OUTILS DE MODELISATION EN ENVIRONNEMENT VIRTUEL.

Patrice TORGUET, Frédéric RUBIO,

Véronique GAILDRAT, René CAUBET

Institut de Recherche en Informatique de Toulouse

118 Route de Narbonne

31062 Toulouse Cedex, France
Tél : (33) 61 55 63 13, Fax : (33) 61 55 62 58
E-mail : {torguet|rubio}@irit.fr

Résumé :

La réalité virtuelle distribuée, grâce aux nouveaux paradigmes d'interaction qu'elle propose, offre d'intéressantes perspectives pour la modélisation interactive d'objets et de scènes. Nous nous intéressons plus particulièrement aux applications qui permettent à plusieurs utilisateurs d'éditer et de déformer des formes préexistantes. Cet article présente une première réalisation s'appuyant sur une plate-forme générique de développement d'environnements virtuels multi-utilisateurs : VIPER.

Nous proposons de nouveaux outils de manipulation qui mettent en œuvre les déformations de formes libres (FFD) dans un environnement virtuel.

***Mots clés** : réalité virtuelle, modélisation interactive, déformations de forme libre.*

1. Introduction.

Les systèmes de réalité virtuelle se limitaient à leur début à des visites de lieux imaginaires ou reconstitués dont les éléments constituants n'étaient qu'une représentation symbolique d'objets plus complexes (un cube pour une maison, …). Lorsque les concepts de base de la réalité virtuelle ont été plus établis, les concepteurs ont pu se consacrer davantage aux problèmes d'interactivité de leurs systèmes (aspects jusqu'ici moins développés).

L'innovation apportée par ces nouveaux systèmes ne compensait plus le manque de convivialité et d'ergonomie des premières réalisations. Aussi, on a pu constater une nette amélioration des interfaces proposées grâce aux progrès de la technologie (puissance des machines, nouveaux périphériques, …) et grâce à des études approfondies sur l'interactivité des systèmes de réalité virtuelle. Ce sont ces aspects qui nous intéressent plus particulièrement dans le cadre de ces travaux. Nous pensons que la modélisation interactive d'objets peut largement bénéficier des apports de la réalité virtuelle distribuée en terme d'ergonomie, de convivialité et d'efficacité.

Nous présentons une mise en oeuvre de ce principe s'appuyant sur une plate-forme générique de développement d'environnements virtuels multi-utilisateurs : VIPER [9].

2. Le cadre du projet.

La réalité virtuelle, grâce aux nouveaux périphériques qu'elle propose, offre de nouvelles perspectives pour l'ergonomie de la modélisation, en introduisant de nouveaux concepts d'interaction. Nous souhaitons introduire des métaphores de haut niveau dans le domaine de la modélisation interactive de formes. Cette idée trouve son origine dans l'étude des systèmes de modélisation en environnement virtuel. Nous avons constaté que les modèles d'interaction proposés sont assez contraignants ou bien inexistants [1] [4].

Nous proposons donc de fournir à l'utilisateur un ensemble d'outils lui permettant d'interagir dans un environnement virtuel de la manière la plus naturelle qui soit : réaliser lui-même les opérations de modelage qu'il désire effectuer sur un objet en reproduisant des gestes qu'il effectuerait naturellement dans une situation réelle (i.e. marteler un objet, creuser une cavité, …). L'utilisateur dispose ainsi d'une véritable boîte à outils qui est fournie en fonction de l'application. Nous nous intéressons plus particulièrement aux applications qui permettent l'édition et la déformation de formes préexistantes dans un environnement collecticiel [10].

Nous avons défini deux types d'usagers qui peuvent évoluer dans l'espace de modélisation. Chacun est doté d'une forme géométrique (main) et d'un point de vue. De plus on lui assigne un rôle spécifique [8]. La première catégorie d'utilisateurs regroupe les **visiteurs**. Un visiteur est un participant passif de la simulation. Il peut se déplacer, visiter les différentes pièces, mais il ne peut pas intervenir dans une tâche de modélisation. Toute personne qui rejoint une session en cours reçoit les droits d'un visiteur. Le second type d'usager définit les **sculpteurs**. Un sculpteur participe activement à la modélisation. Il déforme les objets par l'intermédiaire de l'outil qu'il détient (outils qui seront décrits par la suite). A tout instant, un visiteur peut devenir un sculpteur. Il lui suffit de *saisir* un outil. Ces outils sont actuellement présentés sur des tables disséminées dans le monde virtuel.

