KIOSK ASSISTANT (KAS): APPLICATION DE LA PLANIFICATION À UN ASSISTANT DE PRÉSENTATION MULTIMÉDIA

par

Kokou Mawuenyigan AHADllTSE

mémoire présenté au Département de mathématiques
et d'informatique en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES
UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, octobre 1999

Le d 7 odo/,e / fff: le jury suivant a accepté ce mémoire dans sa version finale.
date

Président-rapporteur: M. ShengruiWang
Département de mathématiques et d' inffiÎiflatique

Membre: M. Roger N'Kambou
Département de mathématiques et d'informatique

Membre: M. Froduald Kabanza
Département de mathématiques et d' m:fôrmatiqut..

Sonimaire

Ce travail s'inscrit dans le cadre de mon programme de maîtrise en Génie Logiciel à l'Université de Sherbrooke. Il consiste à animer un agent et à diriger ses mouvements par un planificateur afin de réaliser une présentation multimédia. La présentation est appliquée au Département de Mathématiques et d'Informatique de l'Université de Sherbrooke (D.M.I.) et porte sur les différents domaines d'études, ainsi que les professeurs qui effectuent des recherches dans ces domaines. Nous mettrons ainsi en évidence le gain en temps et la convivialité pour s'acquérir des informations au D.M.I..

La facilité de production des médias due à l'évolution et au développement technologiques ainsi que la limitation des outils et applications impliquant des interactions homme-machine ont engendré de nouvelles manières de communiquer ou de présenter l'information. Ainsi, des systèmes multimédia deviennent de plus en plus nombreux sur le marché et présentent beaucoup d'utilitaires pour communiquer avec des utilisateurs soit par le son, soit par des images ou par la voix. Mais beaucoup de ces systèmes restent encore implémentés à partir de fonctions simples et ne répondent pas aux besoins des utilisateurs.

Par ailleurs, la progression des travaux en Intelligence Artificielle a amené plusieurs à considérer le fait que des applications interactives intelligentes (agents logiciels) peuvent aider l'homme à résoudre certains de ses problèmes jusque là difficiles à résoudre.

Mais la capacité de construire des applications plus adéquates pose d'autres problèmes, parmi lesquels, celui du nombre gigantesque de cas à considérer en une seule

opération. Pour s'attaquer à ce problème, la technique souvent utilisée est la planification qui représente un domaine très actif de l'intelligence Artificielle (IA).

Le problème auquel nous nous attaquons à travers ce travail est de nous servir des particularités du planificateur SimPlan pour améliorer les systèmes de présentation multimédia existants.

Renierciements

Ce travail n'aurait *pas* pu être mené à terme sans l'aide et l'encouragement de nombreuses personnes qui m'ont toujours entouré et soutenu. J'aimerais d'abord remercier mon directeur de mémoire, le Professeur Froduald Kabanza qui a toujours été disponible pour ses étudiants et qui m'a donné goût à l'intelligence artificielle.

J'aimerais par ailleurs dire merci à tous mes amis et connaissances qui m'ont aidé à travers ce programme de maîtrise et qui m'ont été d'un grand recours à l'Université de Sherbrooke.

Mes parents ont grandement contribué à ce succès par le sens du devoir qu'ils ont su m'inculquer depuis l'enfance, ainsi que leur encouragement constant .

Enfin, j'aimerais remercier particulièrement ma femme, Kpodar Kayi qui a toujours été à mes côtés dans les moments difficiles pour me montrer les chemins à suivre.

Table des matières

Liste des figures

Introduction

Avec la nécessité de faire évoluer et de rendre plus convivial la communication entre les hommes, plusieurs techniques ont été développées tant sur le plan didacticiel que sur l'aspect rhétorique, principalement dans le domaine des présentations à base de l'ordinateur. Des technologies nouvelles ont vu le jour pour mettre l'utilisateur dans un environnement qui simule au maximum son monde réel, permettant ainsi la réalisation des présentations effectives et dynamiques à partir de plusieurs médias. La conception de ces présentations doit prendre en compte les considérations temporelles surtout au niveau de la coordination des sorties et du présentateur.

Par ailleurs, l'idée de pouvoir réaliser des applications qui réagissent comme l'homme a amené les intervenants dans le domaine à adopter la technologie des agents, qui représente d'ailleurs une problématique sur laquelle l'intelligence artificielle (IA) s'est penchée depuis longtemps [4] . Cette technologie a aujourd'hui dépassé les bornes de la recherche et s'est étendue dans les industries et dans les compagnies commerciales.

Mais quoique chacun tienne à sa définition du terme [7], nous pouvons dégager une conception plus ou moins standard qui puisse nous amener vers une première tentative de définition. Pour l'instant, un agent peut généralement être défini comme une entité ou un programme informatique qui communique avec son environnement par des capteurs et des effecteurs. L'agent perçoit son environnement par des capteurs et agit sur cet environnement par des effecteurs. Nous reviendrons sur une àéfinition plus complète d'un agent plus tard, mais pour l'instant voyons d'abord quelques exemples.

Un exemple d'agent est le Personal DJ de Timothy Shih [14}. C'est une application qui laisse l'utilisateur choisir rapidement des chansons par classification. L'interface présente un écran où l'utilisateur doit choisir ses préférences telles la période des chansons (moderne, classique, etc.), le style (symphonie, concert, rock, jazz, blues, etc.), la nationalité du chanteur, l'occasion (mariage, festivals, etc.) ou le tempéremment (joyeux, romantique, etc.) .

L'usager doit aussi répondre à certaines questions telles "Avez-vous besoin d'une explication de la musique ?" afin de décider du niveau de description nécessaire. D'autres options sont aussi disponibles pour que l'utilisateur choisisse s'il veut avoir une visualisation lorsque la musique joue. Persona! DJ est capable d'apprendre en demandant à l'utilisateur de sélectionner des mots clés au cours de la performance.

Un autre exemple d'agent est le PPP Persona [2J du Centre allemand de recherches en intelligence artificielle (DFKI). PPP Persona est un agent à caractère humain qui présente, explique, montre et commente verbalement à l'aide de graphiques, d'images et de texte.

Avec la croissance de la popularité des outils multimédia, les systèmes multimédia tels la réalité virtuelle et les jeux, les logiciels de présentation deviennent de plus en plus nombreux: sur le marché.

De nos jours, beaucoup d'applications de présentation multimédia sont basées sur des outils auteurs qui permettent de concevoir des documents hypermédia. A la différence des anciens systèmes de présentation, on assiste aujourd'hui à des applications de présentation qui présentent des boutons et qui laissent l'utilisateur naviguer à travers différents sujets. Les outils auteurs englobent la capacité de sélectionner, de générer et de structurer le contenu de l'information à présenter. Ils permettent par ailleurs de sélectionner le média convenable à l'information et permettent une coordination spatiale et temporelle de la présentation.

Les systèmes de présentation multimédia traditionnels étant limités dans leur capacité de prendre en compte efficacement un grand nombre de situations ou d'états dans le processus de décision, le besoin de penser les choses autrement s'est imposé. La planification longtemps orientée vers la robotique peut apporter beaucoup d'amélioration à ces systèmes, notamment avec sa capacité de produire rapidement des séquences d'actions différentes au cours d'une présentation.

Considérant un robot qui peut effectuer des actions atomiques, la planification lui fournit des séquences d'action qui lui permettent d'accomplir des tâches de haut niveau tel qu'éviter des obstacles dans un couloir.

La planification s'attaque à des problèmes de complexité due à l'explosion d'états au niveau des séquences d'actions qui peuvent être disponibles dans un domaine donné. Même dans un environnement simple composé de quelques locaux seulement, le nombre de déplacements possibles pour un agent présentateur est considérable. Par exemple, imaginons les mouvements d'un agent qui peut aller vers l'avant, reculer , tourner vers la gauche, tourner vers la droite ou encore faire d'autres types de mouvements possibles. Imaginons aussi la présence des obstacles dans cet environnement, et que l'agent doit se déplacer en effectuant des tâches concises. Nous pouvons remarquer qu'il n'est pas aisé de mettre au point un programme qui puisse facilement permettre à notre agent de trouver le meilleur chemin parmi la multitude qui existe. Pour ce faire, il va falloir trouver des techniques sophistiquées qui permettront de représenter de manière efficace l'espace d'états à explorer et de contrôler cette exploration.

L'application de la planification aux présentations varie suivant la nature du problème à résoudre. Il peut s'agir de la présentation d'une chaîne de montage de véhicules, d'un assemblage de jouets ou encore de l'utilisation d'un appareil photographique. Dans ce dernier cas, nous pouvons citer le système de "Planification des présentations multimédias" ou PPM [3] qui utilise le même système de planification que celui utilisé dans notre travail.

Le travail que nous allons présenter consiste à implanter un agent par un personnage animé, et à diriger ses mouvements par un planificateur afin de réaliser une présentation multimédia du Département de Mathématiques et d'Informatique de l'Université de Sherbrooke.

Cette présentation porte sur les différents domaines d'études, ainsi que les professeurs qui effectuent des recherches dans ces domaines. Le projet peut être vu comme un exemple d'optimisation des méthodes existantes pour s'acquérir des informations au sujet de D.M.I. Pour l'instant, pour s'informer sur le D.M.I., il faut soit téléphoner et avoir un employé au bout de la ligne, soit chercher les documents relatifs au domaine choisi, soit accéder au site Internet du D.M.I. et commencer à naviguer. Les moyens existants permettent bien sûr d'arriver à la finalité, c'est à dire obtenir les informations désirées. Cependant le temps de recherche peut être parfois très long. D'autre part, l'employé du D.M.I. au bout de la ligne peut être pris à faire autre chose au moment où l'on a besoin des informations.

L'outil auteur que nous utilisons est *Director* [1]. Les séquences d'action de notre agent sont générées par le planificateur ***SimPlan*** [2] qui utilise des prédicats pour exprimer son evironnement et des contraintes temporelles pour contrôler ses espaces de recherche.

Ce travail est divisé en plusieurs chapîtres qui seront présentés successivent dans ce document et qui sont identifiés comme suit.

- Les agents
- Les systèmes de présentation multimédia
- Director
- La planification

[1] Director est un produit de la compagnie Macromédia

[2] SimPlan est un planificateur développé par Froduald Kabanza. Toutes les informations se trouvent sur le site web http://www.dmi.usherb.ca/kabanza/~simplan.

- **La présentation du Kiosk Assitant.**

Chapter 1

Les agents

1.1 Conceptualisation

Sujet de recherche depuis quelques décénies en intelligence artificielle, le concept des agents est devenu dernièrement un point de mire pour un grand nombre d'intervenants en informatique. Dans l'industrie comme dans les compagnies informatiques, sur le World Wide Web, on parle d'agents. Plusieurs termes sont utilisés pour désigner le concept ou ses variétés: agents intelligents, agents logiciels, systèmes multi-agents, etc.

Ne voulant pas s'éloigner de leur habitude, les informaticiens n'ont pas mis du temps pour attribuer des définitions ou plutôt de tentatives de définition souvent divergentes au concept.

