Déformations de formes libres à l'aide d'outils de Réalité Virtuelle

Rubio Frédéric, Torguet Patrice, Gaildrat Véronique, Caubet René

I.R.I.T. Université Paul Sabatier,
118 Route de Narbonne, 31062 Toulouse Cedex, France
Tél : (33) 61 55 63 13, Fax : (33) 61 55 62 58
e-mail : {rubio | torguet}@irit.fr

Résumé : Les techniques de modélisation actuelles ne sont plus adaptées aux besoins des utilisateurs. Les modalités d'interaction n'ont pas suivi l'évolution des fonctionnalités des modeleurs classiques.

Or la réalité virtuelle offre des perspectives intéressantes pour la modélisation qui n'ont pas encore été totalement exploitées.

Cet article présente une nouvelle technique de déformation d'objets et 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. Parmi ces outils, l'utilisateur trouvera un marteau avec lequel il pourra "taper" sur des objets.

Mots clés : réalité virtuelle, modélisation, interaction, modélisation déclarative, déformations de forme libre.

Abstract: Nowadays, modelling techniques are no longer well suited compared with user needs. In fact, classical modeller fonctionalities have been more investigated than their interaction modalities.

On the other hand, we think that, virtual reality applied to object modelling can be further investigated.

In this paper, a new deformation technique is exposed. and a first implementation based on a generic virtual environment design platform: VIPER.

New deformation tools based on VFFD (Virtual Free-Form Deformation) are proposed. Among these tools, will be found a hammer used to bump objects.

Keywords: virtual environments, modelling, interaction, declarative modelling, free-form deformations.

1. Introduction

Les premiers modeleurs ont été développés autour du modèle géométrique utilisé (CSG, surfaces de Bézier, surfaces B-Splines, …).

Bien souvent, déformer un objet se limitait à modifier directement les paramètres définissant le modèle géométrique. Ainsi, l'utilisateur déplaçait les points de contrôle d'une surface, retouchait le profil d'une courbe d'une surface de révolution …

Ces modeleurs nécessitaient une connaissance approfondie des propriétés mathématiques du modèle pour obtenir exactement la forme désirée.

Puis les domaines d'application se sont diversifiés et l'emploi des modeleurs s'est généralisé. Ces nouveaux utilisateurs n'étaient pas des spécialistes de la modélisation géométrique d'objets. Les concepteurs de modeleurs ont dû répondre à de nouvelles attentes, de nouveaux besoins (outils plus puissants et moins dépendants du modèle géométrique).

Ce sont ces idées qui prédominent depuis quelques années dans l'étude et le développement de nouveaux modeleurs.

La réalité virtuelle, grâce aux nouveaux paradigmes d'interaction qu'elle propose, offre d'intéressantes perspectives pour la modélisation interactive d'objets.

Cet article présente une nouvelle technique de déformation d'objets utilisant des outils virtuels. Nous décrivons ensuite une première mise en oeuvre de ce principe s'appuyant sur une plate-forme générique de développement d'environnements virtuels multi-utilisateurs : VIPER.

2. Techniques d'interaction pour la modélisation d'objets

Nous pouvons distinguer deux axes principaux dans l'évolution des modeleurs : les fonctionnalités offertes aux utilisateurs et les techniques d'interaction permettant l'accès à ces fonctionnalités.

La plupart des recherches se sont concentrées jusqu'à présent sur le développement du modeleur en terme de puissance (diversification des modèles géométriques et variétés des opérateurs proposés). Et, comme nous allons le voir, les interfaces de ces nouvelles techniques de manipulation offertes aux utilisateurs dépendent bien souvent de l'outil mathématique mis en œuvre.

Ainsi donc, ces études ont permis l'émergence de nouvelles techniques de modélisation.

Ces techniques, peu précises d'un point de vue "numérique", sont plus intuitives et certainement plus ergonomiques. Elles peuvent se regrouper en deux catégories :

• la déformation par manipulation directe.

L'opérateur transforme l'objet en modifiant directement sa structure géométrique par l'intermédiaire de fonctions. Le rôle de l'utilisateur varie selon les techniques.

