Modélisation volumique de surfaces non-manifold

Karine Lamboglia

Institut National Polytechnique de Lorraine

Ecole Nationale Supérieure de Géologie de Nancy
(E.N.S.G.-L.I.A.D.)
Centre de Recherche en Informatique de Nancy
(C.R.I.N.)

THESE

présentée devant l'Institut National Polytechnique de Lorraine
en vue de l'obtention du titre de

Docteur de l'I.N.P.L.

Informatique

par

Karine LAMBOGLIA

Modélisation Volumique de Surfaces Non-Manifold

Soutenue publiquement le 11 février 1994 devant le Jury composé de:

	[illegible]RDT	*Président*
Jean-Claude	*PAUL*	*Rapporteur*
Jean-Laurent	*MALLET*	*Rapporteur*
Pierre	*BOUCHET*	*Examinateur*
		Examinateur

Institut National Polytechnique de Lorraine

Ecole Nationale Supérieure de Géologie de Nancy (E.N.S.G.-L.I.A.D.)
Centre de Recherche en Informatique de Nancy (C.R.I.N.)

THESE

présentée devant l'Institut National Polytechnique de Lorraine
en vue de l'obtention du titre de

Docteur de l'I.N.P.L.

Informatique

par

Karine LAMBOGLIA

Modélisation Volumique de Surfaces Non-Manifold

Soutenue publiquement le 11 février 1994 devant le Jury composé de:

Philippe	*CINQUIN*	*Président*
Pascal	*LIENHARDT*	*Rapporteur*
Jean-Claude	*PAUL*	*Rapporteur*
Jean-Laurent	*MALLET*	*Examinateur*
Pierre	*BOUCHET*	*Examinateur*

A mes parents, avec tout mon amour...

Résumé

La plupart des systèmes de modélisation volumique utilisant la représentation par surfaces-frontières ne considèrent que la géométrie manifold. Pour un objet manifold, et plus précisément 2-manifold, chaque point a un voisinage homéomorphe à un disque 2D. Cette restriction du domaine de représentation constitue un inconvénient majeur pour des applications manipulant des surfaces naturelles, comme c'est le cas par exemple en Géologie ou en Médecine. Le système de modélisation proposé permet d'étendre le domaine de représentation en prenant en compte à la fois les conditions manifold et non-manifold.

Les objets utilisés sont représentés par des facettes triangulaires. Une surface est divisée en plusieurs morceaux de triangles connectés appelés faces. Alors qu'une surface peut être non-manifold, une face est toujours manifold. En fait, par définition, une condition non-manifold apparait seulement aux frontières d'une face.

Ce système de modélisation consiste à découper l'espace 3D en plusieurs volumes distincts fermés (régions), définis par leurs frontières (shells). Afin de déterminer les faces adjacentes composant la frontière d'une région, il faut introduire des structures spécifiques décrivant les relations d'adjacence entre les faces. Plus précisément, la topologie du modèle apparait à deux niveaux: la Macro-Topologie décrit les adjacences entre faces, la Micro-Topologie décrit les adjacences entre triangles. Les structures d'adjacence permettent de détecter et définir automatiquement toute fermeture d'un volume de l'espace 3D.

Deux méthodes de construction du modèle volumique ont été développées. La première est une méthode interactive utilisant un outil de collage de surfaces, modifiant ainsi la topologie du modèle en créant des relations d'adjacence entre les faces. La seconde méthode est entièrement automatique et consiste à créer les structures d'adjacence du modèle en s'appuyant sur sa topologie.

Abstract

Most of the existing solid modeling systems using the boundary representation consider only manifold geometry. For a manifold object, and more precisely 2-manifold, each point has a neighbourhood homeomorphic to a 2D disc. The restriction of the representation domain constitutes a major drawback for applications handling natural surfaces, as in the fields of Geology and Medecine. The proposed modeling system enables to extend the representation domain by taking into account both manifold and non-manifold conditions.

The objects used by this system are surfaces represented by triangular facets. A surface is divided in several parts of connected triangles called faces. While a surface may be non-manifold, a face is always manifold. As a matter of fact, by definition, a non-manifold condition appears only at the boundaries of a face.

This modeling system consists in dividing the 3D space into several distinct closed volumes (regions), defined by their boundaries (shells). In order to determine the adjacent faces composing the boundary of a region, there is a need to introduce specific structures describing the adjacency relationships between the faces. More precisely the topology of the model appears at two levels: the Macro-Topology describes the adjacencies between faces, the Micro-Topology describes the adjacencies between triangles. The adjacency structures enable to detect and define automatically every single volume closure.

Two methods for building a solid model have been developed. The first one is an interactive method using a tool for gluing surfaces, thus modifying the topology of the model by creating new adjacency relationships. The second method is entirely automatic and consists in creating the adjacency structures of the model by leaning on its actual topology.

Remerciements

En tout premier lieu, je tiens à remercier cordialement le Professeur Jean-Laurent Mallet, Directeur du Département Informatique de l'Ecole de Géologie pour la confiance qu'il a su m'accorder. Durant ces trois années de recherche, il m'a orientée avec sa compétence, son enthousiasme et sa gentillesse. Grâce au consortium GOCAD, il offre à ses étudiants un environnement exceptionnel, tant sur la qualité du matériel mis à leur disposition que sur l'opportunité de parcourir le monde au gré de congrès internationaux.

C'est au cours de l'un de ces congrès que j'ai eu la chance de rencontrer Monsieur Chuck Sword, que je tiens à remercier vivement pour l'aide précieuse qu'il m'a apporté tout au long de mes travaux de recherche.

J'aimerais également remercier Monsieur Philippe Cinquin, Professeur d'Informatique Médicale au laboratoire TIMB-IMAG de Grenoble, qui a accepté d'être le Président du jury ainsi que le rapporteur de ce mémoire.
Mes remerciements s'adressent également à Messieurs Pascal Lienhardt, Professeur à l'ULP de Strasbourg, et Jean-Claude Paul, directeur de l'équipe Graphis du CRIN, qui ont gentiment accepté d'être les rapporteurs de cette thèse.
De même, j'aimerais remercier Monsieur Pierre Bouchet, de l'équipe Infographie du CRIN pour son soutien lors des travaux de recherche et de la rédaction de ce manuscrit.

Merci également à tous les membres du LIAD et en particulier à Madame Monique Cugurno pour sa compétence, sa gentillesse et son dévouement.

Les travaux décrits dans ce mémoire font partie du projet GOCAD dont je tiens à remercier les sponsors actuels:

Universités	Compagnies
Université de Bonn	Amoco
Colorado School of Mines	B.R.G.M.
Université de Freiburg	C.G.G. - Petrosystems
I.P.G. (Paris)	Chevron
Université de Karlsruhe	Digital Equipment
Ecole de Géologie de Nancy	Elf Aquitaine
Osservatorio Geofisico di Trieste	Exxon
Ecole des Mines de Paris	Gaz de France
Politecnico di Milano	Hewlett-Packard
Université de Stanford	I.B.M.
	I.F.P.
	Kubota Pacific
	Phillips Petroleum
	Shell Research
	Silicon Graphics
	Statoil
	Sun Microsystems
	T.N.O. Institut
	Total
	Unocal

Table des matières

Objectif

La majorité des systèmes de modélisation volumique que l'on trouve dans les logiciels de Conception Assistée par Ordinateur (C.A.O.) et utilisant la représentation par frontières ne considèrent que des cas de géométrie **manifold**. Pour des applications manipulant des surfaces naturelles complexes, comme c'est le cas en Géologie et en Médecine, cette limitation du domaine de représentation est un inconvénient majeur.