Loin d'avoir une définition unanime sur ce que peut être un agent, nous arrivons à noter quelques notions qui reviennent presque toutes les fois chez les différents intervenants.

Une définition très simpliste du concept d'agent serait un outil ou un programme informatique qui réalise une tâche similaire à celle que peut réaliser un humain. Par exemple, on peut citer l'agent "PPP Persona" de DFKI [1], un agent à caractère humain qui synthétise du matériel multimédia et planifie une présentation.

1.2 Évolution du concept d'agent dans le temps

Le concept d'agents remonte au début des travaux en Intelligence Artificielle (IA) avec une notion de modèle primaire d'objet interactif qui possède des capacités intrinsèques qu'il est convenu de nommer acteur. Cet objet encapsule un état interne et peut répondre aux messages provenants d'autres objets. Par la suite, la programmation distribuée et parallèle a amené un changement dans les motivations de recherche au sein de la communauté de l'IA.

Les premiers travaux sur les agents (1977 - 1990) étaient essentiellement concentrés sur les agents qui possèdent un modèle explicite et symbolique du monde, et qui raisonnent de manière symbolique. Les préoccupations tournaient autour de l'interaction et la communication entre les agents, de la décomposition et la distribution des tâches, de la coordination et la coopération entre les agents, et de la résolution des conflits entre agents par la négociation.

La suite des premiers travaux sur les agents (1990 - à nos jours) est caractérisée par deux directions distinctes : d'une part les travaux sont effectués dans la recherche et le développement sur la théorie, l'architecture et les langages des agents, et d'autre part sur les différents types d'agents qui peuvent exister.

1.3 Caractéristiques principales d'un agent

Nous avions précedemment défini un agent comme étant un outil ou un programme informatique qui réalise une tâche plus ou moins similaire à celle que peut réaliser un humain. Mais d'aucun diraient qu'il y a beaucoup de programmes qui réalisent des tâches similaires à celles réalisées par les humains et la question qui se pose est de savoir si tous œs programmes sont des agents. Nous pouvons citer par exemple un programme de gestion de réseau qui fait de la surveillance et qui notifie les utilisateurs quand il le faut.

De là vient sans doute la nécessité de faire la différence entre un agent et un programme informatique simple.

D'un point de vue général, un agent peut être défini comme une entité autonome et proactive qui peut percevoir son environnement et qui est capable de prendre des décisions rationnelles pour agir sur celui-ci. La prise de décision est fortement liée au raisonnement et c'est sur cet aspect qu'intervient la proactivité de l'agent. Cette définition peut être plus complexe ou plus explicite du point de vue des caractéristiques qui vont être implémentées au sein de l'agent.

Les informations perçues par l'agent sont transformées en séquences d'actions qui vont lui permettre d'arriver à un but.

L'agent perçoit son environnement par ce qu'on appelle communément un senseur, et il se sert d'un effecteur pour agir sur son environnement. Dans le cas de notre agent KAS (Kiosk ASsistant), il existe des variables qui lui permettent d'avoir des messages provenant de l'environnement Director, et il agit en tenant compte de ces messages. Il peut par exemple recevoir un message qui lui indique de revenir à sa position initiale. De l'autre côté, KAS va effectuer des actions qui vont changer l'environnement en provoquant par exemple des changements sur la scène.

La proactivité de l'agent relève de ses capacités à prendre des décisions. Il existe aujourd'hui une pléthore d'agents pour accomplir des tâches différentes.

La notion d'agent couvre généralement les agents matériels (robots) et les agents logiciels (KAS par exemple). Pour notre travail, nous allons nous intéresser rien qu'aux agents logiciels.

Certains agents sont mis au point pour réaliser des tâches spécifiques, telles que la présentation d'un environnement physique dans le cas de KAS, et des pièces détachées d'une auto par exemple ; d'autres servent d'outils d'amusements. Un autre exemple est Julia[9], un agent mis au point par Michael Mauldin de l'Université de Carnegie-Mellon (CMU). Julia est un agent qui peut faire plusieurs choses : jouer, chanter, etc.... Mais il

est surtout programmé pour passer le "test de Turing".

Il existe plusieurs efforts pour définir le terme agent; Nwana et Ndumu [10] se basent sur les propriétés pour classifier les agents, tandis que pour Charles Petrie[11], la notion d'agent est fortement liée à la communication.

Les principales propriétés que nous retrouvons chez les agents sont :

- L'autonomie.
- L'interaction avec l'environnement et avec d'autres agents éventuels.
- La réaction à l'environnement.
- La prise d'initiative.
- La rationalité.

Il faut noter qu'un agent peut ne pas regrouper toutes ces propriétés mais peut quand-même rendre ces actions conformes à la philosophie générale des agents. Dans la pratique, un agent doit démontrer de la rationalité qui va l'amener à maximiser à chaque fois ses performances pour chaque séquence d'actions effectuées. Cette maximisation des performances doit tenir compte du résultat de la perception ainsi que des dispositions de connaissances acquises ou intégrées à l'agent. La rationalité doit permettre de dire jusqu'à quel degré l'agent a satisfait à un but qui lui a été assigné; généralement, des mesures sont prises par exemple sur l'environnemnt ou sur les ressources pour juger du degré de satisfaction d'un but. Il est important qu'un utilisateur remarque l'avantage que KAS peut procurer par rapport aux systèmes traditionnels.

Pour décrire un agent dans ce document, nous allons nous baser sur les critères de Franklin et Graesser[7] qui énumèrent les points suivants :

- L'environnement. L'environnement est très important dans la description d'un agent. On peut dire que c'est le monde extérieur avec lequel agit directement

l'agent. L'environnement est un système dynamique qui change dans le temps par rapport à tous les états possibles menant vers le but de l'agent. Dans le cas de KAS, l'environnement est constitué par les événements du système auteur Director.

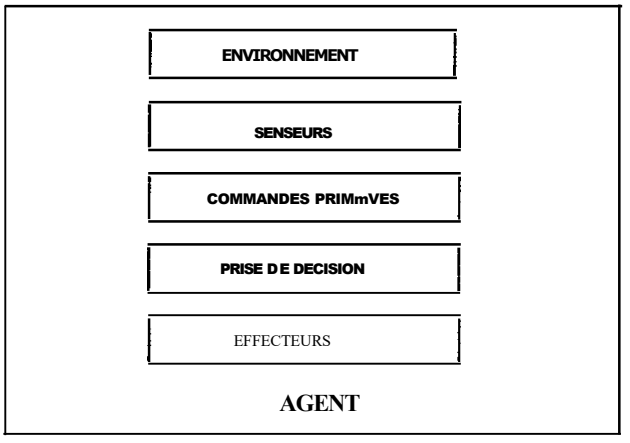
- La capacité de percevoir. La perception de l'environnement dans lequel l'agent se trouve est réalisée par les senseurs qui retournent à l'agent des informations sur l'environnement. C'est à partir de ces informations que l'agent décide de l'action à prendre. Pour ce qui concerne KAS, la perception est faite par des messages reçus à partir des variables d'état qui lui indiquent ce qu'un utilisateur désire observer dans la présentation ou l'état du système par exemple. La perception peut être passive ou active. Lorsque l'agent participe au mécanisme de perception, par exemple en envoyant des commandes, la perception est dite active. Par contre si aucune action n'est requise de la part de l'agent dans ce mécanisme, la perception est dite passive.

- Les actions. L'agent, en réaction aux perceptions de l'environnement agit sur ce dernier pour pouvoir s'approcher de son but final. L'action de l'agent produit un changement de son état courant et celui de l'environnement. Provoquer des changements sur la scène de Director comme résultante d'une initiative est un exemple palpable d'une action que peut effectuer KAS. Les actions peuvent aussi être combinées en séquences d'actions et produire une action d'un niveau plus élevé.

- Les prédispositions primitives (commandes de motivation). Ce sont des fonctionnalités primitives de l'agent; elles servent à bâtir des actions plus avancées.

- L'architecture de sélection d'une action. Le mécanisme de prise de décision est très important chez un agent. On pourrait dire que c'est ce qui différencie un agent d'un programme simple. L'agent doit être capable de prendre lui-même des initiatives pour décider de ce qu'il faut faire dans un état prochain, compte tenu du but visé. Cette architecture est représentative de notre cas et les modules se

retrouvent aisémemt au sein de KAS.



Ftigure 1: Architecture générale d'un agent

1.4 Quelques types d'agents

La classification des ai.gents dépend essentiellement des propriétés sur lesquelles on se base. Ainsi, les classes cfl.'agents les plus évidentes sont énumérées ci-dessous.

1. **Les agents réactifs.** Ce sont des agents qui ne possèdent pas un modèle symbolique interne de leurr environnement. Ils réagissent à la perception de l'environnement dans lequel ils se t;rouvent et KAS fait partie de cette catégorie. Il existe trois notions principales reliées aux agents réactifs qui sont les suivantes.

   - La fonctionnai.Lité d'emmergence. Les agents réactifs sont relativement simples et interagisse:·nt avec d'autres agents. De cette interaction peuvent émerger des comporte:·ments plus complexes.
   - La décomposition de tâches. L'agent réactif est vu comme un ensemble de modules qui •opèrent de manière autonome et qui accomplissent des tâches particulières = contrôle, perception, etc...

- Les agents réactifs peuvent opérer sur des représentations de bas niveau.

Un exemple d'agent réactif est le Persona! DJ [14]. C'est une applica.tion qui laisse l'usager choisir rapidement des chansons par classification. L'interface présente un écran où l'utilisateur doit choisir ses préférences telles la période des chansons (moderne, classique, etc.), le style (symphonie, concert, rock, jazz, blues, etc.), la nationalité du chanteur, l'occasion (mariage, festivals, etc.) ou le tempéremment (joyeux, romantique, etc..). L'usager doit aussi répondre à certaines questions telles "Avez-vous besoin d'une explication de la musique?" afin de décider du niveau de description nécessaire. D'autres options sont aussi disponibles pour que l'utilisateur choisisse s'il veut avoir une visualisation lorsque la musique joue. Persona! DJ est capable d'apprendre en demandant à l'utilisateur de sélectionner des mots clés au cours de la performance. Chaque présentation comprend quatre parties : la musique (fichiers .WAV), les images (fichiers bitmap), le texte actif et la connaissance.

2. **Les agents mobiles.** Ils ont la particularité de se déployer sur des sites hôtes à travers un réseau, tel le World Wide Web, et performent des tâches en interagissant avec d'autres hôtes. Ils sont autonomes et possèdent la capacité de coopérer et de communiquer avec d'autres agents. Dans le développement des agents mobiles, il existe certains points déterminants qui constitutent la base de tout travail.

   - Le déplacement. Ce point est relatif à la migration de l'agent d'un site à un autre.
   - L'authentification. L'authentification permet de s'assurer que l'on a les renseignements précis sur la nature de l'agent.
   - La sécurité. C'est l'habileté à protéger les données ou informations dont l'agent dispose.

- La discrétion. C'est la capacité de l'agent de conserver le caractère privé de l'usager par rapport à tout son environnement, ainsi que la protection contre l'accès des informations privées.

