à a. Dans notre cas, son premier but g_1 est de *prendre l'objet* o dans la pièce en haut à gauche fermée par une porte vitrée dont la clef est dans la pièce en haut à droite. Son deuxième but, g_2 est de maintenir son niveau d'*énergie au dessus d'une certaine valeur* v.

g_1 influence le comportement de l'agent par une fonction constante. L'influence de g_2 est fonction du niveau d'énergie de l'agent. Deux pommes et une hache sont présentes dans l'environnement, l'action de manger une pomme redonne de l'énergie à l'agent et la hache peut être utilisée pour casser les portes.

---

1. Comme nous l'avons présenté précédemment dans la section 3.1.4, *explorer* est l'interaction qui permet la recherche d'une cible inconnue.
2. La valeur $\pi(explorer) = 1$ n'est utilisée ici que pour mettre en évidence une revalorisation multibut. Si l'agent connaît l'emplacement de la clef et pas celui de la hache, il peut sembler plus rationnel qu'il utilise la clef plutôt qu'il parte à la recherche de la hache qui pourrait ne pas exister. Pour cela nous conseillons d'utiliser une valeur inférieure aux autres actions pour $\pi(explorer)$, car l'exploration doit rester le plus souvent la dernière solution à envisager.

ouvrir :
condition = target.opened = false
and target.locked = false
guard = distance(actor, target) < 1
action = target.opened = true

manger :
condition = actor.own(target)
guard = −
action = actor.energy.add(target.energy)
and remove(target)

prendre :
condition = −
guard = distance(actor, target) < 1
action = target.inventory.add(target)

casser :
condition = actor.own(axe)
guard = distance(actor, target) < 1
action = remove(target)

deverrouiller :
condition = target.locked = true
andactor.own(target.key)
guard = distance(actor, target) < 1
action = target.locked = true

explore et *se déplacer* sont des primitives
remove est une primitive qui supprime de l'environnement

Tableau 3. *Le code des interactions utilisées dans la simulation*

a n'a pas de connaissance *a priori* sur son environnement et il doit l'explorer. Les lieux inconnus apparaissent en noir dans la figure 17 où l'on peut également voir la portée de la vision de l'agent. A chaque pas, a exécute l'action sélectionnée par l'ASM dans le but de résoudre ses buts. Le trajet de l'agent est montré par des pointillés noirs.

Considérons le parcours de a. Au début, a est localisé au point **1**, il perçoit seulement la pomme à droite. Comme notre agent démarre par un niveau d'énergie supérieur à v, le seul but actif est g_1(ie. g_2 est satisfait, sa priorité est de $-\infty$). Comme a ne connaît pas son environnement, il doit l'explorer afin de trouver o. La figure 18 montre les différentes valeurs obtenues par les actions exécutables suite à l'évaluation de notre ASM. Au début, la seule action exécutable est *explorer*, de ce fait elle est sé-

