Architecture parallèle pour les mondes virtuels

Patrice TORGUET, Bruno MOISAN, René CAUBET
I.R.I.T. Université Paul Sabatier,
118, Route de Narbonne, 31062 TOULOUSE CEDEX
e-mail : torguet@irit.fr

Introduction

Depuis plusieurs années déjà, avec le projet VOXAR, l'équipe synthèse d'images de l'IRIT a étendu ses activités de recherche à l'utilisation du parallélisme pour les problèmes de rendu en synthèse d'images. Les résultats obtenus avec ce projet ont montré l'intérêt d'utiliser des machines à mémoire distribuée. Le projet VOXAR nous a notamment permis de construire une machine parallèle (bâtie sur un réseau de 36 Transputers) dédiée au rendu d'images réalistes et plus généralement à des applications de simulation d'objets 3D dans une subdivision régulière de l'espace [1,2]. Un noyau de système de communication, supportant le dialogue transparent entre deux tâches quelconques indépendamment de leur localisation dans le réseau a été développé ainsi que deux applications (un lancer de rayons [1,3] et une radiosité progressive étendue [4,5]). De plus un noyau d'animation conventionnelle a été greffé sur l'application de lancer de rayons [3].

Dans le monde de la synthèse d'images, le souci majeur est de s'approcher le plus possible de la réalité. Ces besoins en réalisme n'ont cessé de s'accroître dans les applications et nécessitent des modèles d'éclairage de plus en plus complexes ainsi que des bases de données de plus en plus gigantesques. De plus de nouvelles gammes d'applications ont vu le jour (simulations physiques, réalité virtuelle...).

L'ensemble du projet concerne l'étude et la réalisation d'un système complet de génération de séquences d'images de synthèse d'un très grand réalisme en minimisant les coûts tout au long de la chaîne de production. Nous souhaitons définir une architecture modulaire offrant l'ensemble des matériels et des logiciels nécessaires tout au long du processus. De plus nous souhaitons définir une architecture suffisamment souple pour pouvoir intégrer facilement des dispositifs matériels spécialisés permettant d'utiliser les récentes techniques de la réalité virtuelle comme aide à la modélisation dans les différentes étapes de création (définition des objets, de l'aménagement de l'espace et des mouvements).

Pour atteindre cet objectif, nous poursuivons plusieurs projets de recherches sur le parallélisme et le rendu. La composante commune de ces travaux est une orientation volontaire, vers l'exploitation d'un parallélisme de données pour utiliser au mieux les capacités de stockage et de calcul des machines à mémoire distribuée. Dans ce sens, pour disposer d'une plate-forme commune de prototypage, nous avons développé une bibliothèque de classes d'objets parallèles, implantant une forme étendue de la notion d'agrégat concurrent. De plus nous avons démarré depuis peu l'étude d'un système générique permettant la gestion d'environnements virtuels et le développement d'applications de réalité virtuelle.

1- Suite des travaux en synthèse d'images réalistes

1.1- Illumination globale

Actuellement les recherches en illumination globale de l'équipe s'orientent dans plusieurs directions :

La première voie est la poursuite des travaux en cours. Il s'agit d'une part d'améliorer la qualité des images produites par la méthode déjà développée en résolvant les problèmes d'échantillonnage des surfaces par un traitement dynamique adaptant la scène à l'illumination [6]. La prise en compte d'un plus grand nombre de longueurs d'onde débordant du spectre visible permet d'élargir le champ d'application à d'autres domaines (la télédétection, par exemple). La deuxième voie est la conception d'une nouvelle méthode parallèle d'illumination globale qui met en oeuvre une approche radicalement différente dans laquelle l'espace et les objets ont une représentation discrète unique : le voxel. Cet échantillonnage permet de résoudre les problèmes d'aliassage photométrique (ombres). De plus la méthode permet une meilleure répartition des données et un meilleur équilibre des charges. Il est possible de réduire les communications en utilisant la très forte cohérence spatiale des voxels.

1.2- Animation

Sous le thème d'animation, nous nous intéressons aux techniques d'accélération pour le calcul de séquences d'images. Les modèles employés pour gérer le mouvement des objets sont des modèles phénoménologiques (descriptifs). Actuellement, l'étude des méthodes pour exploiter les cohérences spatio-temporelles a abouti à la réalisation d'un lancer de rayons 4D [7] qui permet de calculer toutes les images d'une séquence en même temps, en évitant les calculs redondants.