3. **Les agents d'information.** Ils manipulent et collectent des informations venant de plusieurs sources distribuées. Les intérêts suscités par ces agents deviennent de plus en plus grandissants vu l'explosion des sources de données disponibles aujourd'hui, surtout sur Internet (le World Wide Web en particulier). Ils sont en général supportés par les fureteurs sur Internet, tels Infoseek et Yahoo. d'agent d'information est le "softbot" de Oren Etzioni[5]. "softbot" est un assistant personnalisé plus ou moins intelligent pour l'accès à Internet. Il est construit sur la base des techniques de planification et d'apprentissage. Il prend les requêtes de l'usager dans un langage évolué proche du langage naturel et génère des plans qu'il exécute pour satisfaire au but donné par l'usager. Il est aussi doté d'une capacité d'apprentissage. En outre, "softbot" a la capacité de clarifier la requête de l'usager si celle-ci est floue ou mal spécifiée, et agit par exploitation de l'inférence et de la connaissance pour satisfaire aux différents buts donnés par l'usager.

   Sur le plan de la génération des séquences d'actions, nous pouvons faire un rapprochement entre "softbot" et KAS du fait que dans les deux systèmes, les séquences d'actions sont générées par un planificateur (XII (eXecution and Incomplete Information)) dans le cas de "softbot" et SimPlan dans le cas de KAS.

4. **Les agents de caractères.** Ils sont capables de démontrer beaucoup de caractères (états émotionnels par exemple) comme chez un humain. Un exemple d'agent dans cette catégorie est le "PPP Persona" [2]. "PPP Persona" est un agent qui performe une présentation avec des démonstrations, des explications, et des commentaires verbaux basés sur des textes et des graphiques.

C'est un système client/serveur. Le client envoie des requêtes au module serveur de "PPP Persona", en se basant sur l'état courant du présentateur. Une confirmation est envoyée au client après satisfaction de la requête.

L'architecture à un niveau d'abstraction élevé est composée du module de génération multimédia, du planificateur de présentation et du module de présentation multimédia. Le module de génération multimédia comme son nom l'indique se charge de générer le matériel multimédia qui va servir dans la présentation. Il reçoit des actes de production comme entrée (lui indiquant quel matériel produire) et, en retour, envoie des détails sur le matériel produit au planificateur de présentation. Le planificateur de présentation envoie des actes de présentation au module de présentation et reçoit de celui-ci des signaux: et des événements.

Dans ce chapitre, nous avons situé le niveau conceptuel des agents logiciels et nous nous sommes attachés aux principales caractéristiques inhérentes aux agents en général. Nous avons vu que des principes tels que l'autonomie, la proactivité et la rationalité doivent être observés dans le comportement des agents. Nous avons aussi introduit quelques types d'agents, entre autres, les agents mobiles, les agents d'information, les agents de caractères et les agents réactifs qui nous intéressent particulièrement dans ce travail. Étant donné que notre travail porte sur un agent de présentation multimédia, nous présentons dans le chapitre prochain, les principes généraux de conception et de mise en oeuvre des systèmes de présentation multimédia en général.

Chapter 2

Les systèmes de présentation multimédia

Le développement des technologies de l'information a permis de constater plusieurs nouvelles manières d'appréhender la communication de l'information. On remarque beaucoup plus de qualité, de finesse, de réalité et beaucoup plus d'efficacité dans le traitement de l'information à communiquer ou à présenter. Cette efficacité se manifeste par une rapidité de présentation de l'information et une grande capacité de stockage. De plus, il est plus facile de produire du matériel vidéo et audio de grande qualité, ce qui entraine d'autres façons de concevoir la présentation de l'information.

Parallèlement, les exigences des utilisateurs face à cette dynamique de progrès grandissante sont devenues prioritaires surtout dans la présentation de l'information. On veut que l'information soit la plus réelle et la plus effective possible. Les systèmes existants sont ainsi appelés à satisfaire à un besoin d'automatisme et de facilité d'utilisation.

Les systèmes de présentation multimédia comme celui que nous allons implanter dans ce travail se basent sur des présentations pour communiquer des informations à un utilisateur. À partir des médias et de leur combinaison par exemple, nous pouvons communiquer à un utilisateur la manière de choisir un domaine de recherche en mathématiques ou en

informatique à l'Université de Sherbrooke. Ces systèmes orchestrent la mise au point d'une présentation qui délivre de l'information en utilisant des techniques de présentation et des médias, et satisfont l'utilisateur sur le plan communicationnel et dans la satisfaction de ses buts.

2.1 Caractéristiques des systèmes de présentation multimédia

Notre système d'agent doit posséder certaines caractéristiques qui se retrouvent d'ailleurs à la base de tout système de ce genre. Ces caractéristiques sont formalisées par Steven Roth et William E. Hefley [13], dans leur litérature générale sur le sujet et discutées dans la section suivante.

- La détermination du contenu de l'information appropriée à communiquer. Ce n'est pas toujours facile de sélectionner la bonne information à présenter. Ceci résulte d'un mécanisme qui peut parfois être complexe dû à la diversité des sources d'information et à la coordination des éléments provenant des médias. Quels sont les points les plus attrayants pour un utilisateur quand il s'agit d'entreprendre des études de recherches ? Quels sont les grands succès du département de mathématiques et d'informatique de l'Université de Sherbrooke ? Ce sont là des questions qui doivent être répondues à travers les capacités dont KAS va être dotées.

- Représentation de l'objectif communicationnel et des tâches. Ici intervient la notion d'utilité de conception et d'expression de ce que le programmeur va communiquer à l'utilisateur. Ceci peut vaguement dépendre du contexte. Pour répondre par exemple dans notre cas à la question de savoir si les sujets de recherches doivent être représentés par du texte, par une image graphique ou encore par une animation défilante, une analyse minutieuse doit être faite à ce sujet.

- La capacité de pouvoir sélectionner et assembler les médias appropriés. Nous notons ici la souplesse du système présentateur à assembler au moment opportun des bouts de média afin de satisfaire l'usager.

- La coordination de différents médias. La combinaison de médias multiples (images, voix, vidéo, graphique, etc..) permet d'accroître l'efficacité de la présentation [6]. Mais la coordination a toujours été un des aspects difficiles à gérer. Comment faire arrimer des fichiers de son à la cadence d'une image défilante pour simuler par exemple le bruit des pas de notre agent KAS ? Comment contrôler les pauses dans le défilement de manière à ne pas interrompre la continuité du son ou de l'image ? Dans l'enchainement de l'action tourner à gauche et de l'action marcher dans le couloir, par exemple, il faut assurer la continuité dans la démarche de notre agent ainsi qu'une continuité dans le décor. Ce sont là quelques cas simples mais qui peuvent déjà montrer la complexité du sujet. Ceci paraît d'autant plus vrai qu'il faut s'attendre à traiter des médias de sources très variées.

- Développement des techniques pour l'interaction avec l'usager. Dans la plupart des systèmes de présentation, moins l'usager interagit avec le système tout en jouissant de sa satisfaction, plus le système est attrayant. La plupart des tâches primitives peuvent être automatisées, évitant ainsi à l'usager de recourir à chaque étape de la présentation à une interaction avec le système. Il faut rappeler ici que l'usager s'attend à voir une présentation et non à assumer certaines tâches du système qui présente.

Un système de présentation multimédia comme celui décrit à travers œ document doit au maximum automatiser les tâches à effectuer et trouver un moyen adéquat pour soutenir les combinaisons possibles de médias qui vont être utilisés au cours de la présentation.

2.2 Architecture des systèmes de présentation multimédia

La figure suivante montre l'architecture générale des systèmes de présentation multimédia.

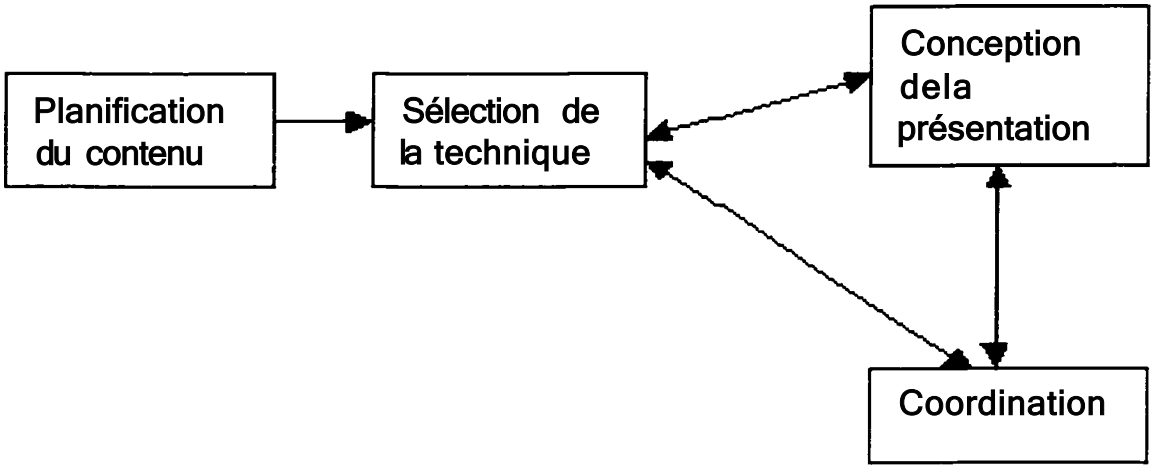


Figure 2: Architecture générale des systèmes de présentation multimédia

Les systèmes de présentation multimédia ont constitué une révolution dans la manière de présenter l'information en associant des médias et des données pour satisfaire aux usagers. Nous avons montré les grands principes des systèmes de présentation multimédia dont la détermination du contenu approprié à communiquer, la bonne représentation de l'objectif communicationnel, la coordination des médias et l'interaction avec l'utilisateur. Ces deux dernières caractéristiques ne sont rendues possibles que par l'aide des environnements logiciels spécialisés appelés systèmes auteurs. Dans le chapitre suivant nous

allons présenter le système auteur Director qui nous a servi d'environnement de réalisation de KAS.

Chapter 3

Director

La communication et l'interaction jouent un rôle très important dans notre société moderne. Beaucoup d'efforts ont été investis pour mettre au point des éléments de communication très performants, ce qui a profondément changé le côté social et communicationnel de notre quotidien. Notre société est devenue une société d'information marquée par un changement perpétuel dû au développement technologique. Les systèmes auteurs (exemple Director) occupent une place très importante dans cette transformation en permettant de créer des applications multimédias interactives, des films contenant des éléments graphiques (images, son, animations), des présentations professionnelles, des simulations techniques et des productions commerciales telles que des CD-ROM. Dans le cas de KAS, nous avons utilisé le système auteur Director pour assembler nos divers médias et produire ainsi des séquences de films correspondant aux différents déplacements de l'agent. Director est un outil de développement multimédia très robuste qui permet de mettre de l'art dans les présentations.

3.1 Description des concepts de Director

Director offre un écran ou une fenêtre à chaque fois que son utilisateur veut effectuer une tâche particulière en commençant par le point d'entrée qui est la fenêtre principale. Pour comprendre comment Director marche, il est essentiel de connaître la terminologie de ses différents éléments.