Notre objectif est de rendre la tâche de modélisation plus intuitive. Ainsi nos outils ont un rôle double : leur forme illustre non seulement leur fonction (marteau, pince, …) mais aussi la façon de les utiliser (taper avec le marteau, …). La métaphore du sculpteur est pleinement réalisée : il y a correspondance directe entre le geste effectué par l'utilisateur dans le monde réel et l'action réalisée par son clone dans le monde virtuel.

3. Modèle de déformation.

Nous allons maintenant présenter le modèle mathématique associé à chacun de nos outils de déformation.

Il existe deux grandes catégories d'opérateurs dans les techniques de modélisation :

• la déformation par manipulation directe : l'utilisateur transforme l'objet en modifiant directement sa structure géométrique. C'est l'approche choisie par Delingette et al. [3] qui déforment des structures osseuses définies par des maillages simplexes en appliquant des forces sur les noeuds du réseau.

• la déformation par manipulation indirecte : cette catégorie d'opérateurs est totalement indépendante des modèles géométriques. L'utilisateur n'agit plus directement sur la forme de l'objet. Il passe par le biais d'une entité de déformation qui l'englobe. Il s'agit pour l'utilisateur de déplacer les points de contrôle définissant le volume englobant (treillis). C'est ce type de déformation que nous avons choisi notamment en raison de son indépendance de tout modèle géométrique et de sa souplesse d'utilisation.

Le modèle que nous avons retenu est plus connu sous le nom de FFD (Free-Form Deformation) [7]. Déformer par FFD consiste à enfermer l'objet (ou une partie de celui-ci) dans un bloc de plastique souple. Dès lors, les déformations ne sont plus appliquées directement à l'objet mais à l'espace qui l'englobe. Les transformations subies par l'objet ne sont qu'une interpolation de la déformation du volume englobant. Les FFD ont fait l'objet de nombreux articles. Aussi, nous ne rappellerons que brièvement leurs fondements mathématiques.

Le volume englobant un objet est un parallélépipède qui définit une partition de l'espace en deux sous-ensembles : les points intérieurs et les points extérieurs. Ces derniers ne seront pas affectés par le processus de déformation qui se décompose en plusieurs étapes.

Il s'agit tout d'abord de définir le volume englobant en indiquant son origine, ses axes directeurs et le nombre de points de contrôle selon chacun de ces axes. On crée ainsi un treillis de points dont les coordonnées sont exprimées dans le repère propre au treillis (Fig. 1). Dans une seconde phase, il faut calculer les nouvelles coordonnées des points de l'objet dans le repère local.

A partir de ce moment, toute déformation du treillis (déplacement d'un ou de plusieurs points de contrôle) peut être répercutée à l'objet par une interpolation de type Bézier ou B-Spline. L'exemple suivant illustre une interpolation à base de polynômes de Bernstein :

$$P_{ffd} = \sum_{i=0}^{l} \binom{l}{i} (1-s)^{l-i} s^{i} \left[\sum_{j=0}^{m} \binom{m}{j} (1-t)^{m-j} t^{j} \left[\sum_{k=0}^{n} \binom{n}{k} (1-u)^{n-k} u^{k} P'_{ijk} \right] \right]$$

où (s, t, u) sont les coordonnées locales du point traité,
et P'_{ijk} sont les points de contrôle après déplacement.

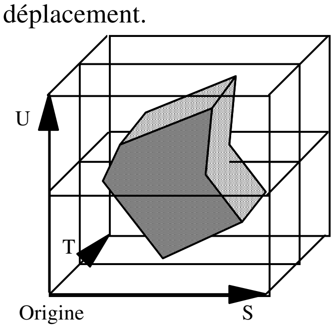

Figure 1 : Un exemple de treillis.