2- Un environnement de programmation parallèle à objets pour la synthèse d'images

Le développement de l'application de radiosité sur VOXAR à partir de l'application de lancer de rayons a montré la difficulté de réutiliser des composants logiciels dans un contexte parallèle particulier. Les résultats que nous avons obtenus, les nouvelles applications et les nouvelles technologies matérielles nous ont donc conduit à nous intéresser à l'étude d'un environnement de programmation générique permettant de simplifier la parallélisation d'applications graphiques 3D telles que :

- le rendu réaliste par lancer de rayons (étendu aux méthodes d'illumination globale utilisant le lancer de rayons et au calcul de séquences animées) ;
- les algorithmes génétiques appliqués à la synthèse d'images (génération de formes, de textures et de comportements) ;
- les simulations physiques ;
- les applications de réalité virtuelle.

Pour faciliter la distribution des données nous avons introduit une méthode simple utilisant les techniques d'héritage des langages à objets. Des classes d'objets parallèles (généralement appelés agrégats parallèles [8,9]) qui simplifient le travail du programmeur ont été définies. Le système général propose un ensemble de classes génériques gérant des schémas de répartition de données fréquemment utilisés (dans ce type d'applications).

2.1 Les choix techniques

Avant de présenter la structure générale de l'environnement, arrêtons nous sur quelques choix techniques importants :

Une démarche fortement objet semble bien adaptée car nos applications sont intrinsèquement orientées-objet (des objets qui envoient ou reçoivent des sollicitations de ou vers d'autres objets). De plus l'encapsulation dans des classes avec les techniques classiques de la programmation par objets comme la généricité, l'héritage et le polymorphisme semblent être les plus appropriées pour proposer une structure de contrôle des événements génériques.

Une structure de programmation SPMD (Simple Program Multiple Data) a été choisie de façon naturelle car les mêmes phénomènes peuvent se produire en différents points de l'espace.

Un modèle de communication virtuelle semble être le plus approprié pour encapsuler la couche des ressources physiques. L'apparition des Transputers T9000 qui supportent les canaux virtuels de façon matérielle, et l'existence de systèmes d'exploitation comme Chorus nous ont conduit à faire ce choix qui permet en outre une plus grande flexibilité du système de communication. De plus, d'importants travaux ont été menés pour étudier les problèmes d'inter-blocage grâce à des protocoles de communication basés sur des canaux virtuels [10,11].

2.2 Structure générale du système

Nous détaillons maintenant notre système (exposé en [12-14]) en insistant tout particulièrement sur les objets parallèles (cf. figure 2).

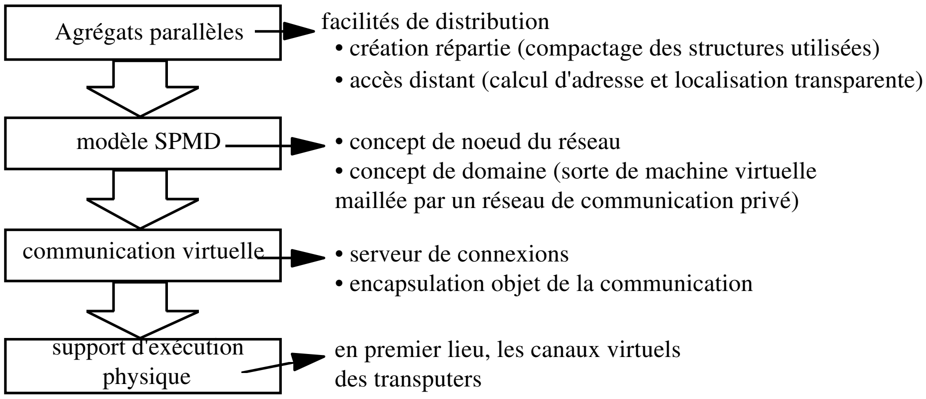

Figure 1 : La structure du système.

La première couche encapsule de manière stricte le contexte d'exécution en assurant l'évolution possible du système depuis la première version sur Transputer vers d'autres plateformes comme PVM, MPI ou Chorus. Au niveau suivant, nous avons introduit un serveur de connexions, présent sur chaque site, qui permet le contrôle dynamique d'un ensemble de connexions. Au niveau supérieur, on trouve la structure de contrôle SPMD qui encapsule le concept de noeud du réseau. Au même niveau nous avons implanté le concept de domaine qui peut être assimilé à une sorte de machine virtuelle. Chaque domaine possède son propre réseau complet de communication. Au niveau le plus haut on trouve les classes d'objets parallèles. Elles encapsulent la répartition des données et permettent donc la conception de systèmes plus indépendants de cette répartition.