1. **Le film.** Un film constitue l'élément de base d'une animation Director. Toutes les présentations de KAS sont des films Director qui jouent et qui s'enchaînent.

2. **La fenêtre principale.** La Fenêtre principale est celle qui se présente au démarrage de l'outil.

3. **Les acteurs.** Les acteurs représentent les éléments de base d'un film. Ils peuvent être une image bitmap, du texte, du son, des boutons, ou autres, et sont souvent réalisés dans d'autres applications et importés dans Director. Dans le cas de KAS, tous les acteurs sont importés dans Director.

4. **Les images.** Les images sont des objets qui représentent quand, où et comment les acteurs apparaissent dans le film.

5. **La scène.** la scène est l'endroit d'assemblage d'un film Director. La scène montre ce qui se passe dans le film à un instant donné. C'est sur la scène que tous les déplacements et mouvements de KAS sont présentés à la personne qui visualise la présentation.

6. **Le scénario.** le scénario montre ce qui se passe dans le film sur une période. C'est un enregistrement image par image de l'animation. Le scénario montre l'état de tous les éléments dans le film durant un période de temps donnée.

7. Les cellules. Le scénario est composé de cellules, unités de stockage qui contiennent toutes les informations de l'animation. Chaque colonne de cellules forme une image et chaque rangée de cellules forme une piste.

8. La fenêtre de distribution. La fenêtre de distribution est le lieu de stockage des acteurs de Director.

9. Tableau de commande. Le tableau de commande ressemble à celui d'un magnétoscope. Il permet de rembobiner, de lire, d'avancer rapidement, de stopper et de faire une pause sur une animation, ainsi que de régler la cadence de l'animation.

3.2 Fonctionnement de Director

Pour produire une animation, il faut suivre des étapes successives et bien précises. La première chose à faire est de rassembler les acteurs. Les acteurs peuvent être des graphiques, des sons, du texte, des boutons, des vidéos numérisées ou des palettes de couleurs. L'animation est créée en plaçant les acteurs dans le scénario. Le scénario est un enregistrement détaillé qui dit à Director ce que les acteurs doivent faire image par image.

L'animation est jouée sur la scène de Director, qui est l'arrière-plan sur lequel les acteurs graphiques sont animés.

Les acteurs sont stockées dans la distribution qui représente en quelque sorte une base de données multimédia.

Les acteurs sont numérotés dans les petites fenêtres qu'ils occupent, leur type est indiqué par une petite icône dans le coin inférieur droit.

Une fois créés et réunis dans la Distributuion, les acteurs peuvent être intégrés à l'animation. Il faut les placer dans le scénario, ou les poser sur la scène. Les acteurs sont placés dans le scénario en les déplaçant de la fenêtre Distribution vers une cellule d'une

image et d'une piste spécifique.

Une animation est créée en positionnant les acteurs sur la scène et en faisant varier la position de ces acteurs à travers la succession des images de l'animation. L'animation prend forme quand les images sont jouées rapidement.

Director offre aussi les propriétés d'un magnétoscope par le biais du tableau de commande. Ainsi, l'utilisateur peut par exemple rembobiner un film, changer la vitesse de l'animation et se rendre jusqu'à l'image voulue.

3.3 Le langage de script Lingo

Lingo est le langage de script de Director qui permet de faire plus que tout ce que l'utilisateur peut faire avec le scénario. Lingo est un langage de script avec lequel on peut notamment naviguer et explorer une animation, prendre en compte des actions utilisateurs (interactivité), manipuler du texte et contrôler du son.

Le mécanisme de base dans Lingo est constitué par des messages et des scripts. Par exemple, des messages comme **enterFrame** et **exitFrame** montrent que la tête de lecture est arrivée ou a quitté l'image courante.

3.3.1 Présentation de quelques éléments de programmation

Le langage Linga comprend un certain nombre d'éléments pour construire les instructions et réaliser des effets désirés. En voici quelques uns.

1. `put`. La commande "`put`" est principalement utilisée pour afficher un message dans la fenêtre de message.

2. `go to`. La commande "`go to`" permet de faire un branchement vers un emplacement spécifié.

3. if ...**then... else.** Cette instruction permet de tester une condition et d'exécuter ou non une instruction.

4. **repeat while.** Toujours terminée par un **"end repeat",** cette commande exécute toutes les instructions à l'intérieur de cet espace d'exécution tant que la condition qui suit **''while"** est vraie. Un déplacement typique de KAS est une superposition des images du personnage dans différentes positions. Cette superposition est réalisée à l'aide d'une boucle **"repeat while".**

5. **the visible of sprite.** Cette commande permet de rendre visible ou invisible un acteur. Dans les présentations de KAS, nous remarquons que lorsque l'agent est dans le bureau d'un professeur, les domaines d'intervention ne sont pas visibles et il faut amener le pointeur de la souris sur l'ordinateur pour afficher les domaines d'intervention; tout ceci est réalisé par la commande **"the visible of sprite".**

6. **the castNum of sprite.** Cette commande retourne le numéro de l'image d'un acteur.

7. **updateStage.** Cette commande permet de mettre à jour la scène. Dans notre cas par exemple, une fois une présentation terminée, il faut utiliser cette commande pour préparer le système à une autre présentation.

Nous avons vu dans ce chapitre que le système auteur Director offre d'énormes possibilité pour produire divers médias et les programmer à l'aide du script de programmation Lin.go. En présence d'un nombre considérable de médias et par conséquent d'une multitude d'étapes de combinaison pour arriver à produire un système de présentation efficace, il nous faut des techniques appropriées pour y arriver. La planification est la technique la plus adaptée pour résoudre ce genre de problème et elle fait l'objet d'une présentation dans le chapitre suivant.

Chapter 4

La planification

4.1 Introduction

La planification est l'un des thèmes les plus anciennes et les plus dynamiques de l'Intelligence Artificielle. Les problèmes que couvrent la planification portent essentiellement sur l'exploration (search) et la résolution de problèmes, la représentation de la connaissance et la décomposition des problèmes. La résolution d'un problème impllique une bonne représentation de la connaissance et une bonne méthode de résolution.

Un plan représente les intentions d'un agent concernant la satisfaction d'un but. Il est constitué d'une séquence d'actions et peut être exécuté à tout moment; il faudra s'assurer que chaque action du plan est exécutable et que l'exécution du plan va nécessairement conduire à la satisfaction du but qui l'a initié (12].

L'exécution d'un plan entraîne un changement dans le monde environnant. Chaque action constituante du plan doit contribuer au succès de ce dernier.

Un système de planification procède généralement à la résolution d'un problème suivant les cinq étapes suivantes:

- Choix de la bonne règle de résolution du problème basée sur des heuristiques

- Application de cette règle pour créer un nouvel état
- Perception de la présence d'une solution
- Perception des états inutiles qui doivent être évités
- Perception d'un état proche de la solution et utilisation de celui-ci pour arriver à une solution

Ainsi, les systèmes de planification se démarquent par la représentation ou la modélisation des objets de leur environnement, mais aussi par la modélisation de la manière dont les actions peuvent changer le monde.

4.2 Le planificateur SimPlan

Dans la plupart des applications informatiques, nous observons un certain dynamisme dans les comportements des différentes unités constituantes. Ces changements de comportement sont dus au fait que le système entier doit se mettre dans plusieurs états successifs pour pouvoir accomplir une tâche. Au départ le système se place dans un état donné, et avec la réception des messages relatifs à l'exécution des tâches, les variables changent et le système se place dans un autre état. Il faut noter que ce processus est répétitif.

En étant dans un etat donné, il n'est pas toujours facile de décider quelles sont les instructions à exécuter pour atteindre un autre état. Ceci nous amène dans un processus de planification qui va prendre en compte les actions de base et des règles pour contrôler effectivement le système. Nous présentons dans la suite de ce chapitre, le planificateur SimPlan[8] qui est responsable de la génération des séquences d'actions pour notre système.

4.3 Caractéristiques de Sim.Plan

Sim.Plan est un système de planification basé sur la simulation des comportements des systèmes. La nature changeante dans le comportement d'un système démontre un passage continuel d'un état à un autre en vue de réaliser un but donné. La tâche principale de SimPlan est de générer des plans sous forme de séquences d'actions. Les actions sont représentées par des fonctions de transition d'état. Pour parvenir à un plan, les actions sont combinées et transitent par différents états (actifs ou inactifs) successifs; SimPlan simule alors différentes combinaisons de ces actions pour ensuite retenir celui qui satisfait le mieux au but visé.

SimPlan se distingue des autres systèmes de planification par les points suivants:

- Actions non-déterministes. Dans Sim.Plan, des actions non-déterministes peuvent être spécifiées; ceci est utile dans le cas où ces actions peuvent avoir plusieurs effets potentiels difficiles à déterminer avant l'exécution.
- Buts temporels. Pour exprimer un état final souhaité ou bien des stratégies de contrôle, Simplan permet d'exprimer des buts temporels.
- Stratégies de contrôle de simulation. Les stratégies de contrôle peuvent être spécifiées pour orienter le système vers les comportements qui sont plus susceptibles d'amener au but souhaité.
- Plans optimaux. Dans une situation dans laquelle résultent plusieurs plans qui peuvent satisfaire à un but temporel, SimPlan peut en générer un qui est optimal lorsque l'option est spécifiée.
- Plans alternatifs. Dans une situation dans laquelle l'on a plusieurs plans qui peuvent satisfaire à un but temporel, Simplan peut les générer tous si l'option est spécifiée.

4.4 Utilisation de SimPlan

Pour permettre à Sim.plan de raisonner et de trouver un plan, l'utilisateur doit lui communiquer certaines informations sous forme d'actions primitives concernant ce que son système fait en général. Dans le cas de KAS par exemple, nous avons spécifié à Simplan que l'agent peut tourner à gauche, traverser un couloir, aller dans un bureau, etc. Par la suite, l'utilisateur va spécifier des buts désirés qui constituent des tâches spécifiques que le système doit performer. On peut signifier à K.I\S par example d'aller dans un bureau bien précis. Mais il y a d'autres procédures à faire avant de lui communiquer les différents états (initial et but).

- Chargement de SimPlan. Celui-ci se fait par l'intermédiaire de certains fichiers décrivant tous les aspects du planificateur.

- Spécification des préférences. Les préférences sont divisées en trois catégories: la définition du domaine, la définition du problème et les préférences indépendantes du domaine. Un domaine est un ensemble de comprtements spécifiés par une fonction de transition d'état. Un problème est un sous ensemble de comportements qui satisfont à une contrainte temporelle dans un domaine. Les préférences indépendantes du domaine sont des paramètres qui déterminent la méthode et les heuristiques utilisées par SimPlan pour générer un plan.

  Les entrées de SimPlan constituent une part des préférences. Ce sont des actions primitives décrivant les caratéristiques du système auquel il sera appliqué. Dans le cas de KAS par exemple, nous avons modelisé certaines actions de base telles *(DansCouloir KAS ?coul)* pour signifier la présence de KAS dans un couloir quelconque, ou encore *(Porteferme ?bureau ?porte)* pour signifier que la porte d'un certain bureau est fermée. C'est sur ces actions primitives que SimPlan va se baser pour simuler les comportements du système tout entier.