Cet outil illustre parfaitement la métaphore du sculpteur : du point de vue de l'utilisateur, il suffit de modeler le bloc de plastique pour déformer l'objet, ce qui est très intuitif. Dans la réalité, on s'aperçoit qu'il n'est pas toujours facile d'obtenir certains effets (comme aplatir une bosse). Nous avons donc été amenés à concevoir un système qui masquerait totalement la notion de fonctions mathématiques pour la modélisation d'objets.

Nous proposons à l'utilisateur un ensemble d'outils mettant en oeuvre ce type de déformation [6]. Ces outils ne nécessitent que des paramètres ayant une signification réelle (partie de l'objet qui sera modifiée, taille de la tête d'un marteau, …) et communiqués de la manière la plus naturelle (désignation graphique, choix du marteau, …). On peut les considérer comme une interface entre l'utilisateur et le système de modélisation.

Dans cette optique, la réalité virtuelle nous fournit la possibilité de mettre en œuvre ces concepts ; ainsi à un outil est associé un geste qu'il est facile d'analyser : une main qui tient un marteau et qui *tape* avec force une surface est un événement facilement interprétable (du moins dans le contexte d'un éditeur de formes).

Néanmoins, la réalité virtuelle impose des temps de réponse extrêmement brefs. Aussi, nous avons défini des treillis aussi simples que possible. La hauteur et la largeur du treillis dépendent du rayon d'action de l'outil. La profondeur est calculée en fonction de la rigidité de l'objet. Précisons ici que cette rigidité n'a rien de physique mais elle permet de varier les effets. Ainsi pour un objet donné et un type d'outils, nous créons un treillis.

Nous allons maintenant présenter brièvement un de nos outils : le "crochet". Cet outil nous permet de déformer un objet à volonté. Il se présente sous la forme d'un stylet doté d'un crochet à l'une de ces extrémités. Dès que le crochet touche la surface d'un objet, nous créons un treillis. A chaque déplacement de l'outil, nous calculons la déformation à appliquer au treillis (Fig. 2). L'étendue de cette déformation sur la surface dépend des caractéristiques définissant le crochet. Puis nous déformons la partie de l'objet affectée. L'utilisateur a ainsi l'impression que la surface de l'objet s'est "collée" à l'outil et qu'elle suit tous ses déplacements. Enfin, lorsque l'utilisateur lache le stylet, celui-ci se décolle de l'objet.

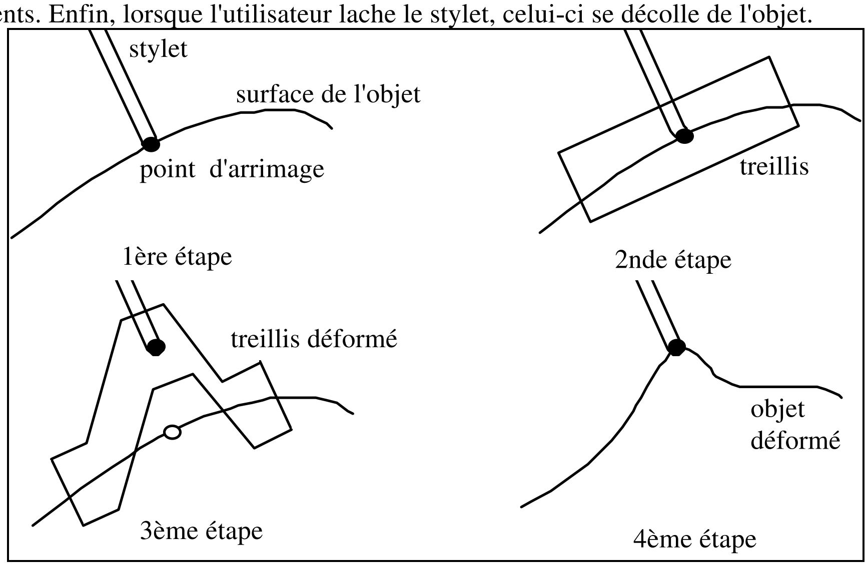

Figure 2 : Déformation par crochet

4. VIPER.

Après la présentation du modèle de déformation, nous allons maintenant décrire la plate-forme logicielle qui va permettre la réalisation de l'application de modélisation en environnement virtuel.