Dans le cas de formes polyédriques [KAU 91] [KEN 92], il suffit à l'utilisateur d'indiquer les objets à fusionner et le degré de transformation. L'interaction est ici très limitée puisque les possibilités de l'utilisateur sont restreintes.

Les surfaces B-Splines hiérarchiques [FOR 88] illustrent bien la notion de manipulation directe : l'utilisateur déplace lui-même les points de contrôle définissant la surface avec la possibilité de raffiner le polygone de contrôle pour des déformations plus localisées. Or comme l'a déjà fait remarquer [HSU 92], il est difficile de positionner des points de contrôle pour obtenir une forme particulière. [BLA 92] résoud ce problème en proposant des opérateurs normalisés (opérateur "creux" par exemple) pour les surfaces paramétriques : l'usager choisit l'opérateur, désigne la zone d'application et spécifie les différents paramètres.

Dans le "ThingWorld Modeling System" [PEN 90], les objets sont déformés par l'application de "forces virtuelles". Ce système propose deux niveaux d'interaction. L'utilisateur définit l'allure générale de l'objet à partir d'une forme initiale. Il peut ensuite éditer certaines parties via une carte d'élévation qui sera reportée sur la région manipulée. Cette méthode permet en outre de reproduire un objet à partir de données numériques. Les résultats présentés (en temps de réalisation et en qualité) sont remarquables. Néanmoins, l'emploi d'ascenseurs graphiques (slider) et d'une carte d'élévation semble peu intuitif.

[GAL 91] présente un système original de modélisation interactive. A l'aide d'un périphérique d'entrée 3D, l'utilisateur construit des objets. Il dispose d'outils qui lui permettent d'ajouter de la matière, d'en supprimer, de poncer la surface d'un objet, … Selon les auteurs, il est difficile de réaliser des opérations précises. Le modèle géométrique sous-jacent (voxels), s'il est bien adapté à ce type d'application, pose de sérieux problèmes d'affichage tant les données manipulées sont volumineuses.

• 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 la surface du treillis.

C'est notamment dans cette catégorie que l'on retrouve les FFD [SED 86] où cette entité de déformation est un parallélépipède (treillis). Les déformations subies par le treillis sont ensuite appliquées à l'objet par différents types d'interpolation.

[COQ 90] a étendu ce principe en définissant des treillis de forme arbitraire (Extended Free-Form Deformation) ou bien encore en utilisant un axe comme entité de référence [LAZ 92]. Ces treillis, plus proches de la forme réelle de l'objet, assurent des résultats plus prévisibles.

Pour sa part, Hsu a défini des outils de manipulation sur ces treillis [HSU 92]. L'utilisateur déplace un (des) point(s) de la surface de l'objet. Ces points ont la particularité d'être liés à un point de contrôle de la surface du treillis. A partir de ce déplacement, un treillis reproduisant ces positions est calculé, puis appliqué à l'objet. Cette déformation laisse invariants les points déplacés par l'utilisateur mais lisse la surface de l'objet au voisinage de ces points.

L'apport de ces nouvelles techniques de modélisation en matière de facilité d'utilisation et d'interactivité est indéniable. Elles ont permis d'importants progrès. Néanmoins elles présentent certaines lacunes.

La principale caractéristique des techniques d'interaction à base de manipulation directe est leur forte dépendance du modèle géométrique sous-jacent. Même si des efforts ont été faits pour affranchir l'utilisateur des contraintes liées au modèle employé, ce modèle reste omniprésent et impose ses propres limites.

Le reproche que l'on peut faire aux techniques de manipulation indirecte est le décalage évident entre les transformations réalisées par l'utilisateur sur l'entité de déformation et le résultat observé sur l'objet. Cette différence est prévisible car le treillis n'est qu'une approximation (assez grossière parfois) de l'objet à déformer. Néanmoins ces méthodes restent plus intuitives.

Malgré ces quelques inconvénients, ces nouvelles méthodes ont amélioré la qualité et l'ergonomie des modeleurs.