- Appel de SimPlan. Après avoir spécifié ses préférences, l'utilisateur doit lancer le planificateur par la commande appropriée.
- Obtention des résultats. Lorsque SimPlan finit d'exécuter, il retourne un plan qui doit être interprété pour produire l'effet désiré.

  Les plans dans Sim.Plan sont de plusieurs sortes. Tout d'abord les plans peuvent être vus comme des programmes, dans ce sens que les actions primitives peuvent être comparées aux instructions primitives des programmes. Ces instructions primitives sont combinées pour réaliser une tache donnée et parallèlement, les actions primitives sont combinées au niveau de SimPlan pour décider quelles exécutions à entreprendre.

  Un plan est une liste de règles de décision qui est une structure de quatre éléments: `id`, `state`, `action`, `alt` dont les descriptions sont.

  - `id` : nombre entier unique identifiant la règle de décision
  - `state`: Etat retourné par la fonction de création d'états
  - `action` : action à prendre
  - `alt` : liste de décisions alternatives

Nous pouvons retenir suite aux descriptions faites dans les différentes sections de ce chapitre que les systèmes de planification doivent, pour résoudre un problème, choisir les bonnes règles de résolution basée sur des heuristiques, appliquer ces règles et percevoir des états de solution ou proches de la solution. Nous avons vu qu'en plus de respecter ces principes, SimPlan se démarque des autres systèmes de planification en permettant de spécifier des buts temporels, en générant des plans alternatifs et optimaux et en adoptant des stratégies de contrôle qui lui permettent de mieux s'orienter vers des comportements qui amènent au but.

Dans les chapitres précédents, nous avons montré les concepts, les techniques ainsi que les principaux caractéristiques que nous avons intégré à notre agent KAS qui représente le noyau de notre travail. Nous présentons notre agent en détail dans le chapitre suivant, c'est à dire ses librairies d'actions primitives, son architecture, ses différents modules et leur fonctionnement ainsi que quelques cas concrets de son utilisation.

Chapter 5

Présentation du Kiosk Assistant (KAS)

Après avoir présenté dans les chapîtres précédents les éléments et concepts qui vont servir à réaliser la présentation, nous allons maintenant voir en détails comment KAS agit pour satisfaire son utilisateur.

Le système KAS permet de réaliser différentes présentations dans un environnement réel à son utilisateur. Il repose sur différentes composantes qui sont la librairie des actions primitives, le planificateur SimPlan, et l'environnement essentiellement constitué par le système auteur Director.

Le système opère en fonction des entrées que l'utilisateur lui fournit comme paramètres. Il s'agit généralement des préférences, d'un état initial et d'un but. À partir de œ moment, il communique avec SimPlan en lui fournissant l'état initial et le but, et en obtenant en retour les séquences d'actions à performer pour satisfaire l'usager. Une fois les séquences d'actions déterminées, le système sélectionne des médias (actions) ou bien génère de nouvelles actions. Ensuite les scripts Lingo entrent en action, permettant de produire efficacement des présentations dynamiques comme sortie à l'écran pour l'utilisateur.

Nous allons, dans les parties suivantes, présenter de façon plus détaillée, la librairie des actions et des éléments primitifs, les différentes fonctions du système et nous allons évaluer les résultats obtenus en se référant à certains travaux similaires.

5.1 Réalisation de la librairie des actions et d'éléments primitifs

Les actions et éléments primitifs qui vont permettre à KAS d'effectuer ses mouve ents sont stockés dans une librairie de base permettant ainsi une sélection au moment opportun. La libraire contient des actions de base de KAS comme par exemple une action de faire demi-tour. Ces actions primitives peuvent être des séquences de film de Director qui sont jugées plus utiles qu'un assemblage en ligne. Elles peuvent aussi être constituées de séries d'images liées ou déjà assemblées, ou bien de simples médias prêts à une combinaison.

Rappellons que l'idée principale qui motive ce travail consiste à implanter un agent qui doit rendre l'environnement de travail au Département de mathématiques et d'informatique (D.M.I.) de l'Université de Sherbrooke le plus réel possible, en vue de porter une amelioration au système existant. Sur ce point, l'effet visuel est très important.

Les médias servant à la constitution de la librairie sont réalisés et modelisés à partir d'un outil de modélisation 3D (3D Studio MAX) [1]. A chaque série d'images ou à chaque combinaison de média va correspondre une action donnée. La librairie comporte plusieurs gabarits d'actions qui peuvent soit correspondre à une utilisation générale (l'action de tourner à gauche par exemple), soit à une situation fixe (la vue d'un bureau spécifique).

Les médias étant constitués, ils peuvent être contrôlés à partir des scripts Lingo pour produire l'action désirée. Les scripts sont exécutés en arrière plan du système et produisent différents effets sur la scène. Par exemple pour tourner à gauche, le script doit

[1] 3D Studio MAX est un produit de la compagnie Kinetix

sélectionner les images qui correspondent à cette action.

Nous avons par exemple le décor suivant qui est le résultat d'une intégration de plusieurs médias par Lingo.

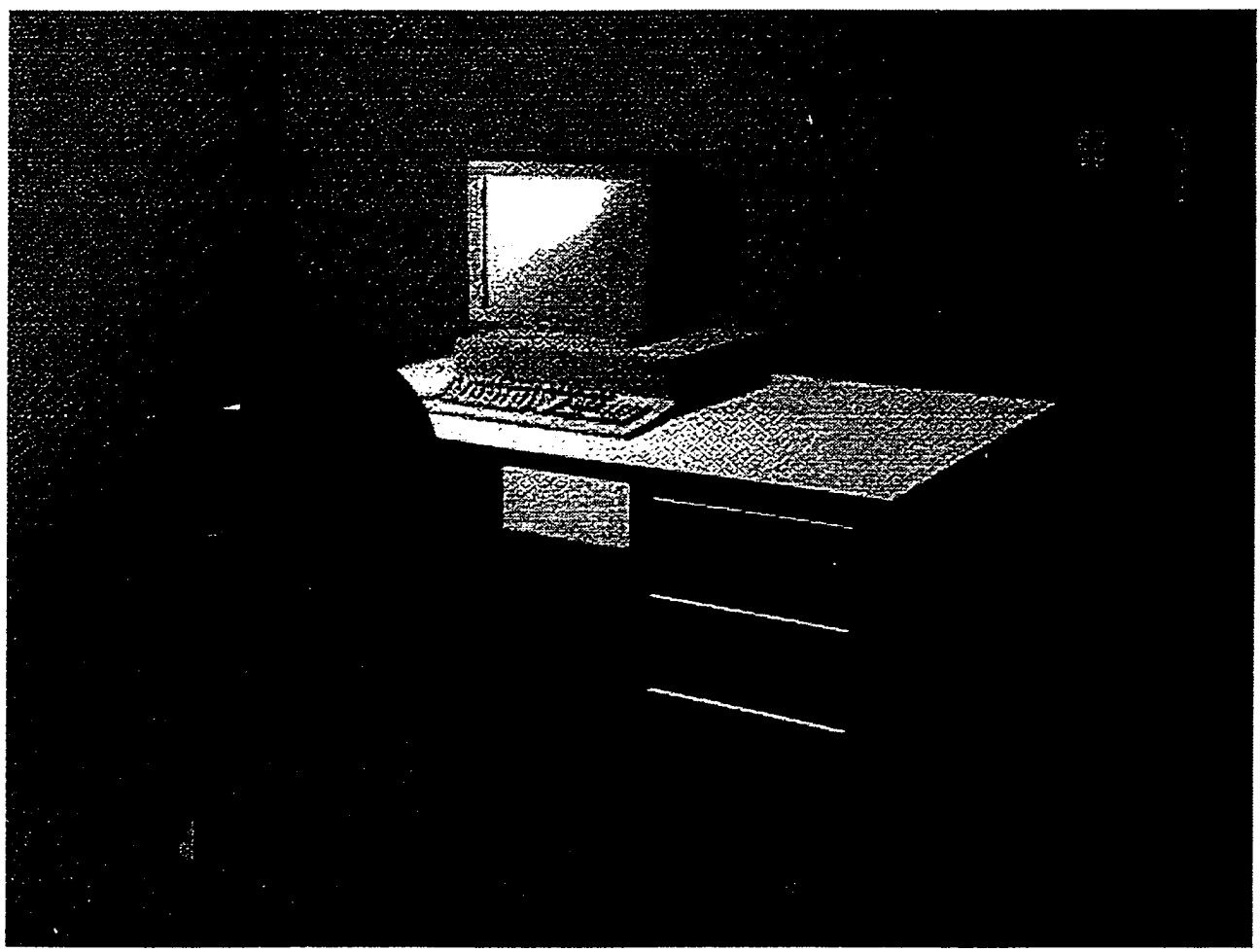


Figure 3: Exemple de décor

Nous avons vu plus haut que les médias peuvent être contrôlés par des scripts Lingo car ceci rend plus souple leur manipulation. La figure 4 montre l'exemple du script Lingo pour effectuer l'assemblage d'images en les faisant défiler sur la scène par superposition. Il faut noter dans ce script la notion de position sur la scène, ainsi que la dimension des

m dias contrôlés. Avec Lingo, la taille de tout média peut être contrôlée, permettant ainsi à KAS de montrer à l'utilisateur les différentes formes d'un objet suivant sa position et suivant la distance à laquelle l'objet est situé.

```
on exitFrame
    repeat while the mouseUp
            beep
    end repeat
    set the visible of sprite 5 to false
    if rollover(7) then
    set the visible of sprite 5 to true
    end if
    go to frame the frame
    repeat with i = 1 to 5
            set the memberNum of sprite 1 = i
            set newX = centerX + i
            set newY = centerY
            put newX
            put newY
            set the locH of sprite 1 to newX
            set the locV of sprite 1 to newY
            updatestage
            go frame the frame
            if the memberNum of sprite 1 = 5 then
            go to frame 12
            end if
    end repeat
end exitFrame
```

Figure 4: Exemple de script d'assemblage d'images

5.2 Description de KAS

La seconde partie (système KAS) va consister à planifier les mouvements de notre agent dans la présentation proprement dite.

5.2.1 Description des fonctions réalisées

Nous allons décrire les différentes modules fonctionnels regroupés dans le système de présentation. Ces modules réfl.ètent aussi les différentes manières d'utilisation présentées à l'utilisateur.

Avant de présenter en détail les différentes fonctions, il serait bon de mettre en évidence le rôle de l'utilisateur du système. Il est en effet en contact direct avec le système et se limite à rentrer des donnés ou à faire des sélections nécessaires au pilotage du système.

Parmi les principaux modules fonctionnels du système, nous distinguons :

1. **Le Planificateur.** C'est le module qui génère les séquences d'actions qui vont constituer la présentation. Notons que le planificateur utilisé ici est SimPlan. Il accepte des entrées telles l'état initial, le but et les contraintes, et produire une sortie sous forme de plan.