Notre objectif a été de construire un modèle volumique permettant la représentation des objets **non-manifold**, dans le cadre du projet GOCAD. Cette méthode, dont le principe fondamental est la description et l'utilisation de la topologie des divers éléments composant l'objet à modéliser, a été inspirée de la **Radial Edge Structure** introduite par K. Weiler [38].

ORGANISATION DU MANUSCRIT

Ce manuscrit se décompose de la façon suivante:

1. **Introduction:**
   Cette partie expose dans un premier temps certaines notions sur la modélisation géométrique en général, et la modélisation volumique en particulier. Une présentation du cadre de la recherche est également proposée dans cette introduction, présentation comprenant quelques définitions utiles pour une bonne compréhension des travaux présentés dans ce manuscrit.

2. **$1^{ère}$ Partie: Modélisation volumique non-manifold:**
   Dans un premier chapitre, nous présenterons le modèle qui a servi de base à notre recherche: le modèle de Weiler. Le deuxième chapitre de cette partie exposera les structures définissant les volumes de notre modèle. Le troisième chapitre sera consacré à la présentation des concepts de base de description des relations d'adjacence, introduits dans notre système de modélisation volumique. Enfin le quatrième et dernier chapitre de cette partie présente les moyens mis en place pour détecter et gérer les volumes clos dans notre modèle volumique.

3. **2^{nde} Partie: Construction interactive du modèle:**
   Les trois chapitres de cette partie sont consacrés à la première méthode de construction du modèle volumique mise en place au cours des travaux de recherche. Le chapitre 5 donne une définition précise de l'outil créé pour lier deux surfaces: la Glue-Line. Le chapitre 6 présente les divers processus permettant de créer des Glue-Lines afin de coller les surfaces entre elles. Le chapitre 7 décrit l'utilisation des Glue-Lines lors du collage de surfaces.

4. **$3^{ème}$ Partie: Construction Automatique du modèle:**
   Cette dernière partie est consacrée à la deuxième méthode de construction du modèle, méthode mise au point pour pallier aux inconvénients de la méthode interactive. Le chapitre 8 expose le traitement imposé aux données initiales pour optimiser le processus de calcul du modèle. Le chapitre 9 présente les diverses étapes de la construction du modèle volumique non-manifold.

5. **Conclusion:**
   En conclusion, on donnera les avantages et inconvénients de notre méthode de modélisation volumique non-manifold, ainsi que ses applications actuelles et envisagées.

6. **Appendices:**
   On pourra trouver en annexe diverses informations relatives aux deux méthodes exposées. Ces informations sont: la base de données du modèle, les opérations de manipulation du modèle...

7. **Lexique:**
   Les mots spécifiques à notre système de modélisation seront indiqués en italique, et on pourra trouver leur définition dans le lexique en Page 153.

INTRODUCTION

Introduction

0.1 Généralités

Avant de parler de notre système de modélisation volumique non-manifold, il est important de définir précisément ce qu'est un modèle et ce qui caractérise une bonne modélisation géométrique. Selon Foley et Van Dam [9], "*L'objectif du modèle est de permettre à l'utilisateur de visualiser, comprendre et interpréter la structure ou le comportement d'une entité.*".

Dans ce chapitre d'introduction, nous énumèrerons dans un premier temps les critères d'une bonne modélisation géométrique. Nous décrirons ensuite brièvement les divers types de modélisation existants. Enfin nous exposerons les caractéristiques de notre système de modélisation volumique non-manifold.

0.1.1 Critères d'une bonne modélisation géométrique

Une modélisation géométrique est dite bonne, ou de haut niveau, si elle vérifie les conditions suivantes [11]:

1. **Validité:** (ou Cohérence)
   le modèle ne doit pouvoir comporter que des représentations valides, c'est-à-dire correspondant à des objets réels.

2. **Puissance:**
   l'étendue du domaine définit la puissance du modèle, c'est-à-dire l'ensemble des objets modélisables.

3. **Suffisance:**
   une représentation est dite **non-ambigüe** si elle correspond à un seul objet. Le modèle est dit non-ambigu si toute représentation de celui-ci est non-ambigüe. Une représentation non-ambigüe contient assez d'informations pour définir un objet et pour tout calcul mathématique.

4. **Unicité:**
   une représentation est dite **unique** si l'objet correspondant n'admet pas d'autre représentation. Le modèle est dit unique si tous ses éléments sont uniques. La plupart des modèles ne sont pas uniques. La vérification que deux représentations d'un même modèle donné

sont images du même objet imposera, même dans les rares cas où c'est décidable, la mise en oeuvre d'algorithmes complexes.

5. **Calculs:**
   un certain nombre de grandeurs géométriques peut être calculé (volume, masse...).

0.1.2 Petit historique des méthodes de modélisation géométrique

Nous présentons dans ce paragraphe un bref résumé de l'évolution des techniques de modélisation géométrique [32].