La modélisation déclarative [DJE 91] s'inscrit parfaitement dans cette démarche. Mais ses objectifs sont plus ambitieux. L'idée générale est qu'il n'est plus nécessaire de communiquer d'informations numériques à la machine [GAI 93].

Dans le cas de la modélisation d'objets, l'utilisateur n'a plus à se préoccuper du type de modélisation pas plus que des moyens mis en œuvre pour exécuter ses commandes.

Les systèmes de réalité virtuelle se limitaient à leur début à des visites de lieux imaginaires ou reconstitués. Les utilisateurs veulent désormais avoir la possibilité d'interagir avec l'environnement dans lequel ils évoluent. Les éléments du décor sont devenus des objets à part entière : on peut les saisir, les déplacer, les poser les uns sur les autres, …

La réalité virtuelle, grâce aux nouveaux dispositifs d'entrée qu'elle propose, offre de nouvelles perspectives pour l'ergonomie de la modélisation, en introduisant de nouveaux concepts d'interaction.

Nous envisageons d'intégrer la modélisation déclarative dans un environnement de réalité virtuelle. Nous offrons à l'utilisateur la manière la plus naturelle qui soit pour exprimer sa commande : il la réalise lui-même en reproduisant des gestes qu'il effectuerait naturellement dans une situation réelle (i.e. marteler un objet, le forer, …).

A cet effet, nous développons un modeleur dans lequel l'utilisateur travaille sur un objet qu'il a lui-même créé ou bien qu'il a choisi dans une bibliothèque. Il le manipule jusqu'à obtenir la forme désirée.

Pour ce faire, nous utilisons une plate forme de développement d'applications de réalité virtuelle : VIPER.

3. VIPER : un système de définition d'environnements virtuels

VIPER (VIrtuality Programming EnviRonment) [TOR 95] est un système générique permettant la gestion d'environnements virtuels et, plus généralement, le développement d'applications de Réalité Virtuelle. Ce système gère en temps réel plusieurs utilisateurs munis de périphériques spécialisés grâce à une plate-forme distribuée. Nous allons présenter ici la structure générale d'un environnement virtuel telle qu'elle est définie dans VIPER.

Un environnement virtuel est composé d'entités, de stimuli et d'un univers virtuel (Fig. 3.1).

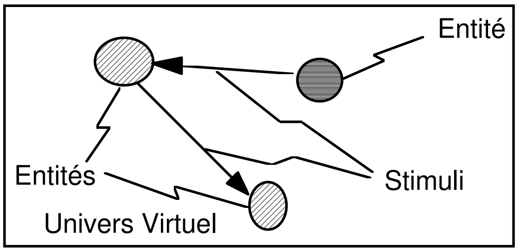


Figure 3.1 : Structure d'un environnement virtuel.

• La notion d'*entité* permet de gérer de façon uniforme les décors des mondes virtuels, les objets virtuels mais aussi des "clones" (objet associé à un utilisateur [MOU 93], à une application [WES 92] [CAR 93] ou à un robot, qui permet à celui-ci de participer à la simulation comme si il était réellement présent dans l'univers virtuel). 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 exposant le mécanisme de communication inter-entité et les mécanismes associés à la définition et à la prise en compte du comportement de ces entités (Fig. 3.2).

Les entités et le milieu, dans lequel elles évoluent, interagissent de deux façons :

- depuis le milieu vers les entités : un changement du milieu constitue un stimulus, tel un signal visuel ou sonore, perçu par les entités. Un capteur spécifique reçoit ce stimulus, l'interprète puis transmet une information à la partie comportementale de l'entité.
- depuis chaque entité vers le milieu : après analyse de l'information, la partie comportementale modifie l'état interne de l'entité et si nécessaire commande une action aux effecteurs. Ceux-ci agissent sur le milieu et en le modifiant créent de nouveaux stimuli.