Pour définir un comportement parallèle plus spécifique, avec de simples mécanismes d'héritage, nous avons dérivé trois classes d'objets parallèles : les objets répartis, les objets dupliqués et les objets globaux. Ces classes possèdent les caractéristiques suivantes :

• Les données internes des *objets distribués* sont réparties sur le domaine suivant une méthode de répartition virtuelle.
• Les *objets dupliqués* sont une restriction des objets distribués. Leurs données internes sont dupliquées sur chaque noeud du domaine et les changements sont répercutés au reste du domaine de manière transparente.
• Les *objets centralisés* ont leurs données internes centralisées sur un noeud unique et accessibles de tout le domaine. Ce type d'objet parallèle est utile pour implanter des accès à des ressources de manière transparente.

Le travail le plus important est réalisé dynamiquement à la création d'un objet parallèle. L'appel à son constructeur alloue la mémoire nécessaire pour stocker les données locales de l'objet sur le noeud courant (cela avec un coût mémoire minimum). Un réseau de communication privé est créé sur le domaine et un serveur local est lancé pour assurer l'accès distant à partir des autres noeuds. Chaque méthode appelée ensuite sur un objet est filtrée par un processus de contrôle d'accès (en tenant compte de la méthode de répartition interne) qui indique la marche à suivre pour la localisation des données, le calcul et la conversion des adresses et l'accès distant si nécessaire. Nous avons défini un ensemble de classes dérivées des classes d'objets répartis ou dupliqués, à partir de la dimension de la structure de donnée de l'objet (comme les vecteurs répartis, les matrices réparties...).

Ces travaux, menés dans le cadre d'une convention C.I.F.R.E. avec la société ARCHIPEL S.A., ont vu la réalisation d'un premier prototype, écrit en C++, pour réseau de Transputers T800 et une application de rendu réaliste (lancer de rayons énergétique) a été parallélisée sous différentes formes grâce à l'environnement. Ces premiers résultats expérimentaux sont très encourageants et mettent en évidence le gain apporté par une telle approche (moins d'une semaine de développement a été nécessaire pour réaliser chacune des parallélisations).

3- Définition d'un système permettant la gestion d'environnements virtuels

Notre objectif est ici de définir un système générique permettant la gestion d'environnements virtuels et, plus généralement, le développement d'applications de Réalité Virtuelle. Ce système devra gérer en temps réel plusieurs utilisateurs munis de périphériques spécialisés grâce à une plate-forme distribuée.

Dans cette partie nous allons présenter la structure générale d'un environnement virtuel et la structure de l'environnement support.

3.1- Structure de l'environnement virtuel

Un environnement virtuel est composé d'entités, de stimuli et d'un univers virtuel.

• La notion d'*entité* permet de gérer de façon uniforme les décors des mondes virtuels, les objets virtuels mais aussi les "clones" des utilisateurs (objets associés chacun à un utilisateur qui permettent à celui-ci de participer à la simulation comme si il était réellement présent dans l'univers simulé) [15-17]. Les entités sont autonomes et possèdent un ensemble de propriétés et de comportements. Elles sont conceptuellement regroupées en familles (population des entités ayants les mêmes propriétés et les mêmes comportements). Globalement les systèmes simulés sont homogènes (beaucoup d'objets regroupés dans peu de familles).

• Les *stimuli* sont les véhicules des interactions entre les entités.

• *L'univers virtuel* représente le milieu tridimensionnel dans lequel les entités coexistent et interagissent.

Dans cette partie nous allons décrire la structure des entités présentes dans l'environnement virtuel en décrivant le mécanisme de communication inter-entité et les mécanismes associés à la définition et à la prise en compte du comportement de ces entités.

Une entité et le milieu dans lequel elle évolue interagissent de deux façons :

- Un changement du milieu constitue un stimulus, tel qu'un signal visuel ou sonore. L'entité y est sensible, c'est-à-dire qu'elle modifie son activité de façon visible (mouvement par exemple) ou non (modification de son état interne). Le stimulus est reçu par un capteur spécialisé qui va interpréter puis transmettre une information à la partie comportementale de l'entité.
- Après analyse de l'information, la partie comportementale va modifier l'état interne de l'entité et/ou commander une action aux effecteurs. Les effecteurs agissent sur le milieu et en le modifiant créent de nouveaux stimuli.