Modélisation Fil de Fer

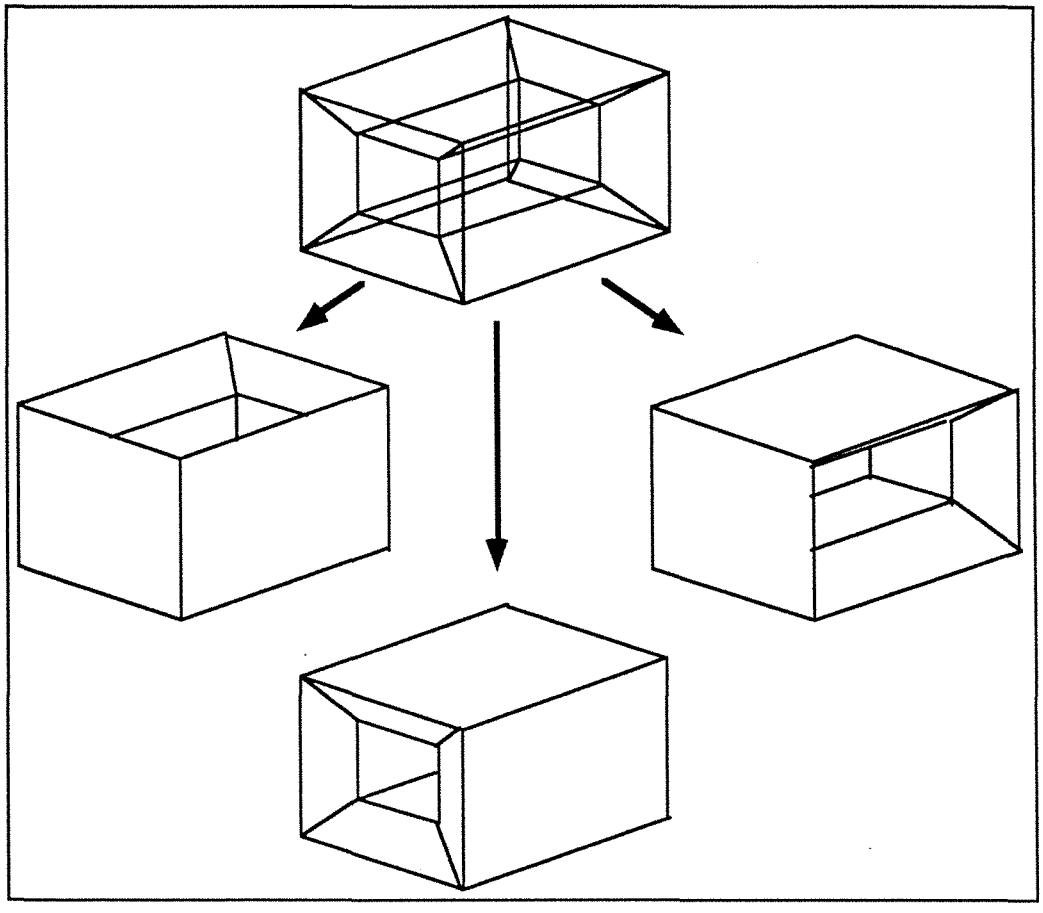


Figure 0.1 Modèle Fil de Fer.

C'est la première des techniques de modélisation mise au point, au début des années 70. Dans ce modèle, les objets sont représentés par un ensemble de courbes. L'avantage principal de cette méthode réside dans le fait que l'ordinateur peut générer rapidement une représentation visuelle d'un objet, quelques soient le point de vue et la projection choisis par l'utilisateur. Les modèles Fil de Fer présentent cependant de nombreux inconvénients:

- certains sont impossibles,
- beaucoup sont ambigus (comme le montre la figure 0.1, un même modèle peut avoir différentes interprétations),
- la plupart sont incomplets,
- tous sont très fastidieux à construire.

De plus, du fait que les surfaces ne sont pas explicitement représentées, il est impossible de localiser les "trous", de décrire des formes complexes, et de générer de façon automatique des vues avec surfaces-cachées qui rendent la visualisation beaucoup plus agréable lors de manipulations d'objets complexes. Actuellement, la modélisation Fil de Fer est utilisée pour mettre au point de façon rapide la représentation visuelle d'un modèle donné; de part son manque d'information, elle ne peut être utilisée pour des interprétations du modèle.

Modélisation Surfacique

Pour surmonter les déficiences des modèles Fil de Fer, on a commencé à développer des méthodes de modélisation de surfaces (voir Figure 0.2). Pour cela, deux techniques sont couramment employées [12]:

1. **Interpolation:**
   Une surface est construite de façon à passer par toutes les courbes d'un ensemble donné. La surface doit donc interpoler les points entre les courbes. En fait, cela revient à construire une surface sur un modèle Fil de Fer.

2. **Approximation discrète:**
   Le système génère une surface approchant la forme d'un ensemble fini de points appelés points de contrôle.

Les modèles surfaciques comportent des avantages très importants par rapport aux modèles Fil de Fer. En effet, il est impossible de créer un modèle non-valide. De plus, la majorité des modèles construits sont non-ambigus. D'autre part, ce type de modélisation permet de créer des objets complexes. Enfin, la localisation des "trous" peut être réalisée de façon complète et non-ambigüe. Cependant, les modèles surfaciques présentent également des inconvénients.

- Tous les objets ne peuvent pas être modélisés (par exemple la bouteille de Klein).
- Il n'existe aucune garantie que les surfaces bordent un objet solide. Le modèle n'est donc pas complet.

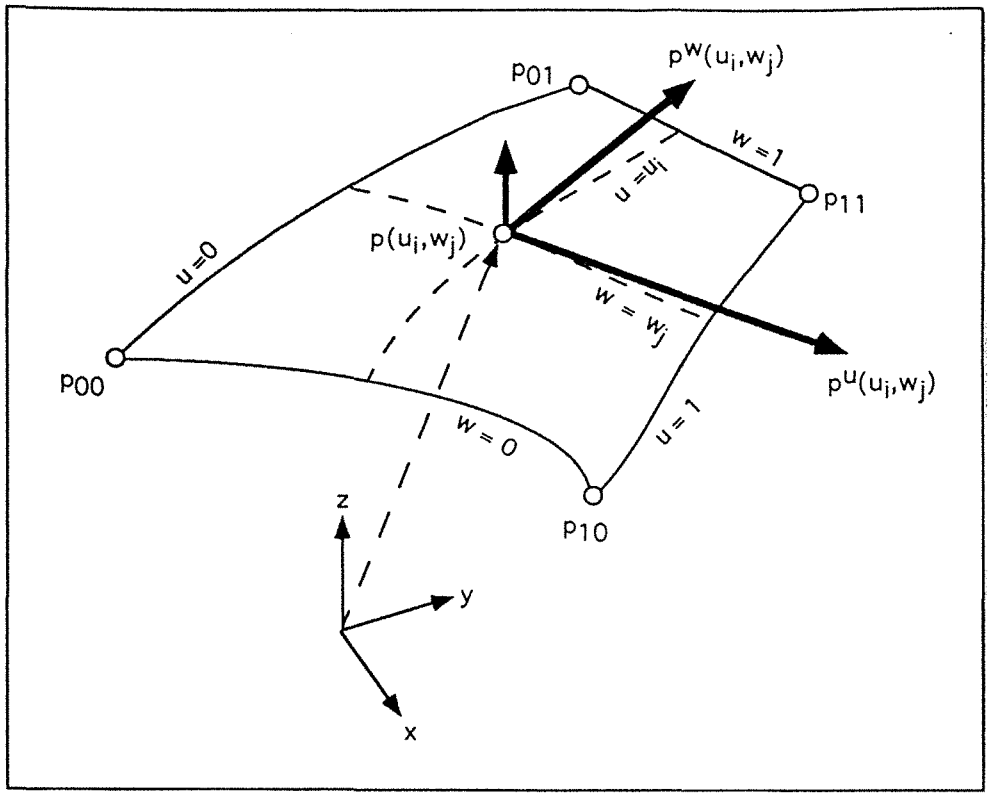


Figure 0.2 Modèle surfacique.

- Ce système de modélisation n'apporte aucune information concernant la connexité des surfaces. En d'autres termes, même si les surfaces individuelles sont parfaitement définies de façon mathématique, il n'y a aucun moyen de déterminer directement, à partir des informations disponibles, quelle surface est adjacente à une autre, sur une frontière donnée. L'absence d'informations topologiques ne permet pas d'effectuer certaines opérations de façon automatique, comme par exemple la vérification de la correction du modèle. Les systèmes de modélisation surfacique sont très répandus dans les industries automobile, aérospatiale et navale.

Modélisation Volumique

La modélisation volumique a pour objectif de représenter un objet par des solides distincts, formant ainsi un découpage de l'espace de modélisation (espace 3D).