VIPER (VIrtuality Programming EnviRonment) [9] est une plate-forme générique permettant la gestion d'environnements virtuels multi-utilisateurs et, plus généralement, le développement d'applications de Réalité Virtuelle collectives. Ce système gère en temps réel plusieurs utilisateurs munis de périphériques spécialisés grâce à une architecture distribuée.

4.1. Structure des applications cibles

Notre but est de proposer un modèle de calcul distribué générique permettant de gérer n'importe quelle application pouvant être modélisée en terme d'échanges (symbolisés par des **stimuli**) entre des **entités**, dans un **univers virtuel** (Fig. 3).

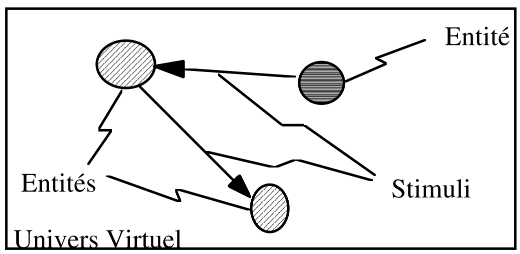

Figure 3 : Structure d'un environnement virtuel.

La notion d'entité nous permet de gérer de façon uniforme les décors des mondes virtuels, les objets virtuels mais aussi des clones (objet associé à un utilisateur [5], à une application [2] ou à un robot, qui leur permet de participer à la simulation comme s'ils étaient réellement présents dans l'univers virtuel). Les entités sont autonomes et possèdent un ensemble d'attributs et de comportements.

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

Les interactions entre les entités, modélisées par des **stimuli** (phénomènes ou événements perceptibles par une entité), transitent par des milieux, appelés **espaces de stimuli**, qui permettent des communications entre plusieurs entités simultanément. Chaque espace de stimuli est une projection de l'univers virtuel selon un axe propre à un type de stimulus (espaces des formes graphiques, espaces des sons, ...). Une entité reçoit les stimuli, qu'elle est capable de percevoir (formes visibles par l'entité, sons proches, ...), grâce à des **capteurs** et peut agir sur cet environnement grâce à des **effecteurs** (Fig. 4).

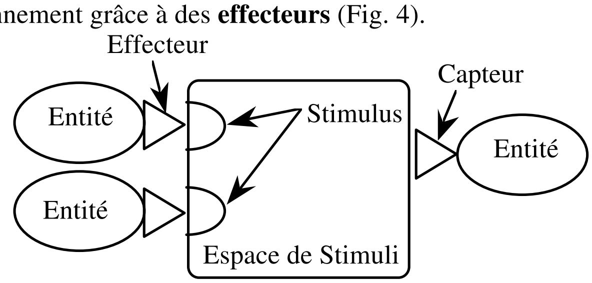

Figure 4 : Mécanisme de communication inter-entités.

Une entité est dotée d'un certain nombre de comportements qui modifient son état interne (ensemble de ses attributs) et commande des actions à ses effecteurs.

Les comportements d'une entité sont de deux types :

- les comportements **réflexes** modélisent les réactions de l'entité aux stimuli provenant de son environnement via les capteurs. On peut citer en exemple de réflexe, la déformation d'un objet suite à un coup de marteau.

- les **raisonnements** modélisent les actions nécessitant une analyse plus complexe et un caractère plus volontaire. Un raisonnement commande des actions en fonction de la connaissance de l'environnement, d'une mémoire et d'un mécanisme de réflexion qui peut être modélisé de diverses façons (automate à états finis [2], moteur d'inférence PROLOG [5], ...).

4.2. L'architecture logicielle de VIPER

Une approche orientée-objet nous a semblé bien adaptée compte tenu de la structure des applications qui est, intrinsèquement, elle-même fortement orientée-objet (les entités envoient/reçoivent des stimuli vers/depuis d'autres entités). Les avantages classiques des langages à objets (encapsulation, réutilisabilité...) et plus particulièrement l'héritage semblent des plus intéressants. Le choix du langage C++ est justifié par sa large diffusion (C++ existe en effet sur la plupart des architectures matérielles) et sa reconnaissance comme standard de facto en programmation orientée-objet.