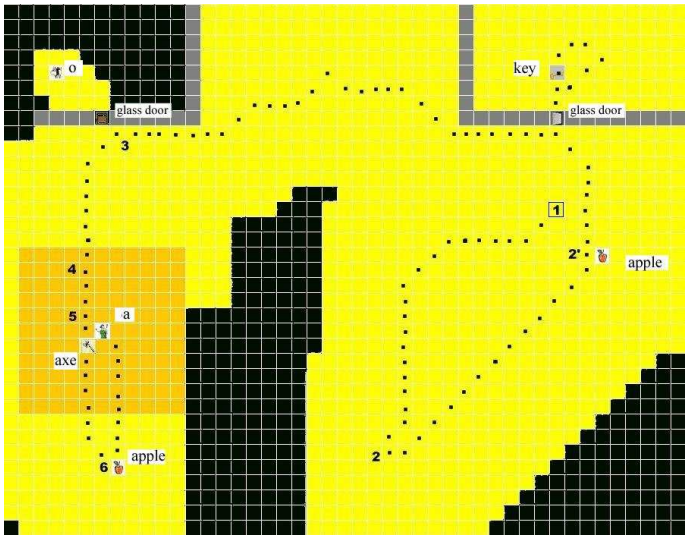

Figure 17. *l'environnement et le cheminement de l'agent dans celui-ci*

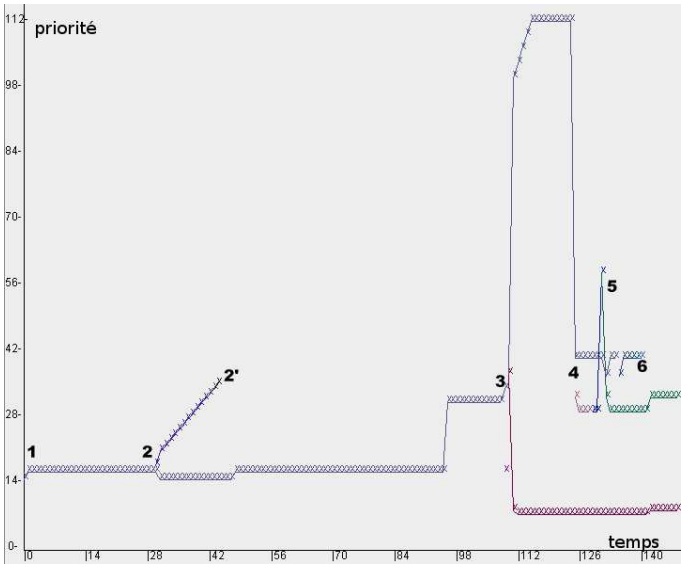

Figure 18. *Courbes d'évaluation des actions exécutables par l'ASM*

lectionnée. Pendant son exploration, a perd de l'énergie. En atteignant le point **2**, son énergie tombe en dessous de la valeur v, le but g_2 n'est plus satisfait, l'influence des buts note donc l'action résolvant g_2 selon la valeur actuelle de v. Cela implique que l'action de *se déplacer vers la pomme* (pour la manger) devient une action exécutable. Comme le montre la figure 18, la valeur de cette action est la meilleure, alors a choisit d'exécuter cette action. Comme son énergie continue à décroître pendant le déplacement, la priorité du but s'accroît et l'évaluation de l'action également. Au point **2'**, a mange la pomme, il reçoit l'énergie et le but g_2 redevient inactif. a reprend donc l'exploration afin de trouver o.

Atteignant le point **3**, a perçoit o, en essayant d'*ouvrir* la porte, a acquiert l'information que celle-ci est verrouillée. Le plan propose alors deux possibilités pour traverser la porte : la *déverrouiller* ou la *casser*. Deux actions exécutables apparaissent dans le graphe correspondant aux deux alternatives : la première, *se déplacer* vers la clef (qui a été vue précédemment), la seconde *explorer* pour trouver un objet pour casser la porte. Dans la mesure où casser a été favorisée, l'agent préfère casser plutôt que de déverrouiller la porte (d'autant plus que déverrouiller a été pénalisée $\pi = 0,5$). C'est pourquoi l'alternative de l'action *explorer* est favorisée par rapport à l'alternative de l'action *se déplacer* qui correspondant à la plus basse courbe démarrant en **3**. L'autre courbe est particulièrement haute, car par coïncidence, au même moment, le but g_2 est redevenu actif, et l'action *explorer* est également présente pour l'alternative concernant la recherche de nourriture. De ce fait, la *revalorisation multibut* a valorisé l'action *explorer* présente dans deux alternatives différentes.

En explorant, a se dirige vers «le bas» et perçoit une hache au point **4**. Alors, l'action exécutable pour *casser* n'est plus *explorer*, mais *prendre* la hache, cette dernière correspond à la nouvelle courbe au milieu. *Explorer* perd donc la faveur de l'évaluateur de *revalorisation multibut*, ce qui explique la chute de sa courbe. Néanmoins, elle reste l'action la plus prioritaire.

Au niveau du point **5**, par l'influence de l'*opportunisme* sur la cible hache, l'action exécutable *prendre la hache* est favorisée, devenant ainsi la plus prioritaire. Le pic au point **5** est dû à l'*opportunisme*, la baisse de l'action *explorer* au même moment est due à la perte de l'*inertie*.

Une fois la hache prise, l'influence de l'*opportunisme* disparaît et l'action *explorer* devient à nouveau la plus prioritaire. Plus tard, a trouve une pomme et la mange au point **6**. *Casser* la porte devient l'action sélectionnée par l'ASM. a se déplace vers la porte, la casse et prend o.