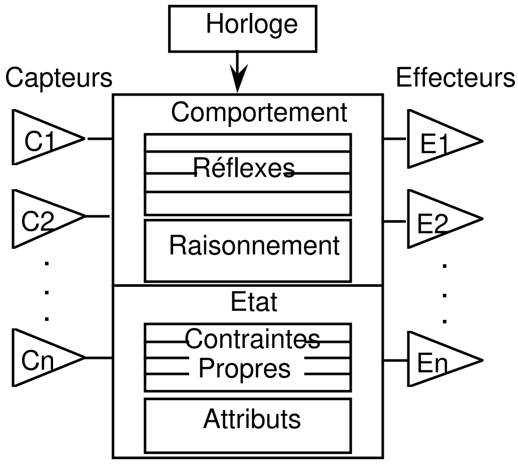


Figure 3.2 : Schéma structurel d'une entité.

Nous proposons de modéliser chaque type d'interaction avec quatre classes d'éléments : des capteurs, des effecteurs, des images et des espaces d'images (Fig. 3.3).

Une image est l'aspect perceptible d'une entité dans son milieu.

Un espace d'images est l'ensemble des images des entités liées à un certain type de stimuli.

Un capteur saisit une vue instantanée de l'espace d'image qu'il filtre puis interprète.

Un effecteur produit, par l'exécution d'une action, une image qu'il place dans l'espace d'images.

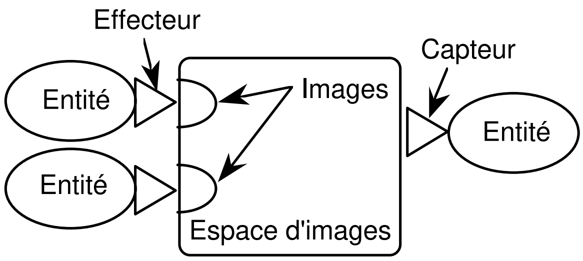


Figure 3.3 : Mécanisme de communication inter-entité.

L'introduction d'un espace d'image, qui joue un rôle de médiateur dans la communication, permet de simplifier la gestion des interactions dans un cadre distribué (collecticiel). L'espace d'image devient alors un agrégat d'images distribué qui gère les interactions de façon autonome. De plus, il traite uniformément les interactions locales à un site et les interactions impliquants des entités qui sont gérées par des sites différents.

Pour la définition de comportements, nous proposons la partition suivante :

- une composante "réflexe" où sont définis les réflexes de l'entité aux stimuli provenant de son environnement (via les capteurs). On peut citer en exemple de réflexe, la réaction d'une entité à une collision avec une autre entité.

- une composante "raisonnement" où est définie la partie nécessitant une analyse plus complexe et un caractère plus volontaire. La poursuite d'un but à atteindre par une entité illustre bien la notion de raisonnement.

La composante "réflexe" est constituée d'un certain nombre de modules chacun assignés à un capteur. Un module analyse les données envoyées par le capteur associé et réagit automatiquement en commandant, si nécessaire, des actions aux effecteurs et des changements de l'état interne de l'entité. Si un module réflexe reçoit des informations trop complexes (i.e. nécessitant un raisonnement de la part de l'entité) il peut activer la composante "raisonnement".

La composante "raisonnement" est régulièrement activée par une horloge interne qui modélise la fréquence propre de l'entité. Cette composante commande des actions aux effecteurs en fonction de la connaissance de l'environnement (obtenue par les capteurs), d'une mémoire et d'un raisonnement qui peut être modélisé de diverses façons (automate à états finis [CAR 93], moteur d'inférence PROLOG [MOU 92]...).

L'introduction de cette partition permet de dissocier la partie active du comportement, encapsulée dans la composante "raisonnement", de la partie réactive, définie au niveau de la composante "réflexe" qui définie les actions automatiques de l'entité.

L'entité possède, de plus, un état interne qui est un ensemble d'attributs (position, vitesse, masse, forme graphique...) auquel est associé un ensemble de contraintes propres (contraintes entre attributs de l'entité, comme par exemple vitesse ^{2} vitesse_maximum).

VIPER est écrit en langage C++ et utilise les concepts de la programmation par objets de façon à supporter des applications spécifiques dans divers domaines. Ceci est fait grâce aux mécanismes d'héritage et d'instanciation générique qui permettent de spécialiser des classes d'entités, de capteurs, d'effecteurs, d'images ou d'espaces d'images pour une utilisation plus spécifique.