Nous proposons de modéliser ces interactions grâce à quatre éléments : des capteurs, des effecteurs, des images et des espaces d'images. Cette modélisation concerne tant les interactions internes à l'environnement virtuel que celles qui ont lieu avec le monde réel (pour les échanges utilisateurs-clones d'utilisateurs notamment).

A un certain type de stimuli va correspondre un quadruplet (image, espace d'image, capteur, effecteur). Un capteur va saisir une vue instantanée de l'espace d'image qu'il va filtrer puis interpréter. Par l'exécution d'une action un effecteur produit une image qu'il va placer dans l'espace d'image.

Pour la définition de comportements nous proposons leur découpage en deux composantes : une composante "réflexe" ou sont définis les réflexes de l'entité aux stimuli provenant de son environnement (via les capteurs) et une composante "réflexion" où est définie la partie impliquant une analyse plus complexe.

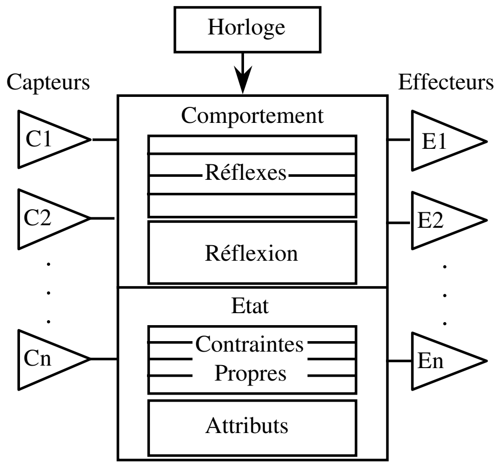

Figure 2 : Schéma structurel d'une entité

La composante "réflexe" est constituée d'un certain nombre de modules assignés chacun à un capteur pour analyser les données renvoyées par celui ci et réagir éventuellement à ces données en commandant, si nécessaire, des actions aux effecteurs. De plus le module réflexe peut déclencher a posteriori la réflexion.

La composante "réflexion" est activée soit par une horloge interne à l'entité, soit par un module réflexe. Le mécanisme de réflexion proprement dit n'est pas détaillé ici car il dépend fortement du type d'entité considéré.

L'entité comporte, de plus, un état : ensemble d'attributs auquel est associé un ensemble de contraintes propres (contraintes entre attributs de l'entité).

3.2 Structure générale de l'environnement support

L'environnement support est la partie du système qui est responsable de la simulation de l'univers virtuel et de sa distribution.

1) La plate-forme distribuée

La plate-forme distribuée est l'ensemble des sites qui participent à la simulation d'un univers virtuel. Elle est découpée logiquement en quatre ensembles de ressources qui sont : les ressources en calcul, les ressources en rendu, les ressources de stockage et les ressources d'entrées/sorties. Chaque site doit être référencé dans l'un au moins de ces ensembles. Une station de travail graphique l'est dans chacun des ensembles par exemple.

Les ressources en rendu correspondent aux sites disposant de processeurs ou de cartes spécialisées à un rendu particulier. Une carte graphique est un bon exemple, ainsi qu'une carte de rendu sonore tridimensionnel. Ces ressources sont généralement connectées à des périphériques sensoriels portés par les utilisateurs : casques stéréophoniques ou casques de visualisation par exemple. Cet ensemble est lui même divisé en sous-ensembles correspondant à chaque type de rendu.

Les ressources d'entrées/sorties sont les sites possédant des ports d'entrée/sortie utilisés par notre système. Cet ensemble est divisé en trois groupes : les ressources d'acquisition qui permettent l'accès aux périphériques de réalité virtuelle (gants et systèmes de localisation notamment), les ressources d'entrées/sorties standards (console) et les ressources de pilotage représentant les commandes de systèmes externes (dispositif de pilotage de robot par exemple). Les ressources d'acquisition et de pilotage sont divisées en sous-ensembles correspondants au divers types de périphériques.

2) Répartition d'une entité sur la plate-forme distribuée

De façon à être facilement répartie sur la plate-forme, chaque entité va être découpée en un certain nombre de parties que l'on nommera sous-entités. Ces sous-entités sont : les sous-entités rendu, les sous-entités entrée/sortie, la sous-entité stockage, la sous-entité comportement et la sous-entité générale.