Cette expérimentation illustre le fonctionnement de l'ASM et la «compétition» entre les motivations. Les motivations *achèvement en temps* et *achèvement en espace* n'ont pas été mises en valeur dans cette expérience. D'autres expériences ont été implémentées et exécutées, avec d'autres environnements, d'autres agents et d'autres motivations. Toutes ces expériences ont montré des agents qui exprimaient les comportements correspondant aux traits désirés lors de la construction des motivations.

7. Conclusion

Concevoir le comportement des personnages non joueurs (PNJ), tels que ceux qui apparaissent dans un jeu vidéo de type jeu de rôles est un problème difficile tant du point de vue de la réalisation informatique que de la modélisation du comportement lui-même.

S'appuyant sur le moteur de comportement de CoCoA, la conception d'un personnage non joueur correspondant à notre proposition consiste d'une part à déterminer ses capacités (les actions qu'il peut effectuer) et, d'autre part, à choisir les paramètres des évaluateurs de motivation, tels que le seuil d'opportunisme θ_{opp}, les préférences des actions π, etc.

Ainsi l'agent s'appuiera sur son moteur comportemental, d'une part, pour construire le *raisonnement* permettant la résolution de ses buts et, d'autre part, pour choisir à chaque tour les actions à exécuter et ainsi établir son *individualité*. Notre proposition favorise la séparation du déclaratif et du procédural. En effet, l'approche centrée interaction permet une séparation des connaissances déclaratives et du processus de planification procédural.

Il en résulte que la mise en place des personnages et de leur comportement ne nécessite pas la programmation de code informatique. Le concepteur peut se concentrer sur le cœur de son problème : la réalisation de sa simulation. Pour cela il lui «suffit» de définir les interactions qui régissent son environnement et de régler les différents profils comportementaux souhaités en jouant sur les différents paramètres influant sur l'individualité des personnages. Certes ce problème n'est pas trivial mais il constitue la raison même d'une simulation, telle qu'un jeu.

Les perspectives à ce travail que nous envisageons sont dans un premier temps la mise en place effective de deux motivations relationnelles que nous étudions actuellement. Nous sommes convaincu que ces motivations peuvent permettre d'exprimer des comportements du type altruisme ou de prendre en compte la réputation d'autres agents.

L'altruisme. Cette motivation a pour objet d'exprimer de quelle manière l'agent est prédisposé à aider les autres agents.

La réputation. Cette motivation tient compte de la réputation des autres agents. Cette réputation peut être issue d'échanges sociaux précédents où d'une réputation sociale accordée aux autres agents. Ainsi un agent exécutera plus volontiers des buts (ordres) provenant d'un agent possédant une bonne réputation.

Ces motivations permettraient d'obtenir des comportements coopératifs, autres que ceux actuellement mis en place dans CoCoA. Pour le moment, les comportements coopératifs d'agents cognitifs dans CoCoA nécessitent l'utilisation d'une structure hiérarchique préalablement définie par la désignation d'un chef connu par tous les agents subordonnés (Devigne *et al.*, 2005). Or, la réputation des agents peut-être vue comme une relation hiérarchique implicite. Ainsi, l'altruisme et la réputation permettraient

une coopération sans pour autant avoir défini explicitement une structure hiérarchique entre les agents.

Dans un second temps, nous souhaitons mettre en place la notion de «familles de profil comportemental» permettant de typer plus facilement les comportements construits. Dans ce cadre la fusion de profils doit également être étudiée. Enfin, l'intégration de ces principes dans le cadre d'un jeu vidéo «réel» est évidemment considérée. Nous sommes, à ce jour, en négociation avec une entreprise de jeux pour gérer les PNJ qui peuplent le jeu avec notre moteur comportemental.

Remerciements

Ce travail est cofinancé par le contrat de plan état-région TAC du Nord-Pas de Calais et les fonds européens FEDER.

8. Bibliographie

Anderson J. R., Bothell D., Byrne M. D., Douglass S., Lebiere C., Qin Y., « An integrated theory of the mind », *Psychological Review*, vol. 111, n° 4, p. 1036-1060, 2004.

Andriamasinoro F., Proposition d'un modèle d'agents hybrides basé sur la motivation naturelle, PhD thesis, Université de la Réunion, 2003.

Arrow K., *Social choice and individual values*, J. Wiley, New York, 1951.

Blum A. L., Furst M. L., « Fast Planning Through Planning Graph Analysis », p. 1636-1642, 1995.