4. Les déformations de forme libre virtuelles

C'est à [SED 86] que nous devons la définition des Free-Form Deformation (FFD). 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. Nous verrons plus loin les avantages, mais aussi les inconvénients, de l'introduction d'une entité intermédiaire de déformation.

Les FFD ont fait l'objet de nombreux articles. Aussi, nous ne rappellerons que brièvement ses 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. 4.1).

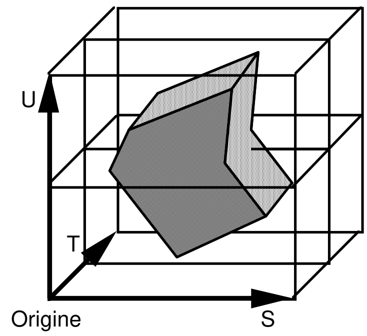


Figure 4.1 : Un exemple de treillis.

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.

La déformation de la normale associée au point traité utilise la matrice Jacobienne de la transformation [BAR 84].

La première remarque que l'on peut faire concerne l'indépendance des déformations libres vis à vis du modèle géométrique. Il est exact que ce type d'outils ne dépend pas du modèle de l'objet. Mais les données traitées (point et normale) nécessitent un échantillonnage de la surface de l'objet. Ainsi après calcul, le modèle géométrique originel (CSG, surfaces paramétriques, …) n'est plus conservé.

On peut facilement comprendre que les déformations que l'on peut réaliser sont fortement conditionnées par le nombre de points de contrôle. Plus ce nombre est élevé, plus les possibilités sont grandes. Mais en réalité, lorsque les points sont trop nombreux, il devient difficile de choisir sur lesquels il faut agir. De plus, l'influence d'un point sur la déformation globale décroît avec le nombre. Pour compenser cette faible influence, il faut déplacer un grand nombre de points. Enfin, le volume de calcul est directement lié à la densité du treillis en points de contrôle. Dans un processus interactif, il est impératif de réduire autant que possible l'attente de l'utilisateur.

Il est cependant indéniable que 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). On peut alors suggérer l'utilisation d'opérateurs de plus haut niveau qui agissent à leur tour sur le treillis [NIE 90]. Néanmoins d'après notre propre expérience avec les opérateurs de Barr [BAR 84] appliqués aux FFD, nous avons constaté que toute la difficulté consiste à communiquer les paramètres nécessaires à leur exécution. 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 ne nécessitant que des paramètres ayant une signification réelle (quelle partie d'un objet doit être pincée, la taille de la tête d'un marteau, …) et communiqués de la manière la plus naturelle (désignation graphique, choix du marteau, …).

Dans cette optique, la réalité virtuelle nous fournit la possibilité de mettre en œuvre ces concepts, notamment avec les outils auxquels sont associés 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 notre application).

A cet effet, nous étudions un ensemble d'outils basés sur les FFD tels que marteau, poinçon, pince, …

Nous allons maintenant présenter le premier qui a été réalisé : le marteau.

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 de la taille de la tête du marteau. 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. Cette même rigidité nous indique la densité de points de contrôle du treillis. Ainsi pour un objet donné et un type de marteau, nous créons un treillis (Fig. 4.2).