2. **La Librairie des éléments et actions primitifs.** C'est la base de données des diflërentes actions et éléments primitifs qui vont servir aux différents mouvements de notre agent(tourner à gauche, aller vers l'avant etc...). Elle sert à une utilisation directe par le programme. Elle sert de base de données pour le programme et fournit des actions de bases.

Mis à part ces modules fonctionnels, nous distinguons des fonctions internes du système qui sont énumérées ci-dessous.

1. **ÉtablirPréférences.** C'est à cette étape que l'utilisateur établit ses préférences par rapport à l'état initial de KAS, des contraintes souhaitées, ainsi que le but qu'il

désire atteindre. Ces préférences sont spécifiées dans des variables d'état fournies en interface à l'utilisateur.

Ce que nous appellons ici par **ÉtablirIPréférences** est cette étape que l'utilisateur doit performer pour permettre à Sim.Plan de générer un plan. Il s'agit de décrire d'une part ce que KAS est sensé faire dams un langage ou un formalisme compréhensible par SimPlan, et d'autre part de sp-écifier des buts souhaités par rapport à des états initiaux et à certaines contraintes. Un état initial peut par exemple placer KAS dans un couloir et un but voudra le faire aller dans un bureau déterminé.
Cette étape est réalisée de façon progressive c'est à dire que l'utilisateur confectionne lui-même ses préférences à partir des prédicats et des actions primitives qui lui sont disponibles. Au fur et à mesure qu'il effectue ses choix, l'utilisateur se voit offir une composition automatique de ses exigences.

2. **Élaborerplan.** Cette fonction est accor:n.plie par l'acteur Planificateur. Il prend en entrée les préférences spécifiées par la fcmction précédente **(ÉtablirPréférences)** et fournit en sortie une séquence d'action sous forme de plan.

   SimPlan décide à quels moments certaines instructions vont être activées ainsi qu'à quels moments d'autres vont être clésactivées. Ceci implique une recherche de la combinaison satisfaisante en exploran..t l'espace de toutes les combinaisons possibles. Il peut aussi s'agir de simulations cde combinaisons d'actions tout en évaluant chaque combinaison par rapport au but... Lorsqu'une combinaison satisfaisante est trouvée, celle-ci est considérée comme l e plan final.
   SimPlan ne simule pas directement les actions primitives dans un langage de programmation, mais des règles qui décriv mt les actions importantes correspondant aux processus de base d'un langage de programmation.

3. **Présenter.** Une fois les préférences d e l'utilisateur établies et les actions et éléments primitifs disponibles, le système s e charge ici de montrer à son utilisateur la

présentation résultant des actions à agencer retournées dans la fonction précédente **(Élaborerplan).** Cette présentation tient compte de la coordination temporelle des différents médias qui vont participer à la présentation.

4. **AssemblerMédias.** L'assemblage des médias est une étape complètement indépendante de l'utilisateur. Compte tenu des messages captés par les senseurs de KAS, un assemblage est taillé pour répondre aux exigences.

5. **ExécuterAction.** Cette fonction très simple n'implique que l'utilisateur lorsque celui-ci veut visionner une présentation. Il s'agit de cliquer sur le bouton qui sert à cette fin.

5.2.2 Architecture

L'architecture est composée des modules suivants :

- L'environnement. L'environnement ici est représenté par notre système auteur Director qui rend possibles toutes les interactions entre les différents composants.
- Les senseurs. Les senseurs sont des récepteurs dans le script Lingo et réagissent au monde externe.
- Les effecteurs. Les capteurs sont des instructions Lingo qui permettent de modifier l'environnement.
- SimPlan. SimPlan est notre planificateur qui génère les plans utilisés dans la présentation.
- La librairie des actions. La librairie d'actions sert de base aux différents mouvements performés par l'agent KAS.

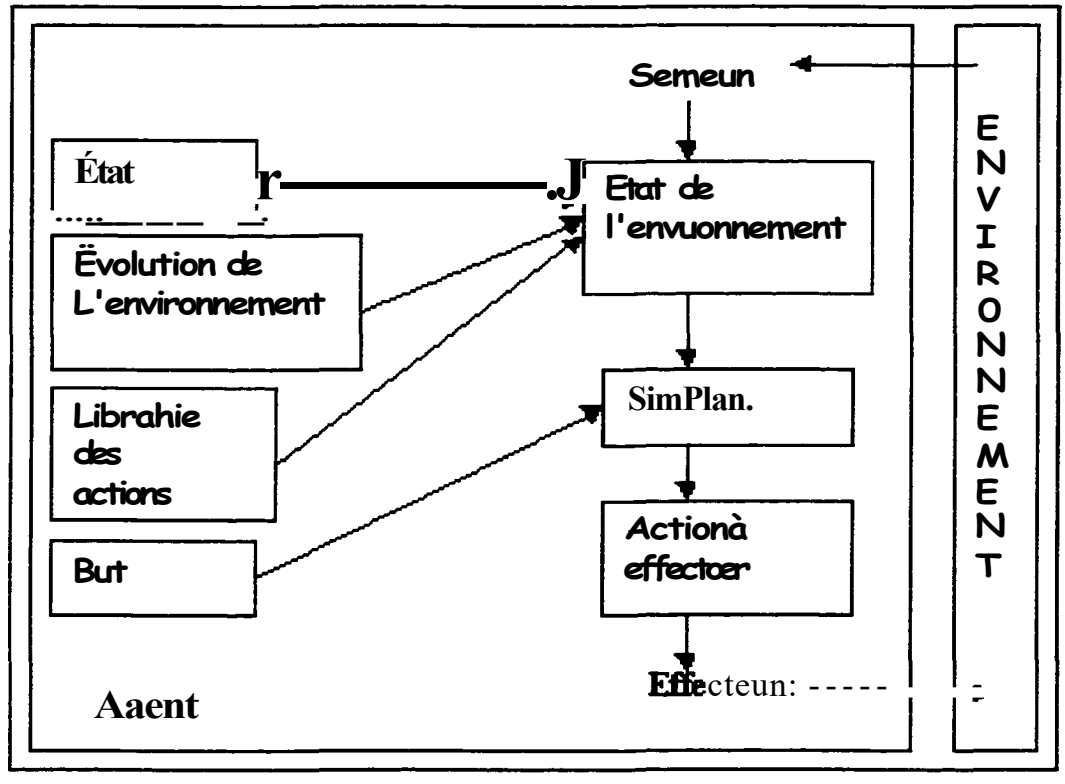


Figure 5: Architecture du système

5.2.3 Application : Modélisation du problème

Mis à part la réalisation des actions et éléments de médias primitifs, nous devons définir un cadre qui permetra au système d'envoyer des exigences à SimPlan dans le but d'obtenir des plans et les exécuter pour réaliser la présentation. Ceci constitue la deuxième grande étape du projet. Pour représenter et modeliser le problème, il a fallu tout d'abord définir les primitives qui sont des prédicats et qui sont spécifiés dans la notation STRIPS.

5.2.4 Définition des prédicats

Les différentes entrées qui sont fournies à SimPlan sont des combinaisons de prédicats déterminant tous les états avant et après l'occurence d'une action. Les prédicats sont spécifiés dans la notation STRIPS. Nous avons pour KAS défini un ensemble de prédicats dont certains sont présentés sur la figure 6. La section suivante donne une description détaillée des opérateurs.

```
(add-strips-op
 :name '(TournerDroite Kas ?coull ?coul2)
 :pre '((Agent Kas)(DansCouloir Kas ?coull)(Droite ?coul2 ?coull)(Gauche ?coull ?coul2))
 :add '((DansCouloir Kas ?cou12))
 :del '((DansCouloir Kas ?coull)(Droite ?coul2 ?coull)(Gauche ?coull ?coul2))))

(add-strips-op
 :name '(TournerGauche Kas ?coull ?coul2)
 :pre '((Agent Kas)(DansCouloir Kas ?coull)(Gauche ?coul2 ?coull)(Droite ?coull ?coul2))
 :add '((DansCouloir Kas ?coul2))
 :del '((DansCouloir Kas ?coull)(Gauche ?coul2 ?coull)(Droite ?coull ?coul2)))
```

Figure 6: Description des actions

L'opérateur *(def - described - symbols)* permet de définir les symboles sous forme de prédicats qui vont être utilisés dans les actions.

Par exemple, *(PorteOuverte predicate 2)* définit un prédicat qui prend deux arguments. Un exemple de son utilisation peut être *(PorteOuverte ?porte ?bur)* qui signifie qu'une certaine porte d'un certain bureau est ouverte.

L'opérateur *(add - strips - op)* définit des opérateurs STRIPS. Nous rappellons qu'un opérateur STRIPS comprend des paramètres suivants.

- :name. Ce paramètre indique le nom de l'opération. Un exemple de son utilisation est *(AllerCouloir Kas ?coul ?bur)*. Cette instruction signifie que le nom de l'opérateur qui indfique à l'agent KAS d'aller dans un certain couloir est Aller-Couloir.

- :pre. Ce paramètre indique les préconditions de l'opération. Ce sont des conditions qui doivent être satisfaites avant que l'opération ne soit déclenchée. Par exemple *((DansBureau Kas ?m1,-r)(PorteOuverte ?porte ?bur)(Agent Kas))* signifie que l'agent KAS est dans un certain bureau et que la porte de ce bureau est ouverte avant l'exécution de l'action.

- :add. Ce paramètre indique des conditions qui seront vraies à l'issu de l'exécution de l'opération. Par exemple *(DansCouloir Kas ?coul)* signifie qu'après exécution de l'action, l'agent KAS se retrouve dans un coulour précis.

- : del. Ce paramètre indique des conditions qui deviendront fausses à l'issu de l'exécution de l'opération. Par exemple *(DansBureau Kas ?bur)* signifie qu'après exécution de l'action, l'agent KAS ne se trouve plus dans le bureau où il se trouvait avant l'exécution de l'action.

La présentation est essentiellement basée sur des plans qui sont fournis par SimPlan qui prend en entrée une spécification d'un état initial, des contraintes et un but. Nous allons montrer par l'exemple ci-après les différentes étapes qui conduisent à l'obtention d'un plan. Pour ce faire, nous allons d'abord spécifier un état initial, un but et s'il le faut des paramètres de contrôle.

5.2.5 Exemples de planification de comportements avec SimPlan

Nous allons passer par deux exemples pour montrer comment nous obtenons les séquences d'actions. À chaque fois, nous allons montrer une entrée (état initial et but) et un plan.

Pour le premier exemple nous avons les spécifications indiquées sur la fi$_{gu}$re 7.

```
( set-prefs
  :init '((Agent kas) (DansCouloir kas cl)(Gauche c2 cl)(Droite c1 c2)
          (PorteFerme portel bl))
  :goal '(eventually (DansBureau Kas hl)))
```

Figure 7: Premier exemple d'état initial

La spécification de l'état initial est décrite par le terme :init qui, comme son nom l'indique, signifie à SimPlan qu'il s'agit d'un état initial. Ensuite viennent des spécifications des prédicats: *(DansCouloir kas* c1) signifie que l'agent KAS est dans le couloir c1, *(Gauche* c2 **cl)** qui spécifie que le couloir c2 est situé à gauche du couloir c1, *(Droite* **cl** c2) qui spécifie que le couloir c1 est situé à droite du couloir c2, et *(PorteFerme portel* bl) qui spécifie que la porte portel du couloir h1 est fermée.