Chaque sous-entité rendu correspond à l'ensemble des capteurs et des attributs spécifiques à un type de rendu. Chaque sous-entité entrées/sorties correspond à l'ensemble des capteurs d'acquisition ou l'ensemble des effecteurs de pilotage et des attributs spécifiques a un type d'entrée/sortie. La sous-entité stockage correspond à l'ensemble des effecteurs et des attributs spécifiques au stockage. La sous-entité comportement correspond à la réflexion, aux contraintes propres, aux autres capteurs et effecteurs, aux réflexes et aux attributs spécifiques au comportement. La sous-entité générale contient l'ensemble des attributs qui ne sont pas spécifiques à une des sous-entités précédentes.

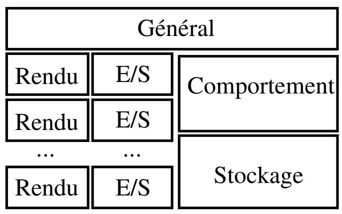

Figure 3 : Découpage d'une entité en sous-entités

Une entité est répartie de la façon suivante : chaque sous-entité est placée sur l'ensemble de ressources correspondant et la sous-entité générale est dupliquée sur tous les ensembles.

3) Parallélisation de l'univers virtuel

La répartition d'une entité, que nous venons de présenter, introduit un parallélisme fonctionnel intra-entité. Un parallélisme plus général de l'univers virtuel peut être introduit. Un univers virtuel est un environnement homogène contenant des entités qui se comportent de façon similaire. Plusieurs parallélismes de donnée sont envisageables ; certains ont été utilisés dans les systèmes existants d'autres sont à étudier. Nous allons ici en présenter succinctement quelques uns.

La parallélisation sur les objets est la parallélisation la plus utilisée car la plus intuitive [18-20]. Son principal avantage est de favoriser un bon équilibre des charges. En effet en donnant à gérer un ensemble d'objets à chaque site, chacun devrait avoir la même charge. L'interaction avec l'utilisateur (via des périphériques de réalité virtuelle) est en général traitée sur des sites particuliers qui peuvent eux aussi gérer des objets.

Cependant, pour la détection des interactions directes (collisions) ou indirectes (échanges d'information dans l'environnement), pour la détection des objets visibles par un utilisateur et pour la résolution des interactions, la communication entre les différents sites entraîne de nombreux problèmes. Nous pouvons cependant noter que deux objets ne peuvent interagir que si ils sont assez proche. L'exploitation de cette propriété de localité des interactions, doit pouvoir résoudre certains de ces problèmes.

Un découpage spatial logique de l'univers virtuel est une solution qui parait intéressante lorsque l'univers virtuel est constitué d'un ensemble de mondes virtuels reliés par des portails (portes, ouvertures, sas...). On peut, en effet, imaginer une parallélisation du modèle consistant à distribuer les mondes virtuels aux différents sites.

Un découpage spatial régulier (bi ou tridimensionnel) peut lui aussi être envisagé. Dans ce cas, on va découper l'espace du monde virtuel en régions carrées (applications pour lesquelles la hauteur importe peu : simulation automobile par exemple) ou cubiques (applications fortement tridimensionnelles : simulation de vol par exemple) de tailles équivalentes. Ensuite on va associer les objets aux régions dans lesquelles ils sont spatialement présents, et enfin les régions aux sites. Ce type de découpage a été utilisé avec succès pour VOXAR dans le cadre de la synthèse d'images réalistes.

Ces deux dernières méthodes apportent un modèle de calcul local, car tout ce qui passe dans une partie de l'espace est traité localement sur le site associé. Le site s'occupera, notamment, du comportement de ses objets et des interactions (prise en compte des collisions...) qui peuvent se produire dans cette région de l'espace.

Le choix d'une parallélisation est difficile à faire. En fait ce choix dépend beaucoup de l'application visée. Pour un simulateur de vol, par exemple, le choix risque de porter sur un découpage spatial régulier, alors que pour un système de visite de bâtiments virtuels il semble plus à propos de choisir un découpage spatial logique, en élevant chaque pièce du bâtiment au rang de monde virtuel et en répartissant les pièces aux processeurs.

Conclusion et perspectives

Une première version du système de gestion d'environnement virtuel est en cours de développement sur un réseau de stations de travail.