Pour gérer simplement les aspects distribués des applications cibles, nous avons choisi d'offrir au développeur un ensemble de classes génériques qui lui permettent de créer et d'utiliser des agrégats concurrents. Un agrégat concurrent encapsule la distribution de ses données et les mécanismes gérant les accès distants éventuels. Des classes gérant divers modes de distribution ont été implantées (réplication active, répartition, …).

L'architecture met à la disposition des développeurs deux niveaux de programmation. Le premier niveau masque totalement l'aspect distribué de l'application qui apparaît ainsi totalement séquentielle. A ce niveau, le développeur définit des entités (comportements et attributs), des capteurs, des effecteurs et des stimuli spécifiques à l'application considérée en spécialisant des classes existantes. La gestion distribuée de l'application est totalement prise en compte par les espaces de stimuli et les univers virtuels qui sont, respectivement, des agrégats concurrents de stimuli et d'entités. Le second niveau permet de redéfinir les mécanismes de distribution. Ceci est réalisé par héritage et spécialisation de classes préexistantes d'agrégats concurrents. Le programmeur est ainsi libre d'optimiser l'application à sa convenance [9].

5. Mise en oeuvre dans le cadre d'un éditeur collecticiel de formes

Nous allons décrire l'application en nous appuyant sur les deux niveaux de programmation définis dans le paragraphe précédent. L'introduction de la notion d'outil dans VIPER correspond au premier niveau. Alors que la description des aspects distribués met en oeuvre le second niveau de développement.

5.1. Introduction de la notion d'outils dans VIPER

Un outil de modélisation dans le modèle VIPER est un type particulier d'entité manipulée par une ou plusieurs entités et qui, à son tour, manipule une ou plusieurs entités. Ces manipulations sont modélisées par des échanges d'ordres (nouvelle classe de stimuli) dans un espace de stimuli spécifique appelé espace d'ordres (Fig. 3).

Un outil possède au moins un capteur qui récupère les ordres qui lui sont transmis ; ainsi qu'un effecteur d'ordres qui envoie une commande de manipulation aux entités contrôlées par l'outil.

Nous allons illustrer ces principes par la mise en œuvre d'un *coup de marteau* (Fig. 5).

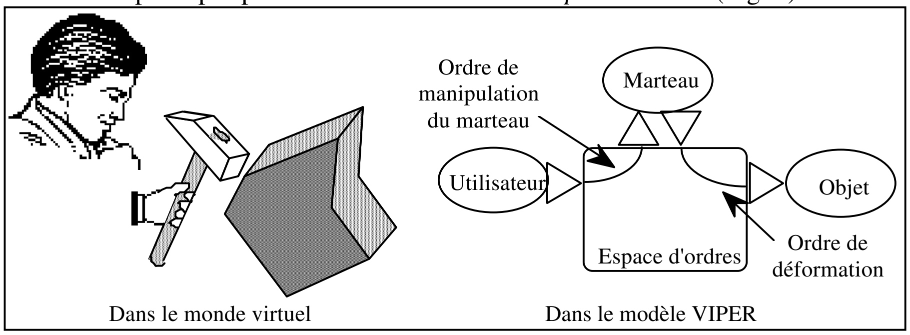

Figure 5 : Le coup de marteau.

Dans cet exemple, l'entité *clone de l'utilisateur* contrôle l'entité *marteau* en lui envoyant des ordres de manipulation. Un tel ordre est composé de l'identificateur de l'outil, du label *déplacement* et de la dernière position de la main. L'entité marteau analyse ces données pour déterminer si la tête du marteau heurte effectivement un objet. Si une collision est détectée, l'entité marteau envoie un ordre de manipulation à l'objet touché. Cet ordre comprend l'identificateur de l'objet, le type de l'interaction (**martèlement**), les deux dernières positions de la tête du marteau ainsi que son diamètre. Dès la réception de cet ordre, l'entité objet crée un treillis, le modifie conformément aux paramètres (type de déformation, positions, diamètre) et calcule sa nouvelle forme.