Le but est spécifé par le terme :goal. Dans l'exemple précédent, nous obtenons comme but *(eventually (DansBureau Kas* bl)).

Comme on pourrait le constater, le but est qu'éventuellement, l'agent KAS se retrouve dans le bureau b1.

Une fois l'étape de spécification de l'état initial et du but terminée, SimPlan est appelé et un plan est généré. Notons qu'il peut ne pas avoir de plan dans certains cas.

La fonction d'interface avec SimPlan fournit en sortie tous les éléments du plan. Le

ID Ostate ((droite c1 c2) (gauche c2 c1) (danscouloir kas c1) (porteferme portel b1))
ACTION ((tournergauche kas c1 c2) 1)
ID 1 state ((danscouloir kas c2) (porteferme portel b1) (agent kas))
ACTION ((ouvrirporte kas portel b1) 2)
ID 2 state ((danscouloir kas c2) (porteouverte portel b1) (agent kas))
ACTION ((allerbureau kas b1) 3)
ID 3 state ((dansbureau kas b1) (porteouverte portel b1) (agent kas))
ACTION ((fermeporte kas portel b1) 4)
ID 4 state ((dansbureau kas b1) (porteferme portel b1)) ACTION ((wait 1) 4)

Figure 8: Premier exemple de plan

plan retourné est interprété par le système qui en plus coordonne les mouvements de KAS en conséquence. Dans notre exemple, les plans produits par SimPlan sont dans un format bien déterminé facile à toute interprétation. L'énumération du plan est décrite dans la figure 8.

Nous notons d'abord un numéro d'identification de l'action, suivi des conditions qui deviendront valides, et de l'action proprement dite. La première action par exemple a pour numéro d'identification 0, et consiste à tourner à gauche pour se rendre dans le couloir c2 *((tournergauche kas* c1 c2) 1). Après avoir touné à gauche, KAS va ouvrir la porte du bureau b1 *((ouvrirporte kas portel* b1) 2), se présente ensuite dans le bureau *((allerbureau kas* b1) 3), et finalement ferme la porte derrière lui par l'instruction *((Jermeporte kas portel* b1) 4).

En modifiant les spécifications en entrée, nous obtenons un plan différent. C'est ce que nous montre le deuxième exemple (fi$_{g\,u}$re 9).

Dans cet exemple, nous avons KAS dans le couloir c1 *(DansCouloir kas* c1**)** et il est

```
( set-prefs
  :init '((Agent kas) (DansCouloir kas cl)(PresDe Kas Bureau!))
        (PorteFerme p1 bureau!))
  :goal '(eventually (DansBureau Kas bureau!)))
```

Figure 9: Deuxième exemple d'état initial

```
ID Ostate ((danscouloir kas cl) (porteferme p1 bureau!) (presde kas bureau!))
ACTION ((ouvrirporte kas p1 bureau!) 1)
ID 1 state ((danscouloir kas cl) (porteouverte p1 bureau!) (presde kas bureau!))
ACTION ((allerbureau kas bureau!) 2)
ID 2 state ((dansbureau kas bureau!) (porteouverte p1 bureau!) (presde kas bureau!))
ACTION ((fermeporte kas p1 bureau!) 3)
ID 3 state ((dansbureau kas bureau!) (porteferme p1 bureau!) (presde kas bureau!))
ACTION ((wait 1) 3)
```

Figure 10: Deuxième exemple de plan

proche du bureau Bureau! *(PresDe Kas Bureau!)*. Ensuite nous fermons la porte du bureau bereaul *(PorteFerme* p1 *bureau!)*. Ces trois spécifications de prédicats sont accompagnées de la spécification d'un but qui veut qu'éventuellement, KAS se retrouve dans le bureau bureau! *(eventually (DansBureau Kas bureau!))*.

Le résultat de la figure 10 mentionné nous démontre que KAS ouvre dans un premier temps la porte du bureau bureau! *(ouvrirporte kas* p1 *bureau!)* 1), puis va dans le bureau *((allerbureau kas bureau!)* 2) et finalement ferme la porte derrière lui par l'instruction *(Jermeporte kas* p1 *bureau!)* 3).

Il faut remarquer que si la spécification en entrée est incorrecte, il n'y aura aucun plan en sortie; ceci est bien entendu différent de la situation dans laquelle SimPlan juge qu'il ne peut y avoir de plan, vu les états à considérer.

5.2.6 Communication avec SimPlan

Pour réaliser des présentations, il faut avoir des plans et pour obtenir des plans, il faut communiquer avec SimPlan. La manière la plus directe de mener à bien cette communication est de la faire en mode transparent lorsque la présentation joue, c'est à dire que l'utilisateur ne s'apercevra pas qu'il y a un appel qui est fait à SimPlan et tout se passera pour lui comme s'il s'agissait d'un seul système. Nous nous servons des fichiers pour dans un premier temps envoyer des requêtes à SimPlan et pour récupérer le plan dans un second temps. Les requêtes envoyées à SimPlan ne sont autres que les préférences de l'utilisateur lorsque celui-ci pilote le système. En effet, après que l'utilisateur ait spécifié ses préférences, nous déclenchons le processus d'exécution de SimPlan et une fois le plan généré, œ processus prend fin et le contrôle est redonné à notre application.

5.2.7 Résultats

Les étapes présentées ci-dessus montrent un cheminement qui mène à l'obtention des séquences d'actions par SimPlan; il ne reste qu'à les interpréter et les mettre à exécution. La prise en compte de ces séquences d'actions et leur interprétation nous ont permis de réaliser l'assemblage des médias qui ont servi pour la présentation. La présentation est réalisée par l'agent KAS dans l'environnement virtuel du D.M.I. obéissant ainsi aux choix de l'utilisateur.

Notre personnage est une image d'une personne réelle réalisée à l'aide de l'outil 3D Studio MAX de Kinetix et d'une série d'images déjà existantes.

Les différents mouvements du personnage sont la combinaison d'une série d'images (autour d'une quinzaine par mouvement). Nous avons modélisé un certain nombre d'actions de base dont les principales sont les suivantes.

- Aller vers l'avant. Cette action montre Pagent qui va vers l'avant à partir de l'endroit où il se trouve, en faisant dos à l'utilisateur.
- Aller vers la droite. Cette action montre l'agent qui va vers la droite à partir de l'endroit où il se trouve, en se déplaçant vers la droite de l'utilisateur.
- Aller vers la gauche. Cette action montre l'agent qui va vers la gauche à partir de l'endroit où il se trouve, en se déplaçant vers la gauche de l'utilisateur.
- Tourner à droite. Cette action montre Pagent qui tourne à droite à partir de l'endroit où il se trouve, c'est à dire vers l'utilisateur.
- Tourner à gauche. Cette action montre l'agent qui tourne à gauche à partir de l'endroit où il se trouve, c'est à dire faisant dos à l'utilisateur.

La librairie peut être étendue plus tard pour inclure certaines actions telles prendre un objet, faire des mouvements de tête etc.

La présentation animée par le personnage de KAS comme nous l'avons vu dans tout le document est réalisée par les images modelisées qui sont programmées en Lin.go (l'outil de programmation de Director). Le système auteur Director offre une grande qualité de production et d'assemblage de médias.

L'ensemble du système permet une interactivité avec l'utilisateur. Au début, il s'agit de faire certains choix (état initial et but par exemple) qui sont offerts par le menu principal. Une bonne utilisation de KAS passe d'abord par une spécification de paramètres qui permettront d'obtenir des plan et de démarrer la présentation. Tout au long de son parcours, l'utilisateur peut arrêter la présentation s'il le veut. SimPlan et son langage de

description des actions nous ont permis de modeliser le monde de KAS par des prédicats et d'obtenir des résultats rapides et satisfaisants au niveau de l'obtention des séquences d'actions. Le système n'aurait pas été aussi performant sans la capacité de génération des actions pas SimPlan. En effet, pour un système ordinaire, il faut contrôler tous les états par le programme, ce qui va entrainer une lourdeur et diminuer sa performance.

Il est possible de modeliser tout un autre problème de présentation. Un autre exemple tout aussi intéressant serait le problème des cours de gymnastique. Dans ce problème, nous aurons des prédicats tels que *(PlierGenoux ?genoux)* pour plier un genoux quelconque de KAS, *(LeverBras ?bras)* pourque KAS lève un bras spécifique.

En définitive, nous avons obtenu des résultats satisfaisants avec SimPlan, mais il nous faudra utiliser d'autres planificateurs pour situer le niveau exact de la compétition.

D'autre part, la souplesse de Lingo nous a permis de contrôler efficacement la grande partie de l'assemblage et du défilement des médias. Nous avons utilisé une structure de programmation simple mais efficace ce qui nous amène à déduire que ce n'est pas le volume du code qui donne de l'efficacité à un système.

Le système de présentation va se dérouler presque de la même manière que le système d'animation en ce qui concerne la présentation des différents écrans. C'est un système interactif; l'utilisateur démarre le système, et sélectionne des options à partir de différentes menus qui lui sont offerts suivant qu'il veut effectuer une présentation ou non.

L'algorithme général qui nous a servi à formuler un problème, à effectuer l'exploration des séquences d'action, et à exécuter une action par notre agent est décrite par les lignes suivantes après les paramètres qui sont les suivants.

- Le paramètres qui représente une séquence d'actions initialement vide
- Le paramètre Etat qui est la description de l'état actuel de l'environnement
- Le paramètre But qui est un but initialement nul

- Le paramètre Problème qui est la formulation du problème

1. Si s est vide alors
2. Formuler le problème et formuler le but
3. Effectuer l'exploration des séquences d'action et prendre une action
4. Mettre s à jour et retourner une action

Dans ce chapitre, nous avons présenté comment les concepts décrits dans les chapitres précédents sont intégrés à KAS. KAS présente les caractéristiques principales de tout agent réactif de même que celles des systèmes de présentation multimédia. Nous avons aussi montré la communication avec SimPlan pour la génération des séquences d'actions et quelques exemples d'utilisation. Ce chapitre nous permet donc de conclure que les systèmes de présentation multimédia peuvent bien être améliorés en y introduisant la planification.

```
 ( ((def-described-symbols)
   '(Agent predicate 1)
   (PresDe predicate 2)
   (PorteOuverte predicate 2)
   (PorteFerme predicate 2)
   (DansCouloir predicate 2)
   (Gauche predicate 2)
   (Droite predicate 2)
   (DansBureau predicate 2)
   (AllerCouloir predicate 2)))

(add-strips-op
   :name '(AllerBureau Kas ?bur)
   :pre '((Agent Kas)(DansCouloir Kas ?coul)(PorteOuverte ?porte ?bur))
   :add '((DansBureau Kas ?bur))
   :del '((DansCouloir Kas ?coul)))

(add-strips-op
   :name '(OuvrirPorte Kas ?porte ?bur)
   :pre '((Agent Kas)(PorteFerme ?porte ?bur)
   :add '((PorteOuverte ?porte ?bur))
   :del '((PorteFerme ?porte ?bur)))

(add-strips-op
   :name '(FermePorte Kas ?porte ?bur)
   :pre '((Agent Kas)(PorteOuverte ?porte ?bur)(DansBureau Kas ?bur))
   :add '((PorteFerme ?porte ?bur))
   :del '((PorteOuverte ?porte ?bur)))

(add-strips-op
   :name '(AllerCouloir Kas ?coul ?bur)
   :pre '((DansBureau Kas ?bur)(PorteOuverte ?porte ?bur)(Agent Kas))
   :add '((DansCouloir Kas ?coul))
   :del '((DansBureau Kas ?bur))
```

Figure 11: Description des actions (suite)

Conclusion

Nous avons démontré à travers ce travail un exemple de communication entre un système et un utilisateur. Tout d'abord, nous avons clarifié le concept des agents logiciels pour être capable de doter KAS des caractéristiques réels que l'on peut observer au niveau l'action des agents en général. Ainsi l'on peut par exemple observer dans l'architecture de KAS qu'il y a une distinction fonctionnelle entre ses senseurs (variables d'états Lingo) et ses effecteurs.