Les surfaces d'un volume solide doivent être orientées et fermées, de façon à ce qu'il y ait une distinction claire entre l'intérieur et l'extérieur du volume. Requicha [33] établit les propriétés nécessaires et suffisantes que doit avoir un objet pour être un solide:

- **Homogénéité:**
  un solide doit avoir un intérieur.

- **Finitude:**
  un solide doit occuper une portion finie de l'espace.

- **Rigidité:**
  un solide doit avoir une forme invariante quelles que soient sa position et son orientation.

La modélisation volumique réunit les avantages des modeleurs Fil de Fer et Surfacique. En d'autres termes, un modèle volumique possède des informations sur:

- les faces d'un objet,

- les courbes qui bordent ces faces.

De plus, un modèle volumique contient des détails sur la connexité des divers éléments composant l'objet modélisé, c'est-à-dire les points, courbes et faces. Ces éléments topologiques doivent obéir aux propriétés suivantes:

1. **Non-Intersection:**
   Les volumes ne doivent pas s'intersecter, sauf au niveau de leurs frontières. Les faces ne doivent pas s'intersecter, sauf le long de leurs frontières. Les courbes ne doivent pas s'intersecter, sauf à leurs extrémités. Les points doivent être distincts. De plus, les courbes ne doivent pas intersecter les faces. Cette restriction est nécessaire pour éviter à des éléments topologiques de pénétrer des faces ou des volumes sans que le système ait la connaissance des structures les représentant.

2. **Finitude:**
   Les points ont une position finie dans l'espace, les courbes ont une longueur finie, les faces ont une aire finie, et les volumes fermés ont un volume fini. De plus, toute forme représentable doit l'être avec un nombre fini d'éléments topologiques.

La description est suffisamment complète pour qu'une grande variété d'applications -outre les applications des modèles Fil de Fer et Surfacique- puissent être réalisées de façon automatique (calcul du volume, de la masse, et des moments d'inertie d'un objet...).

0.1.3 Techniques de modélisation volumique

Ce paragraphe décrit brièvement les trois techniques les plus fréquentes de modélisation volumique [32].

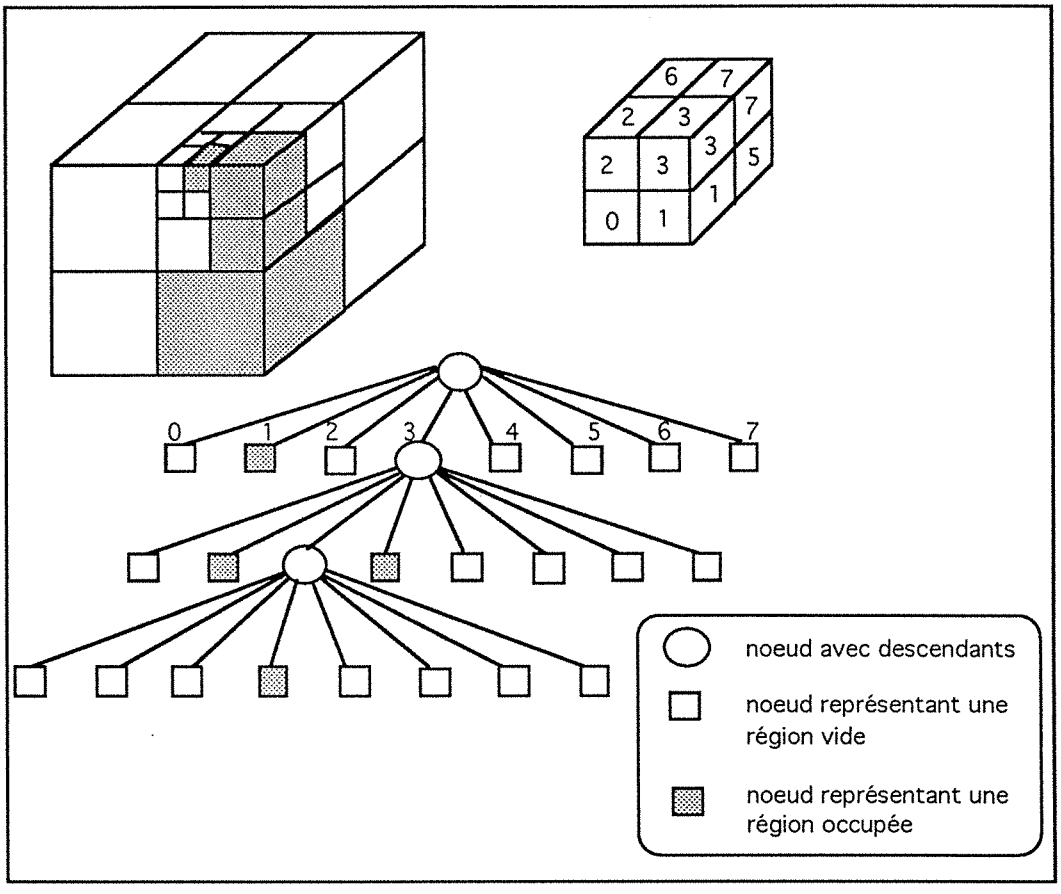


Figure 0.3 Modélisation volumique par octree.

Modélisation Cellulaire

Dans cette méthode, l'objet est contenu dans un tableau de cellules, où chaque cellule peut être occupée ou non. L'exemple le plus utilisé de cette technique de modélisation est l'**octree** (voir Figure 0.3). D.J.Meagher (1982) a développé une méthode de modélisation volumique basée sur l'octree permettant une plus grande rapidité dans la manipulation, l'analyse et la représentation visuelle d'objets solides. Son approche utilise un arbre à huit noeuds pour représenter les solides. Chaque région cubique du modèle initial est divisée en huit régions cubiques, ce qui correspond à un découpage de l'espace par 2, en x, en y et en z. Chaque noeud de l'octree qui n'est pas une feuille a lui-même huit descendants.

L'avantage principal de cette technique est que l'accès à un point se fait de façon très aisée et rapide, par un simple parcours d'arbre. La modélisation cellulaire présente également des inconvénients:

- il n'existe pas de relations explicites entre les diverses parties d'un objet,

- pour obtenir une bonne définition d'un objet, il est nécessaire d'utiliser un grand nombre de petites cellules. D'où des problèmes d'espace-mémoire.

Modélisation par C.S.G.