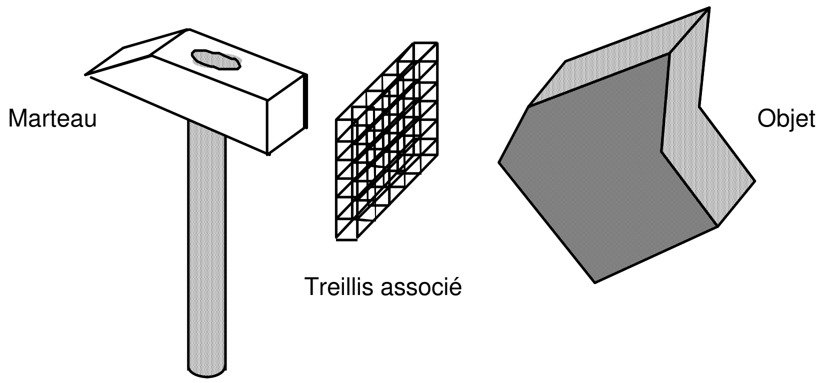


Figure 4.2 : Un type d'outils

Un "coup de marteau" est traité de la façon suivante : dès que la tête du marteau (en fait un point de la tête) pénètre dans un objet, le treillis est orienté selon la trajectoire du marteau (deux dernières positions) et translaté au point d'impact. Les coordonnées des points de l'objet sont calculées dans le repère associé au treillis. Ce dernier est modifié (avec prise en compte de la rigidité) puis l'objet est déformé. La figure 4.3 illustre, en deux dimensions, cet exemple.

Nous avons choisi une interpolation à base de Bézier, car le nombre de points de contrôle des treillis que nous utilisons n'est jamais très élevé et une interpolation à base de B-Spline nécessite un reparamétrage des coordonnées locales et quelques divisions qui sont coûteuses en temps.

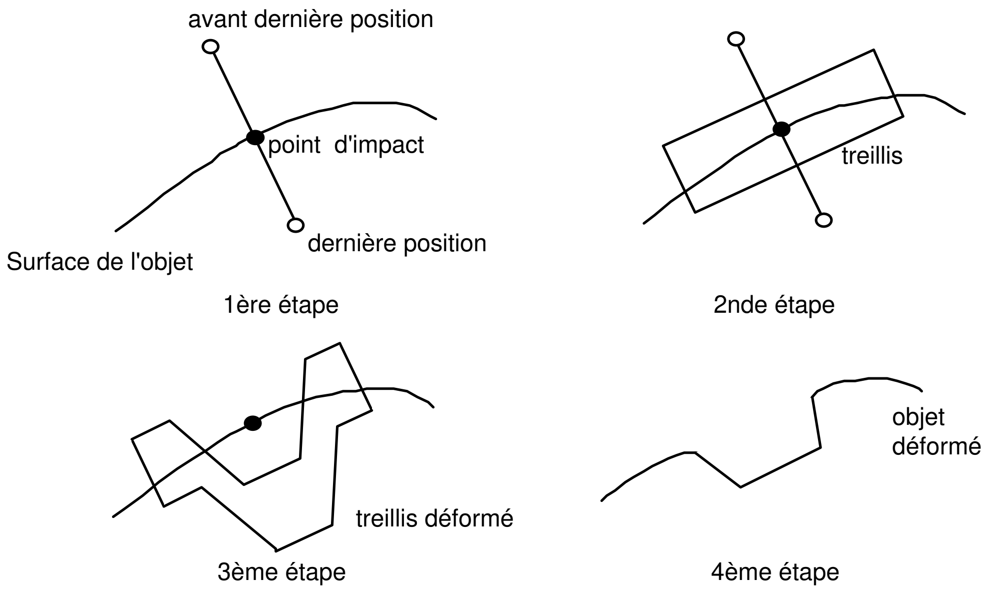


Figure 4.3 : Un "coup de marteau"

Le type de continuité entre la partie de l'objet qui est déformée et celle qui ne l'est pas, est déterminé par la rigidité associée à l'objet. La continuité d'une déformation FFD dépend de la position, relativement au treillis, des points de contrôle qui sont déplacés.

Pour une continuité C^0, les points situés sur la couronne extérieure[1] du treillis sont fixés. Pour une continuité C^1, il faut d'abord assurer une continuité C^0, et ensuite figer les points qui forment la couronne suivante.

5. Introduction de la notion d'outil dans VIPER

Un outil dans le modèle VIPER est un type particulier d'entité qui est manipulé par une ou plusieurs entités et qui, à son tour, manipule une ou plusieurs entités (Fig. 5.1).