5.2 Aspects distribués de l'application

Nous allons, ici, montrer comment certains types d'espaces de stimuli distribués peuvent résoudre les problèmes soulevés par les aspects collecticiels des applications envisagées dans le cadre de ce document.

Le premier problème apparaît dès que plusieurs visiteurs évoluent ensemble dans l'espace de modélisation. Un visiteur doit avoir conscience de la présence des autres utilisateurs. Pour résoudre ce problème on utilise un type particulier d'espace de stimuli distribué, appelé espace dupliqué. Chaque entité est pourvue d'une **forme** contenant : sa position, son orientation et une référence à sa géométrie (ensemble de facettes triangulaires représentant l'enveloppe de l'objet). Un espace de stimuli dupliqué contient toutes les formes. Ainsi, lors de la modification de l'une d'elles (déplacement, rotation ou déformation de l'objet), l'espace de stimuli maintient la cohérence entre les différentes copies. Par la suite, chaque entité disposant de capteurs de formes a accès à la dernière version.

La présence d'utilisateurs de type sculpteur crée un nouveau problème : l'entité effectrice agissant sur une entité peut être située sur un autre site. Dans un contexte distribué, un espace de stimuli dupliqué pourrait résoudre ce problème. On peut néanmoins remarquer qu'un ordre donné ne concerne que l'entité manipulatrice et les entités manipulées. Il n'est donc pas nécessaire de le communiquer à tous les sites prenant part à la simulation. L'ajout du numéro de site de l'entité dans son identificateur permet de limiter la transmission d'un ordre aux sites intéressés (Fig. 6).

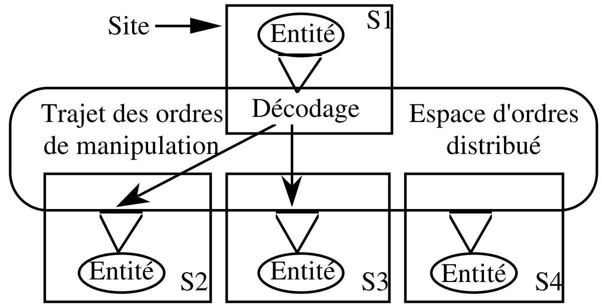

Figure 6 : L'espace d'ordres distribué

Les aspects multi-utilisateurs entraînent d'autres problèmes liés aux conflits risquant de survenir lorsque deux utilisateurs tentent de réaliser des manipulations antagonistes. Ces conflits peuvent intervenir à deux niveaux correspondant à la granularité de l'information manipulée.

Si l'unité de manipulation est l'objet, deux sculpteurs devront obligatoirement travailler sur des objets différents. Dans ce cas, nous proposons la pose explicite d'un verrou graphique sur l'objet. Ce verrou devient alors un outil qui ne peut être manipulé que par un seul utilisateur qui va le saisir, l'amener sur un objet et le lâcher (*drag and drop* des interfaces 2D). L'objet sera désormais verrouillé (Fig. 7) jusqu'à ce que ce même utilisateur retire le verrou.

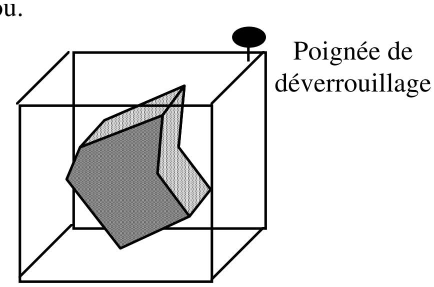

Figure 7 : Un verrou posé sur un objet

Si l'unité de manipulation est une sous-partie de l'objet (objet défini par une hiérarchie de formes géométriques) nous proposons la pose d'un verrou sur cette sous-partie.

Si aucun verrou n'est placé, on autorise deux manipulations sur un même objet en vérifiant qu'il n'y ait pas de recouvrement spatio-temporel des différents treillis mis en jeu. Ce concept est intéressant car il nous permettra à terme de faire évoluer notre application vers la collaboration de plusieurs sculpteurs à une même déformation (torsion de barres, aplatissement, étirements...).