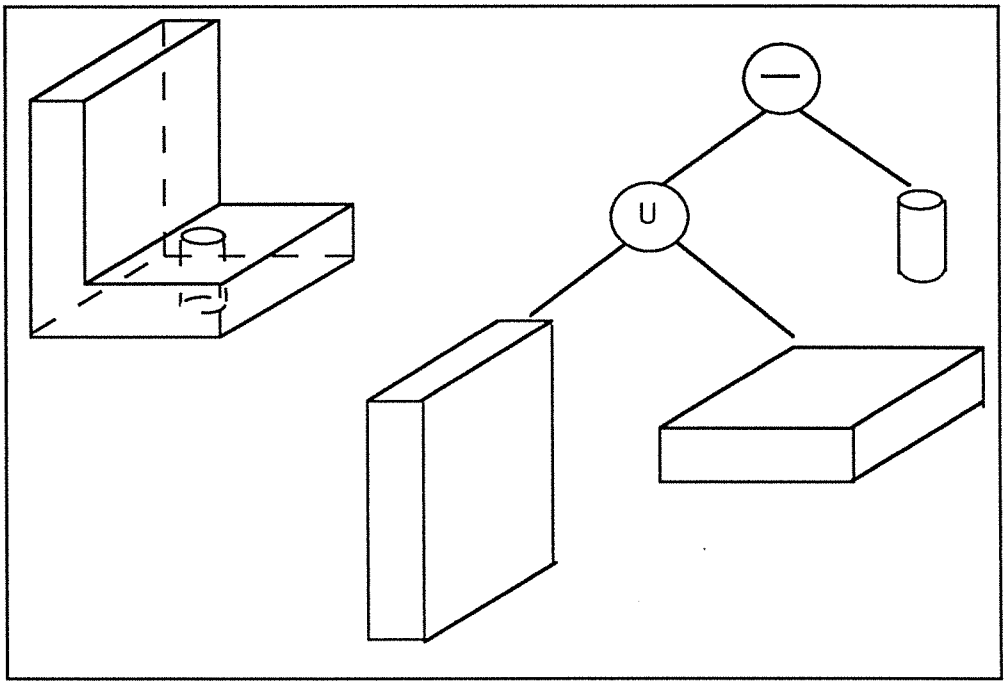


Figure 0.4 Modélisation volumique par C.S.G.

La C.S.G. est la **Constructive Solid Geometry**, introduite pour la première fois à l'Université de Rochester [34]. Dans cette méthode, un objet est représenté par un arbre binaire dont les noeuds sont des opérateurs booléens et les feuilles des primitives volumétriques simples (voir Figure 0.4). Parmi ces éléments de base, on trouve: le bloc, le cylindre, le cône, la sphère, le tore... Les opérateurs booléens sont l'union, l'intersection et la différence.

La validité d'un modèle volumique construit par la méthode C.S.G. se vérifie aisément si les primitives utilisées ont été générées par le système, et si les opérateurs sont parmi les trois énumérés précédemment. Dans le cas contraire, c'est-à-dire si l'utilisateur a créé le modèle à partir de ses propres primitives et/ou opérateurs, c'est à lui de vérifier la validité du modèle. La méthode C.S.G. permet de générer rapidement une représentation compacte d'un objet complexe. Cependant, des calculs supplémentaires sont nécessaires pour générer les frontières des objets afin de leur attacher diverses propriétés.

Modélisation par frontières

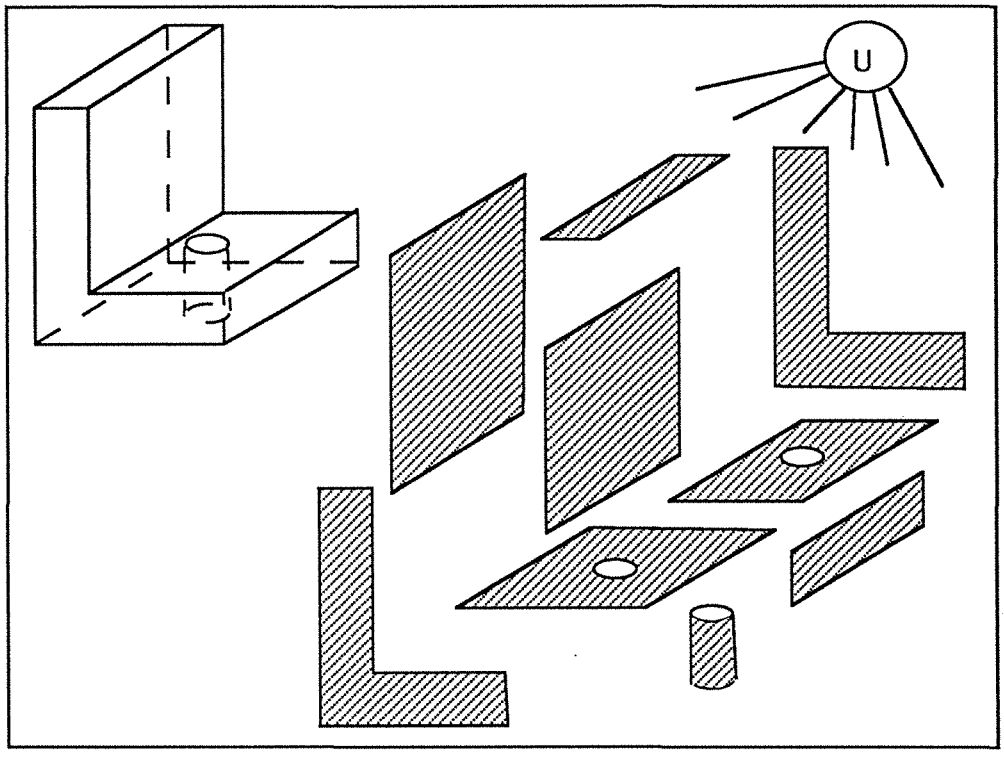


Figure 0.5 Exemple de modélisation volumique par frontières.

Cette méthode est référencée dans la littérature par le terme de **Boundary Representation** (noté **Brep**) [9]. La méthode Brep décrit un objet en terme de surface-frontière (voir Figure 0.5). Certaines Breps sont restreintes à des frontières planes et polygonales, et peuvent également nécessiter que les faces soient des polygones convexes ou des triangles. En effet, déterminer ce qui constitue une face peut se révéler particulièrement complexe si les surfaces courbes sont autorisées. Les faces courbes sont souvent approchées par des polygones, mais peuvent également être représentées par des patch.

Les méthodes Brep sont utilisées dans de nombreuses techniques de modélisation existantes. La plupart ne peuvent prendre en compte que des solides dont les frontières sont 2-manifolds. Par définition, chaque point d'un objet 2-manifold a un voisinage équivalent topologiquement à un disque 2D. Le méthode Brep permet d'obtenir des informations topologiques concernant les interconnexions entre les divers éléments. Il existe neuf types de relation d'adjacence dans un modèle utilisant les éléments topologiques face, courbe et point (voir Figure 0.6).

La méthode Brep possède les propriétés suivantes:

1. **Puissance:**
   L'espace de modélisation des Breps dépend de l'ensemble des surfaces pouvant être utilisées.

Il n'y a aucune raison de limiter celles-ci aux demi-espaces; donc les Breps peuvent être utilisées pour représenter un ensemble d'objets plus large que la méthode C.S.G.

2. **Validité:**
   La validité des modèles Breps est en général relativement difficile à établir. En effet, le critère de validité se divise en deux: les contraintes topologiques et les contraintes géométriques. Alors qu'il est possible de gérer les contraintes topologiques sans grande dépense, il est complexe de forcer la correction géométrique sans pénaliser la vitesse de la construction interactive.

3. **Non-ambigüité et Unicité:**
   Les modèles Breps valides sont non-ambigus. Ils ne sont pas uniques.