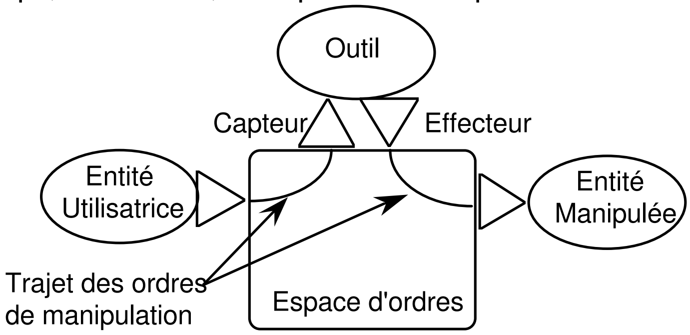


Figure 5.1 : Illustration de la manipulation via un outil.

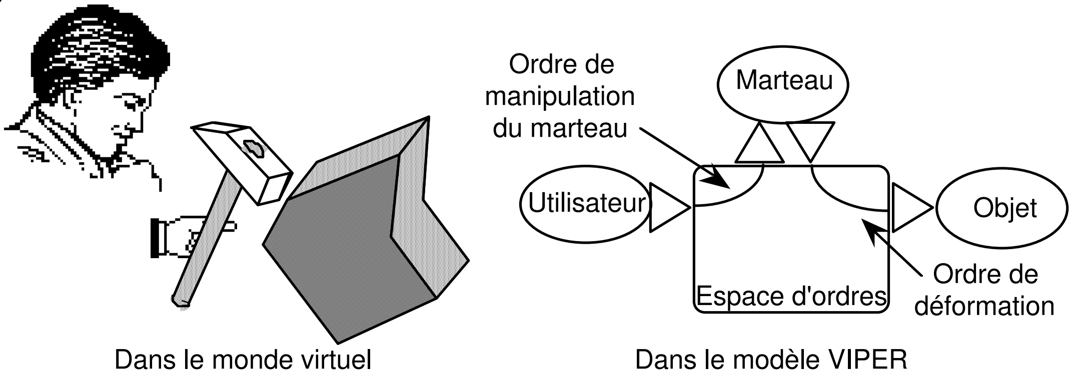
Un outil va posséder au minimum un capteur, appelé capteur d'ordre, qui récupère les ordres qui lui sont transmis ; ainsi qu'un effecteur d'ordre qui envoie un ordre de manipulation aux entités contrôlées par l'outil.

Le comportement d'une telle entité est typiquement un comportement réflexe déclenché par la récupération d'un ordre donné à l'outil et qui, en fonction du résultat de l'analyse, va produire, si nécessaire, un ordre de manipulation.

Chaque entité utilisatrice d'un outil doit posséder un effecteur d'ordre. Symétriquement, chaque entité manipulée dispose d'un capteur d'ordre et d'un module réflexe associé.

Un ordre est un type d'image spécifique qui est composé de :
- du nombre d'entités manipulées,
- des identificateurs de ces entités,
- du type de manipulation,
- des paramètres de cette manipulation.

Nous allons illustrer ces principes par la mise en œuvre d'un "coup de marteau" (Fig. 5.2).



---

[1] La couronne extérieure regroupe les points de contrôle appartenant aux plans du treillis séparant la partie déformée de la partie inchangée.

Figure 5.2 : 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 "coup de marteau" et comme paramètres les deux dernières positions 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 enfin elle calcule sa nouvelle forme.

6. Conclusion et perspectives

Pour l'instant, seul le marteau est disponible. Il nous a permis de valider le concept d'interaction qui marie les déformations de formes libres et la réalité virtuelle.

Pour l'instant, nous avons décidé de traiter quelques problèmes de précision qui subsistent encore. Frapper exactement l'endroit désiré n'est pas chose facile. Nous étudions donc des outils complémentaires de visualisation (loupe) et des outils permettant de raffiner localement la surface d'un objet, pour lesquels de nouvelles métaphores devront être introduites. Puis, nous réaliserons d'autres outils tels que la pince, le poinçon, …

A terme nous envisageons de créer un atelier de sculpture virtuelle où l'artiste pourra modeler des objets à partir d'une forme pré-existante ou d'un quelconque bloc de matière.