5. Conclusion.

Actuellement, nous développons de nouveaux outils de modélisation (générateur d'outils, …). Nous étudions aussi les possibilités d'améliorer notre interface (zoom, sélection d'un objet, intégration de contraintes).

Nous avons illustré les idées présentées dans cet article en réalisant une application de modélisation de terrain. Nous fournissons aux utilisateurs la matière première (terrain), des outils et des éléments (présentés sur une table) pour aménager le paysage qu'ils souhaitent créer. Les utilisateurs creusent des lacs et des rivières ou élèvent des montagnes. Ils peuvent aussi disposer les éléments du décor (arbres, maisons, …). La présence de plusieurs utilisateurs permet d'assigner à chacun une tache bien précise (responsable des vallées, responsable des lacs, …) (Fig. 8).

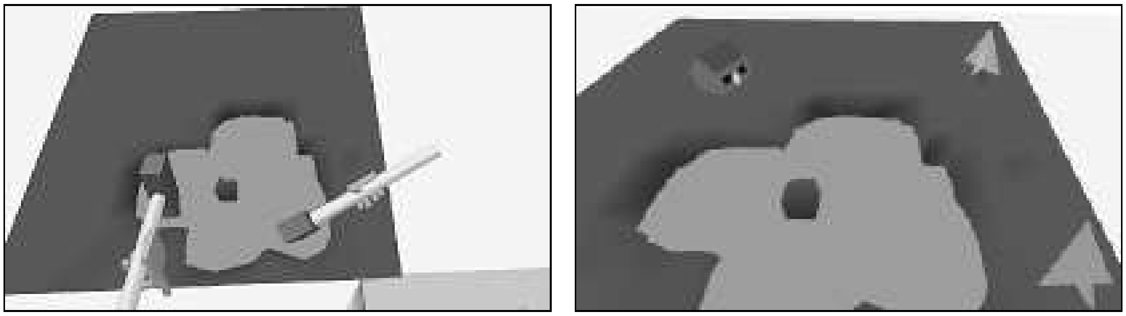

Figure 8 : Modélisation de paysages.

Enfin, nous étudions aussi la possibilité de réaliser des applications dans d'autres domaines d'activité.

6. Bibliographie.

[1] Butterworth J., Davidson S., Hench S. and Olano T.M., "3DM: A Three Dimensional Modeler Using a Head-Mounted Display",
Proc. 1992 Symposium on Interactive 3D Graphics, Cambridge, Massachusetts, March 29-April 1 1992, p. 135-138.

[2] Carlsson C. and Hagsand O. "DIVE - a Platform for Multi-User Virtual Environments", Computer & Graphics, Vol. 17, N° 6, November/December 1993, p. 663-669.

[3] Delinguette H., Subsol G., Cotin S., Pignon J. «A Craniofacial Surgery Simulation Testbed», Visualization for Biomedical Computing (VBC'94) Rochester, USA October 1994.

[4] Liang J., Greeen M., "Geometric Modeling Using Six Degrees of Freedom Input Devices", 3rd International Conference on CAD and Computer Graphics Proceedings, Beijing, China, August 1993, p. 217-222.

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

[6] Rubio F., Torguet P., Gaildrat V., Caubet R., "Déformations de formes libres à l'aide d'outils de réalité virtuelle",
L'Interface des Mondes Réels et Virtuels, Montpellier, Juin 1995, p. 139-149.

[7] Sederberg G T. W., Parry S.R., "Free-Form Deformation of Solid Geometric Models", Siggraph 86 Conference Proceedings, Vol 20, No 14, p. 151-160.

[8] "Rapport de l'atelier "collecticiel"
Comptes rendus des ateliers d'IHM'93, Lyon, Octobre 1993.

[9] Torguet P., Caubet R., "VIPER (VIrtuality Programming EnviRonment): A virtual reality applications design platform",
Actes du 2nd Eurographics Workshop on Virtual Environments, Janvier 1995.

[10] Torguet P., Rubio F., Caubet R., "Atelier de sculpture virtuelle multi-utilisateurs", Actes de Interaction Homme Machine 95, Toulouse, Octobre 1995.