4. **Suffisance:**
   Dans une topologie d'adjacence consistant de trois éléments de base (face, edge et vertex), il peut exister neuf relations d'adjacence possible. Si une représentation topologique contient suffisamment d'informations pour recréer ces neuf relations d'adjacence sans erreur, elle peut être considérée comme une représentation topologique suffisante.

Représentation Multiple

Certains systèmes utilisent des représentations multiples car certaines opérations sont plus efficaces avec une représentation qu'avec une autre. Par exemple, certains systèmes utilisent la représentation C.S.G. pour sa compacité et la méthode Brep pour un accès direct à des données utiles qui ne sont pas spécifiquement représentées par la méthode C.S.G., comme la connexité des éléments.

En plus de ces systèmes qui maintiennent deux représentations complètement séparées, dérivant l'une de l'autre quand c'est nécessaire, il existe également des systèmes dits **hybrides** qui descendent jusqu'à un certain niveau de détail sur une représentation, puis passent à un autre schéma sans qu'il soit nécessaire de dupliquer les informations.

0.1.4 Comparaison des techniques de modélisation volumique

Foley et Van Dam [9] dressent un tableau comparatif des diverses techniques de modélisation volumique présentées précédemment, en fonction des critères de précision, domaine, unicité, validité, fermeture, compacité et efficacité.

Précision

Une représentation précise permet la représentation d'un objet sans approximation.

La modélisation cellulaire et la méthode Brep polygonale ne produisent que des approximations pour de nombreux objets. Pour certaines applications, comme par exemple trouver un chemin pour un robot, ceci n'est pas un inconvénient tant que l'approximation est calculée avec

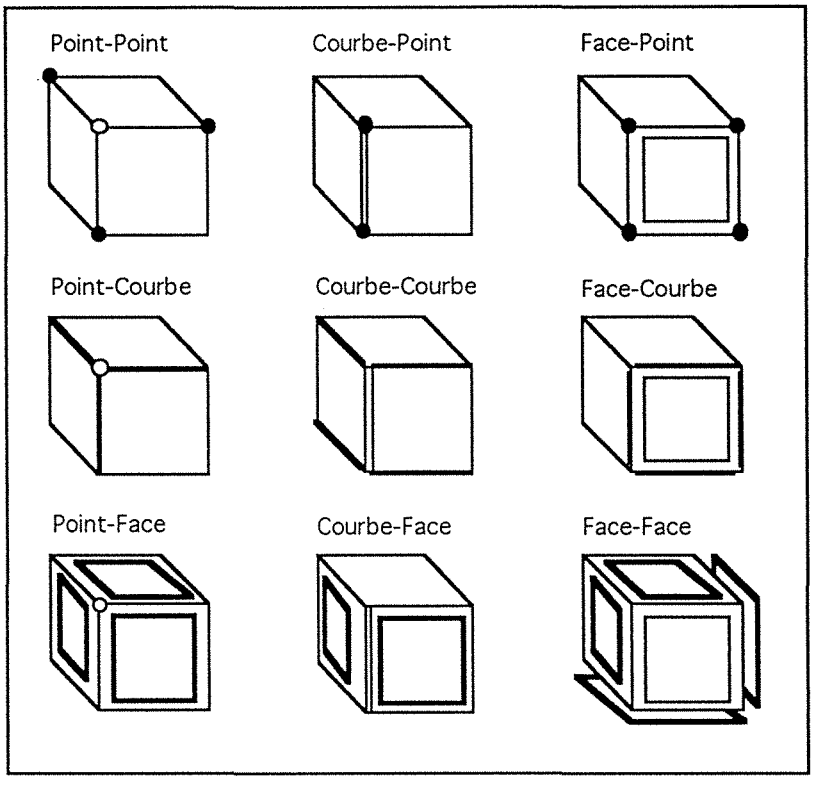


Figure 0.6 Les neuf relations d'adjacence entre les faces, courbes et points.

une résolution suffisante et adaptée au problème. La résolution nécessaire pour produire des images graphiques "acceptables" ou pour calculer des interactions entre objets s'avère cependant trop grande pour être possible.

Les systèmes utilisant des outils graphiques de haute qualité utilisent souvent la méthode C.S.G. ou une méthode Brep manipulant des surfaces courbes.

Domaine

Le domaine de représentation devrait être suffisamment large pour permettre de représenter un vaste ensemble d'objets.

La modélisation cellulaire permet de représenter n'importe quel solide, la plupart du temps sous forme d'approximation. Les systèmes Brep permettent de représenter une grande variété d'objets. Cependant la majorité de ces systèmes sont restreints à des surfaces et des topologies simples. Par exemple, ils peuvent seulement représenter des combinaisons d'éléments 2-manifolds.

Unicité

Une modélisation est unique si elle peut être utilisée pour représenter tout solide de façon unique.

Seule la modélisation cellulaire garantit l'unicité d'une représentation: il n'y a qu'une seule façon de représenter un objet avec une taille et une position précises.

Validité

Un système de modélisation ne devrait pas pouvoir créer une représentation invalide, c'est-à-dire une représentation qui ne correspond pas à un solide. Inversement, il devrait être aisé de créer une représentation valide, à l'aide d'un système de modélisation volumique interactif.

Parmi toutes les techniques de modélisation volumique, la méthode Brep s'avère la plus complexe à valider. Non seulement les points, courbes et faces peuvent être inconsistants, mais les faces ou les courbes peuvent également s'intersecter. Par opposition à la méthode Brep, les techniques de modélisation cellulaire ou C.S.G. ne requièrent que de simples vérifications locales pour assurer leur validité.

Fermeture

La fermeture des volumes doit être maintenue sous les opérations de rotation, translation... En d'autres termes, appliquer ces opérations à des objets valides ne devrait produire que des objets valides.

Malgré certaines méthodes Brep qui peuvent souffrir de problèmes de fermeture dus aux opérations booléennes (comme par exemple l'impossibilité de représenter autre chose que du 2-manifold), ces problèmes peuvent souvent être évités.

Compacité et Efficacité

Une représentation doit être compacte afin d'économiser de l'espace-mémoire. Enfin, une représentation devrait permettre d'utiliser des algorithmes efficaces pour calculer les propriétés physiques désirées et, plus important encore, pour créer des images.

Les schémas de représentation sont souvent classés selon qu'ils produisent des modèles évalués ou non. Les modèles non-évalués contiennent des informations qui doivent être manipulées (évaluées) afin de réaliser des opérations de base, comme par exemple déterminer les frontières d'un objet.

La méthode C.S.G. crée des modèles non-évalués car à chaque étape, l'arbre doit être parcouru afin d'évaluer les expressions. En conséquence, les avantages de la méthode C.S.G. sont sa compacité, le stockage rapide des opérations booléennes et de leurs implications, et l'annulation rapide de ces opérations.

Les méthodes Brep et cellulaire sont souvent considérées comme évaluées car les opérations booléennes utilisées pour créer le modèle ont déjà été exécutées.

0.1.5 Modélisation volumique non-manifold

Notre objectif est de créer un nouveau système de modélisation volumique, prenant en compte à la fois les objets manifold et non-manifold. Les objets à modéliser sont représentés par des surfaces créées par le logiciel GOCAD.