Parallèlement à la conception de ces nouveaux outils, le modèle VIPER nous permettra d'intégrer facilement la notion de collecticiel dans notre atelier grâce à la plate forme distribuée que nous avons développée. Néanmoins, le travail coopératif soulève de nombreuses questions. La prise de conscience du travail effectué par les autres utilisateurs semble primordiale. La modélisation concurrente d'un objet nécessite la mise en place d'un mécanisme d'exclusion mutuelle (gestion du recouvrement spatio-temporel des différents treillis mis en jeu).

Bibliographie

[AKK 93] S. Akkouche, E. Galin, "Déformation d'objets volumiques tridimensionnels par contraintes", Premières Journées Nationales AFIG-GROPLAN, Bordeaux, 1-3 Décembre 1993, pp 83-91.

[BAR 84] A.H. Barr, "Global and Local Deformation of Solid Primitives", Siggraph 84, vol 18, July 1984, pp 21-30.

[BLA 92] C. Blanc, P. Guitton, "Déformations géométriques progressives par des opérateurs normalisés", Groplan 92, pp 109.

[CAR 93] C. Carlsson, O. Hagsand, "DIVE - a Platform for Multi-User Virtual Environments", Computer & Graphics, Vol. 17, N° 6, Novembre Decembre1993.

[COQ 90] S. Coquillard, "Extended Free-Form Deformation : A Sculpting Tool for 3D Geometric Modeling", Computer Graphics 24, 4 (Aug 1990), pp 187-193.

[DJE 91] N. Djedi, "Modélisation en images de synthèse : utilisation d'une méthodologie déclarative", Thèse de docteur d'université, Université Paul Sabatier, 1991.

[FOR 88] David R. Forsey, Richard H. Bartels, "Hierarchical B-Spline Refinement", Computer Graphics, Vol 22, n°4, 1988.

[GAI 93] V. Gaildrat, R. Caubet, F. Rubio, "Declarative Scene Modelling with Dynamic Links and Decision Rules Distributed Among the Objects", IFIP International Conference on Computer Graphics ICCG93 Bombay, Février 1993.

[GAL 91] T. A. Galyean, J. F. Hughes, "Sculpting: An Interactive Volumetric Modeling Technique", Computer Graphics, Vol 25, n° 4, 1991, pp 267-274.

[HSU 92] W.M. Hsu, J.F. Hughes, H. Kaufman, "Direct Manipulation of Free-Form Deformation", Computer Graphics 92 proceedings, pp 177-184.

[KAU 91] A. Kaul, J. Rossignac, "Solid-Interpolating Deformations, Construction and Animation of PIPS", Eurographics 91, pp 493-505.

[KEN 92] James R. Kent, Wayne E. Carlson, Richard E. Parent, "Shape Transformation for Polyhedral Objects", Computer Graphics 26, 2 (July 1992), pp 47-54.

[LAZ 92] F. Lazarus, S. Coquillard, P. Jancène, "Déformations axiales interactives", Groplan 92, p 125.

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

[NIE 90] Z. Nie, B. Péroche, "Déformations libres et arbres de construction", Groplan 90, p 98.

[PAR 89] Shenchang E. Chen, Richard E. Parent, "Shape and Its Applications to Industrial Design", IEEE Computer Graphics and Applications 9, 1 (Jan. 1989) pp 47-54.

[PEN 90] A. Pentland, I. Essa, M. Friedmann, B. Horowitz, S. Sclaroff, "The ThingWorld Modeling System: Virtual Sculpting By Modal Forces", Computer Graphics, Vol 24, n° 2, Mars 1990, pp 143-144.

[SED 86] Thomas W. Sederberg, S.R. Parry, "Free-Form Deformation of Solid Geometric Models", Siggraph 1986.

[TOR 95] P. Torguet, R. Caubet, "VIPER (VIrtuality Programming EnviRonment): A virtual reality applications design platform", Actes du 2nd Eurographics Workshop on Virtual Environments, Janvier 1995.

[WES 92] A. J. West, T. L. J. Howard, R. J. Hubbold, A. D. Murta, D. N. Snowdon, D. A. Butler, "AVIARY - A Generic Virtual Reality Interface for Real Applications", Virtual Reality Systems, sponsored by the British Computer Society, Mai 1992.