Cette mise en oeuvre va utiliser les objets parallèles détaillés dans la partie 2. Des objets centralisés seront utilisés pour encapsuler les différents périphériques de réalité virtuelle (Capteur de position Polhemus 3Space Fastrak, Casque de visualisation Virtual Research Eyegen 3 et Gant de donnée). Des objets distribués vont permettre d'étudier plusieurs types de répartition de l'environnement virtuel. Enfin, les Espaces d'Images vont être, dans un premier temps, implantés par des objets dupliqués.

Cette première version servira de maquette pour la réalisation d'un prototype sur un calculateur de type réseau de processeurs puissants à mémoire locale, élément central d'une architecture matérielle dans laquelle vont s'intégrer des périphériques matériels spécialisés tel que : des outils de réalité virtuelle ou bien encore des cartes graphiques spécialisées pour piloter des dispositifs d'acquisition ou de sortie.

Références bibliographiques

[1] R. Caubet, Y. Duthen, V. Gaildrat-Inguimbert "A tridimentional architecture for fast realistic image synthesis." In Proccedings of Computer Graphics International, Geneva May 1988, pp. 135-149.

[2] P. Pitot "The Voxar Project" in IEEE Computer Graphics and Applications, january 1993.

[3] H. Maurel, B. Moisan, J.P. Jessel, R. Caubet, Y. Duthen "Dynamic scenes management for animation with ray-tracing rendering." In Proccedings of Computer Animation, Geneva 1991.

[4] J.P. Jessel, R. Caubet "The implementation of an extended radiosity on the VOXAR machine" In Proccedings of Transputer 92, Arc et Senans, Mai 92.

[5] C. Metge, J.P. Jessel, R. Caubet "Une radiosité parallèle utilisant un lancer de rayons et un découpage spatial : première évaluation" Actes de MICAD 93, 1993

[6] M. Paulin, J.P. Jessel "Adaptive mesh generation for progressive radiosity : A ray-tracing based algorithn." accepted to EUROGRAPHICS'94, September 1994.

[7] H. Maurel, Y. Duthen, R. Caubet "A 4D Ray Tracing" Computer Graphics forum, proceedings of EUROGRAPHICS'93, 6-10 September 1993.

[8] J.M. Jézéquel "Programming Massively Parallel Architectures with Sequential Object-Oriented Languages" Lecture notes in computer science PARLE'92, Parallel architectures and languages Europe, pp 329-344.

[9] A. A. Chien, W. J. Dally "Concurrent aggregates (ca)." In Proccedings of the Second ACM Sigplan Symposium on Principles and Practice of Parallel Programming, 1991.

[10] L. Kleinrock "Queuing Systems" Volume 2: Computer Applications, Wiley, New York, 1976, pp. 438-440.

[11] W. J. Dally "A VLSI Architecture for Concurrent Data Structures", 5.3.2 Deadlock-Free Routing, Kluwer Academic Publishers, pp. 161-169.

[12] B. Moisan, Y. Duthen, R. Caubet "an Object-oriented Concurrent Programming Environment for simulation applications on distributed memory processor networks" IEEE EUROMICRO workshop 93 on Parallel and Distributed Systems, Gran-Canarya, January 27-29.

[13] B. Moisan, Y. Duthen, R. Caubet "Tools for SPMD object-oriented programming" EUROMICRO'93 Barcelona 6-10 Sept 93.

[14] B. Moisan "Un modèle de programmation parallèle à objets appliqué à la synthèse d’images" Thèse de l’université Paul Sabatier de Toulouse soutenue le 17 Décembre 1993.

[15] S. Rainjonneau "Un Modèle Orienté Objet pour la Simulation Comportementale" Thèse de l'Université Paul Sabatier soutenue le 16 Décembre 1992.

[16] Y. Duthen "Les projets VOXAR (VOXel ARchitecture) et In VitrAm (In Vitro Animats) : Synthèse des travaux" Habilitation à Diriger des Recherches, 4 Mars 1993.

[17] R. Mouli, Y. Duthen, R. Caubet "In VitrAm (In Vitro Animats, a behavioural simulation model)" 2nd IEEE International Workshop RO-MAN'93, 3-5 November 1993.

[18] R. Kazman "Making WAVES: On the Design of Architectures for Low-end Distributed Virtual Environments", Proceedings of the First IEEE Conference on Virtual Reality Technology, 1993.

[19] D.N. Snowdon, A.J. West, T.L.J. Howard "Towards the next generation of human-computer interface", L'interface des mondes réels & virtuels, Informatique 93, Mars 1993.

[20] C. Grimsdale, S. Ghee "Virtual realities - artificial environments" Computer Graphics 90, November 1990.