Méthode de modélisation utilisée

Le système de modélisation que nous proposons dans cet ouvrage fait partie de la catégorie des méthodes Brep. Cette méthode a été choisie car elle présente des avantages importants, malgré quelques inconvénients:

- **Avantages:**
  - on peut obtenir des informations sur les relations d'adjacence des objets, au niveau de leurs frontières.
  - à l'inverse de la méthode C.S.G., qui nécessite la combinaison d'éléments de base pour construire un objet, on peut imaginer plusieurs techniques de création dans la méthode Brep, par exemple en calculant les intersections entre les surfaces ou en créant automatiquement de nouvelles surfaces-frontières en s'appuyant sur celles existantes (patch).
  - l'accès aux informations géométriques se fait de façon rapide car elles sont attachées à chaque frontière de l'objet modélisé.
- **Inconvénients:**
  - il est moins facile qu'avec la méthode C.S.G d'avoir l'assurance de la validité du modèle construit. En effet, celle-ci dépend non seulement de la validité des éléments utilisés (créés par l'utilisateur), mais également de la cohérence des opérations effectuées sur ces éléments.
  - les structures de données sont complexes et généralement très redondantes pour permettre un accès rapide aux éléments.

Modèle non-manifold

Dans une représentation géométrique *manifold* (et plus précisément 2-*manifold*), chaque point d'une surface a un voisinage homéomorphe à un disque 2D. En d'autres termes, certains modèles ne peuvent être représentés (Figure 0.7), comme:

- deux surfaces se joignant en un point unique,
- deux surfaces se joignant le long d'une courbe ouverte ou fermée,
- deux volumes distincts fermés partageant une face comme frontière commune,

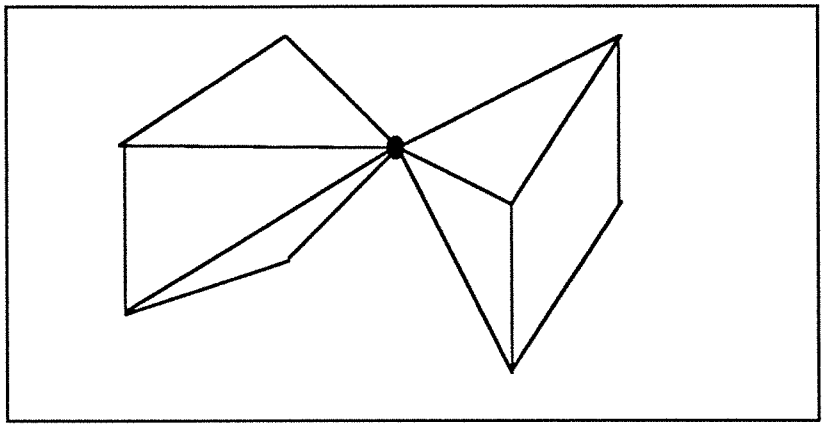


Figure 0.7 Conditions *non-manifold* en un point.

- une droite émanant d'un point d'une surface.

La modélisation géométrique *non-manifold* est un domaine comprenant les conditions *manifold* et *non-manifold*. Par la suite, nous appliquerons le terme *non-manifold* non seulement au modèle, mais également aux objets modélisés, ainsi qu'à leurs frontières. En effet, d'après la définition du terme *non-manifold*, il en découle que ce type de propriété apparait aux frontières des objets modélisés.

0.2 Cadre de la Recherche

Les travaux présentés dans cette ouvrage ont été réalisés dans le cadre du projet GOCAD développé à l'Ecole de Géologie de Nancy (INPL) par l'équipe Infographie du CRIN, sous la direction du Professeur J.L. Mallet.

0.2.1 Introduction

Le système GOCAD est à la base un ensemble de librairies graphiques sur lequel diverses applications se sont greffées. Contrairement à la CAO classique, qui permet de construire des courbes ou des surfaces (Bézier, B-splines,...) de façon mathématique, GOCAD propose un nouveau type de CAO: la CAO de surfaces naturelles. En effet, l'intérêt de GOCAD réside dans le fait que ce système s'applique à des données complexes pour la construction de surfaces. Ces données peuvent être:

- irrégulièrement distribuées,
- plus ou moins précises,
- d'origines diverses.

La triangulation est la méthode de modélisation des surfaces choisie dans le projet GOCAD. En effet, ce type de maillage s'est révélé très efficace pour modéliser les surfaces complexes rencontrées par exemple en Géologie ou en Médecine.

Tout le système est fondé sur un principe unique d'interpolation mis au point par le Professeur Mallet: D.S.I. (Discrete Smooth Interpolation). Cet interpolateur permet de lisser un objet, c'est-à-dire minimiser sa rugosité globale, en minimisant les rugosités locales au niveau des triangles (voir Figures ?? et ??). De plus, D.S.I. permet de prendre en compte diverses contraintes attachées aux objets, comme par exemple:

- Control Node: permet de fixer un point donné (x,y,z) dans une ou plusieurs directions.
- Fuzzy Control Node: permet de fixer, avec un facteur de certitude, un point donné (x,y,z) dans une ou plusieurs directions.
- Fuzzy Control Point: permet de forcer une surface à adapter sa forme à un ensemble de points donné.
- On-Tsurf: permet de forcer le bord d'une surface à rester en contact avec une autre surface.
- ...

0.2.2 Définitions utiles

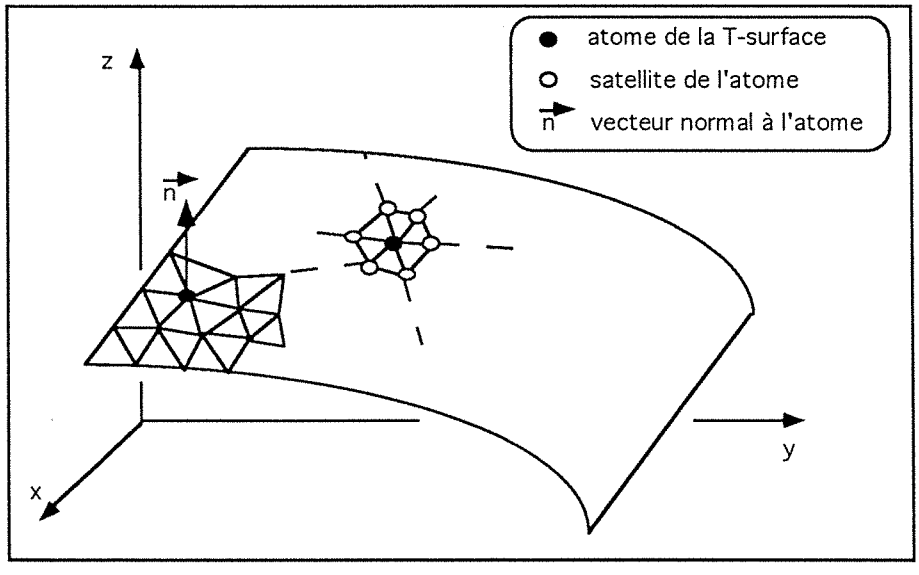


Figure 0.8 Représentation d'une surface dans GOCAD.