En outre, nous avons montré comment le système tout entier répond aux exigences des systèmes de présentation multimédia, en offrant de l'interactivité à son utilisateur, en sélectionnant le bon média à assembler et à présenter, et en déterminant qu'une animation aurait plus d'impact réel sur l'usager que de simples présentations de médias. Aussi KAS répond bien aux tâches dont il est chargé. Nous convenons qu'il n'est que système initial prêt à évoluer et qu'il est encore loin de compétitionner avec les systèmes commerciau.."C disponibles sur le marché.

L'environnement multimédia qui rend la présentation plus réaliste a donné une autre dimension à la portée du concept. Mais notons aussi la performance de SimPlan qui a ajouté à ce système d'agent de présentation particulier sa coloration intelligente. En effet, l'un des enjeux de ce travail était d'exporer un domaine de l'intelligence artificielle. ce qui a été fait par une vue générale des techniques de planification et par l'application de SimPlan.

L'exemple simulé ici est simple mais son évaluation nous montre des résultats satisfaisants au niveau des plans et de l'adéquation dans la présentation. Kas est doté d'une bonne structure, œ qui fait que le nombre de lignes de script Lingo est relativement petit. Ceci permet d'ailleurs une bonne maintenance et une bonne réutilisation. En définitive, bien que Kas soit encore loin de l'état de l'art dans le domaine, il peut être amelioré et constituer un bon example dans sa catégorie.

Nous avons par ailleurs montré à travers œ travail que KAS peut effectuer ses actions dans un environnement physique réel en profitant des nombreuses actions de base qui constituent la librairie des actions. L'architecture du système, flexible et modulaire est composée de l'environnement de Director, du planificateur Sim.Plan, d'une librairie d'actions, des senseurs et des effecteurs.

L'environnement Director permet l'intégration de tous les modules pour proposer une présentation décente à l'utilisateur. L'utilisateur a un vaste choix au niveaux des actions et des présentations qu'il souhaite visualiser dans une coordination temporelle naturelle. Le système aura contribué à améliorer la façon existante de communiquer avec le D.M.I..

Le planificateur SimPlan, apport en intelligence artificielle du système a par ses particularités (actions non-déterministes, buts temporels, simulation des stratégies de contrôle, optimisation des plans, plans alternatifs) permis de prendre un avantage sur d'autres systèmes limités au niveau de la génération des séquences d'actions.

Le fait de penser KAS dans une perspective agent est par ailleurs un avantage considérable si l'on sait que cela permet de réfléter la réalité et entre autre d'ouvrir la voie à une personnalisation de la prsentation.

Le système de KAS est une application rapide et consistante, l'utilisateur aura comme présentation œ qu'il a choisi de voir.

Loin de compétitionner avec les agents de présentation multimédia commerciaux tels PPP, KAS peut néanmoins être amélioré pour plus de performance. L'on peut envisager l'adapter à un environnement distribué et le rendre mobile où un utilisateur peut en

charger une copie spécifique au besoin, ce qui facilitera son apprentissage. Dans ce contexte, Sim.Plan serait enveloppé comme module distribué et vu comme un serveur de plans. Il n'exposera que des services qu'il va offrir à ses utilisateurs. Par exemple on peut avoir un service du nom de **fournirPlan()** qui prendra en entrée toutes les informations qui vont lui être necessaires pour produire un plan. En ce moment, plusieurs instances de Kas peuvent à tout moment utiliser ces services et montrer plusieurs présentations à la fois. Dans une structure distribuée, la librairie d'actions sera un serveur de données multimédia renfermant des images vidéo, de !'hyp ertexte, du son , des fichiers bitmaps et autres.

Au stade actuel, KAS est implémenté pour naviguer dans un environnement physique et faire une présentation, mais il peut être adapté à un grand nombre de situations. Chaque environnement physique étant unique, il faudra pour ce faire produire la librairie des actions primitives qui vont rentrer dans la présentation. L'utilisateur aura donc à effectuer un choix par rapport à son profil avant de procéder. Par exemple, dans un cadre plus général de recherche d'information (sans contrainte physique au niveau de l'environnement), l'utilisateur peut choisir d'ignorer l'environnement et le problème devient beaucoup plus simple.

L'une des améliorations que nous pouvons faire concernant KAS est de lui intégrer la capacité de pouvoir permettre à l'utilisateur de changer d'objectif au cours d'une présentation. À n'importe quel moment d'une présentation, tout le processus de génération de séquences d'actions va être rendu possible et le système sera encore plus souple d'utilisation.

Appendix A

Annexe

A.1 Exemples de scripts Lingo utilisés pour KAS

```
on startMovie
    on readdata pathAndName
    set leContenu to ""
    set monFichier to new(xtra "fileio")
    if objectP(monFichier) then
      if not stringP(pathAndName) then
        if the machineType = 256 then
          setFilterMask(monFichier, "Text Document,*.txt")
        else
          setFilterMask(monFichier, "TEXT ????")
        end if
        set pathAndName to displayOpen(monFichier)
      end if
      if pathAndName il EMPTY then
        openFile(monFichier,pathAndName, 1)
          if not status(monFichier) then
            setPosition(monFichier,0)
            set leContenu to readFile(monFichier)
            closeFile(monFichier)
            put leContenu into field "fiches"
          end if
      end if
    end if
    set monFichier to 0
    set EOL to RETURN  numToChar(l0)
    set leContenu to searchReplace(leContenu, EOL, RETURN)
    return leContenu
end
```

Figure 12: Exemple de scripts Lingo utilisés pour KAS

```
on exitFrame
    put the locH of sprite 10 into centerX
    put the locV of sprite 10 into centerY
    put centerX
    put centerY
    repeat with i = 2 to 122
    repeat with i = 2 to 122
    put centerX
    put centerY
    set the memberNum of sprite 1 = i
    set the memberNum of sprite 2 = i + 129
    set the memberNum of sprite 1 = i
    set the memberNum of sprite 2 = i + 129
    set the locH of sprite 1 to newX
    set the locV of sprite 1 to newY
    sound playFile 1, "duke.wav"
    updatestage
    go frame the frame
    if the memberNum of sprite 1 =122 then
    go to frame 1 of movie "opendoorb.dir"
    end if
    end repeat
end
on exitFrame
    put the locH of sprite 1 into centerX
    put the locV of sprite 1 into centerY
    repeat with i = 46 to 60
    set the memberNum of sprite 1 = i
    set newX = centerX +i
    set newY = centerY
    put newX
    put newY
```

Figure 13: Exemple de scripts Linga utilisés pour KAS (suite)

```
set the locH of sprite 1 to newX
set the locV of sprite 1 to newY
updatestage
go frame the frame
if the memberNum of sprite 1 = 60 then
go to frame 12
end if
end repeat
set theFile to "input.txt"
set the text of field "plan" to ""
set CRLF = ReturnnumToChar(lO)
set partiel = "(sdomain kas)" CRLF "(set-prefs mode GSEMDP)" CRLF
set partie2 = "(set-prefs init ((Agent kas)"
put the text of field "init" into partie3
set partie4 ="(Gauche c2 cl)(Droite c1 c2)(PorteFerme portel bl))" CRLF
set partie5 = "goal (eventually"
put the text of field "but" into partie6
set partie7 = "))"
end
```

Figure 14: Exemple de scripts Linga utilisés pour KAS (suite)

Bibliography

[l] E. André and T. Rist. Coping with temporal constraints in multimedia presentation planning. In *Proceedings of the Thirteen National Conference AAAI,* volume 1, pages 142-147, Portland, Oregon, 1996.

[2] E. André and T. Rist. The ppp persona : A multipurpose animated presentation agent. In *Advanced Visual Interfaces,* pages 245-247, A.CM Press, 1996.

[3] K. Boukerche. Planification des présentations multimédias. Mémoire de maîtrise. Faculté des Sciences, Université de Sherbrooke, Qc, Canada. No. III-1158, 1998.

[4] J. M. Bradshaw. Introduction to software agents. Jeffrey M. Bradshaw, editor, *Software Agents,* pages 3-46.

[5] O. Etzioni and D. S. Weld. Intelligent agents on the internet - facts, fiction, and forecast. *IEEE Expert, Intelligent Internet Services,* volume 10, number 4, pages 44-49, 1995.

[6] B. A. Goodman. Multimedia explanations for intelligent training systems. In *Intelligent Multimedia Interfaces,* pages 148-171, AAAI Press/MIT Press, 1993.

[7] A. Graesser and S. Franklin. Is it an agent, or just a program ? a ta..xonomy for autonomous agents. In *Proceedings of the Third International Workshop on Agent*

Theories, Architectures, and Languages, Intelligent agents III, pages 21-35, Spring-Verlag, Berlin, 1997.

[8] F. Kabanza. Simplan's theoritical background. Unpublished paper that can be found at http://www.dmi.usherb.ca/kabanza/,...., simplan, 1999.

[9] M. Mauldin. Chatterbots, tinymuds, and the turing test: Entering the leobner prize competition. *AAAI,* volume 1, pages 16-21, Seattle,1994.

[10] H. S. Nwana. Software agents: An overview. In *Knowledge Engineering Review,* volume 11, pages 1-40. Cambridge University Press, September 1996.

(11] C. Petrie. What is an agents ? In *Lecture Notes in Artificial Intelligence,* number 1193, pages 41-43, Springer-Verlag, Berlin, August 1996.

[12] M. Pollack. *Inferring domain plans in question-answering.* PhD thesis, University of Pennsylvania.

[13] F. Roth and W. E. Hefl.ey. Intelligent multimedia presentation systems : Research and principles. In M. Maybury, editor, *Intelligent Multimedia Interfaces,* 1993.

[14] T. K. Shih and R. E. Davis. Immps: A multimedia presentation design system. *IEEE Multimedia,* volume 4, number 2, pages 67-78, 1997.