Blumberg B., « Action selection in Hamsterdam : Lessons from Ethology », *Proceedings of 3rd International Conference on the simulation of Adaptative behaviour*, MIT Press, p. 108-117, 1994.

Burloud A., *Principes d'une psychologie des tendances*, Librairie Félix Alcan, Paris (France), 1936.

Devigne D., Mathieu P., Routier J.-C., « Planning for Spatially Situated Agents », *Proceedings of IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'04)*, IEEE Press, p. 385-388, 2004.

Devigne D., Mathieu P., Routier J.-C., « Teams of cognitive agents with leader : how to let them some autonomy », *in* G. Kendall, S. Lucas (eds), *Proceedings of IEEE Symposium on Computational Intelligence Games (CIG'05)*, p. 256-262, 2005.

Devigne D., Mathieu P., Routier J.-C., *Intelligence Artificielle et Jeux*, Hermès, chapter Simulation de comportements centrée interactions, p. 183-210, 2007.

Ferber J., *Les systèmes multi-agents, vers une intelligence collective*, InterEditions, Paris (France), 1995.

Kubera Y., Mathieu P., Picault S., « Interaction-Oriented Agent Simulations : From Theory to Implementation », *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI'08)*, IOS Press, p. 383-387, 2008.

Laird J. E., « It knows what you're going to do : adding anticipation to a Quakebot », *in* J. P. Muller, E. Andre, S. Sen, C. Frasson (eds), *Proceedings of the Fifth International Conference on Autonomous Agents*, ACM Press, Montreal, Canada, p. 385-392, 2001.

Laird J. E., Newell A., Rosenbloom P. S., « SOAR : An Architecture for General Intelligence », *Artificial Intelligence*p. 1-64, 1987.

Laird J. E., van Lent M., « Human-level AI's Killer Application : Interactive Computer Games », *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence.*, AAAI, p. 1171 - 1178, 2000.

Maslow A., *Toward a Psychology of Being, 3rd Edition*, Lowry Richard, J.Wiley and Sons, 1998.

Mathieu P., Routier J.-C., Urro P., « Un modèle de simulation agent basé sur les interactions », *Actes des Premières Journées Francophones sur les Modèles Formels de l'Interaction (MFI'01)*, vol. 3, p. 407-418, 2001.

Müller J. P., Pischel M., « The agent architecture inteRRaP : Concept and application », 1993.

Nau D., Au T.-C., Ilghami O., Kuter U., Murdock W., Wu D., Yaman F., « SHOP2 : An HTN Planning System », *JAIR*p. 379-404, 2003.

Nau D. S., Cao Y., Lotem A., Munoz-Avila H., « SHOP : Simple Hierarchical Ordered Planner », *IJCAI'99 : Proceedings of the 16th International Joint Conference on Artificial Intelligence*, p. pp. 968-973, 1999.

Ponsen M., Spronck P., « Improving Adaptive Game AI with Evolutionary Learning », *in* S. N. Quasim Mehdi Norman Gough, D. Al-Dabass (eds), *Computer Games : Artificial Intelligence, Design and Education (CGAIDE 2004)*, University of Wolverhampton, p. 389-396, November, 2004.

Rosenblatt J. K., « DAMN : A Distributed Architecture for Mobile Navigation », *Journal of Experimental and Theoretical Artificial Intelligence*, AAAI Press, p. 339-360, 1997.

Schmidt B., « Human Factors in Complex Systems : The Modelling of Human Behaviour », *Simulation in wider Europe, 19th European Conference on Modelling and Simulation (ECMS 2005)*, p. 5-14, 2005.

Seth A. K., « Evolving Action Selection and Selective Attention Without Actions, Attention, or Selection », *the 5th International Conference on Simulation of Adaptive Behavior*, MIT Press, p. 139-147, 1998.

Sevin E. D., An Action Selection Architecture for Autonomous Virtual Humans in Persistent Worlds, PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2006.

Spronck P., Ponsen M., Sprinkhuizen-Kuyper I., Postma E., « Adaptive game AI with dynamic scripting », *Machine Learning*, vol. 63, n° 3, p. 217-248, 2006.

Tozour P., « The Perils of AI Scripting », *AI Game Programming Wisdom*, p. 541-547, 2002.

Tyrrell T., Computational Mechanisms for Action Selection, PhD thesis, University of Edinburgh, 1993.