Nous décrivons dans ce paragraphe quelques unes des structures de la base de données géométrique de GOCAD (voir Figure 0.8). Sont présentées uniquement les structures de données de

base utilisées dans les développements exposés dans cet ouvrage[1], soit: le **Vertex**, le **Vset**, l'**Atome**, le **Triangle** et la **Surface triangulée** (ou **Tsurf**).

Le Vertex

Un Vertex[2] est un point de l'espace 3D. Diverses informations sont associées à cette structure, parmi lesquelles nous retiendrons:

- les coordonnées de ce point dans un repère euclidien,
- un pointeur vers l'objet propriétaire de ce point.

Le Vset

Un Vset est l'ensemble des vertices d'un même objet GOCAD. Il est constitué:

- d'un ensemble de vertices qui lui appartiennent, et dont l'usage lui est réservé,
- d'un ensemble de vertices qui lui appartiennent, mais qui sont "exportés" vers d'autre(s) objet(s),
- d'un ensemble de vertices qui ne lui appartiennent pas et qui sont "importés" d'autre(s) objet(s).

Nous n'entrerons pas dans le détail de ces procédés d'importation et d'exportation de vertices, décrits de facon exhaustive dans [22] et [27].

L'Atome

Alors que la notion de vertex existe pour décrire la géométrie d'un objet, c'est-à-dire sa position dans l'espace, une autre entité est nécessaire pour décrire la topologie de ce même objet, c'est-à-dire l'ensemble des relations de voisinage qui caractérisent la forme de l'objet. C'est l'objectif de la structure Atome. Un Atome est un noeud du maillage d'un objet, contenant les informations suivantes:

- un pointeur vers le vertex associé,
- un pointeur vers un tableau décrivant l'orbite de cet atome, c'est-à-dire tous les atomes ayant un lien direct avec l'atome concerné (satellites de l'atome),
- le nombre de satellites de l'atome concerné,
- le vecteur normal à la surface dont l'origine est cet atome.

[1] Tous les détails de l'implantation de GOCAD peuvent être trouvés dans le rapport du projet [27].
[2] Le pluriel de vertex et vertices.

Le Triangle

Un Triangle est un élément de base du maillage d'une surface. Il est décrit par:

- ses trois sommets, représentés par trois atomes stockés dans un tableau,
- ses triangles adjacents (de 0 a 3), c'est-à-dire les triangles ayant une arête commune avec ce triangle.

Un triangle est orienté en fonction de l'ordre de rangement de ses trois sommets dans le tableau d'atomes [3]. D'après cette orientation, un vecteur normal peut être calculé pour chaque triangle. Nous considérerons par la suite que tout triangle a un vecteur normal associé. Pour plus de clarté, nous appellerons **edge** chacune des arêtes d'un triangle et **côté** chacune des faces de ce triangle (côté '+' et côté '−', en fonction de l'orientation de son vecteur normal).

La Surface triangulée (ou Tsurf)

Une T-surface (ou Tsurf) est une surface décomposée sous forme d'un ensemble non-ordonné de facettes triangulaires. Les edges de chaque triangle sont les **arêtes** de la T-surface. Les sommets de chaque triangle sont les **noeuds** de la T-surface. Une T-surface peut être composée de plusieurs morceaux déconnectés, chaque morceau pouvant être ouvert ou fermé.

Bord d'une Tsurf

Le bord d'une Tsurf (ou frontière de la Tsurf) et défini par les edges des triangles pour lesquels il n'existe pas de triangle adjacent. Le bord d'une Tsurf peut être en plusieurs morceaux, comme dans le cas d'une Tsurf avec des trous ou une Tsurf en plusieurs parties déconnectées (voir Figure 0.9).

Lors du processus de modélisation, de nombreux cas requièrent la manipulation directe d'une partie B_i du bord d'une surface S. On peut imaginer, par exemple:

- déplacer B_i sans bouger S en vue d'élargir ou de rétrécir S le long de B_i,
- installer des contraintes D.S.I. le long de B_i (par exemple la contrainte On-Tsurf).

Pour permettre de définir des morceaux du bord d'une surface, on a introduit la notion de **Border-Stone** (ou **BStone**). Un **BStone** est un marqueur placé sur un atome quelconque appartenant à la frontière d'une surface. La caractéristique la plus importante de l'installation d'un **BStone** sur une surface est que celui-ci provoque une rupture dans le voisinage des atomes du bord.

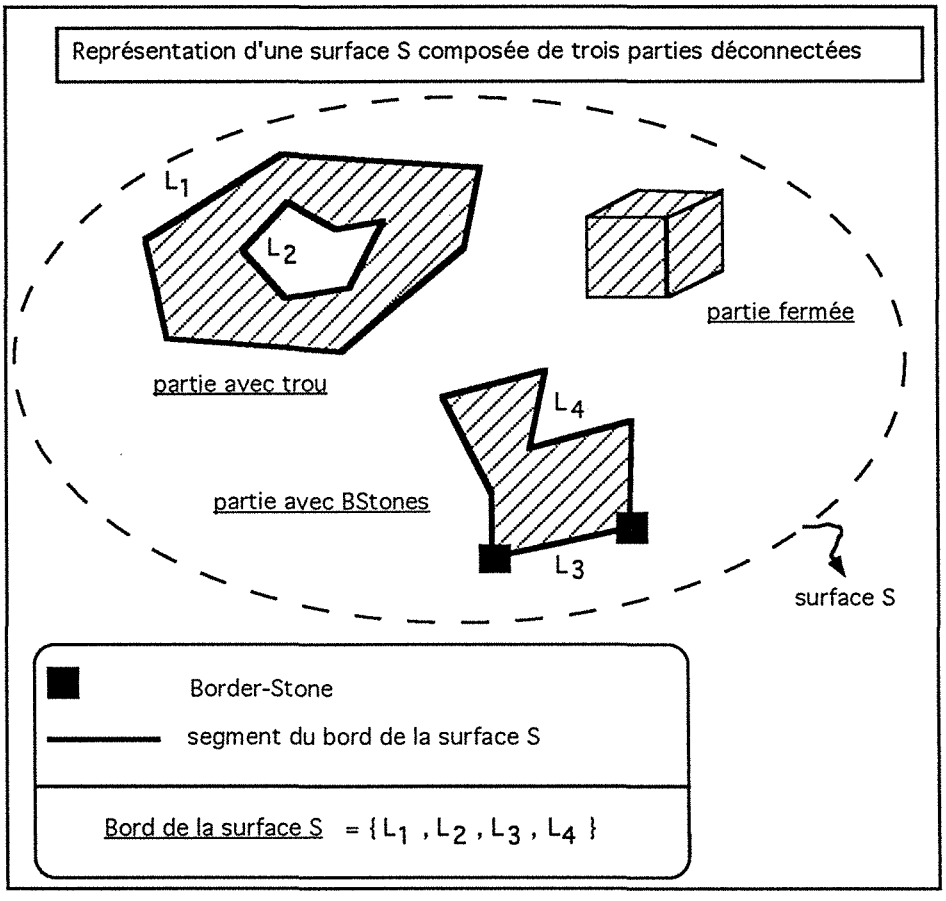


Figure 0.9 Définition du bord d'une surface dans GOCAD.

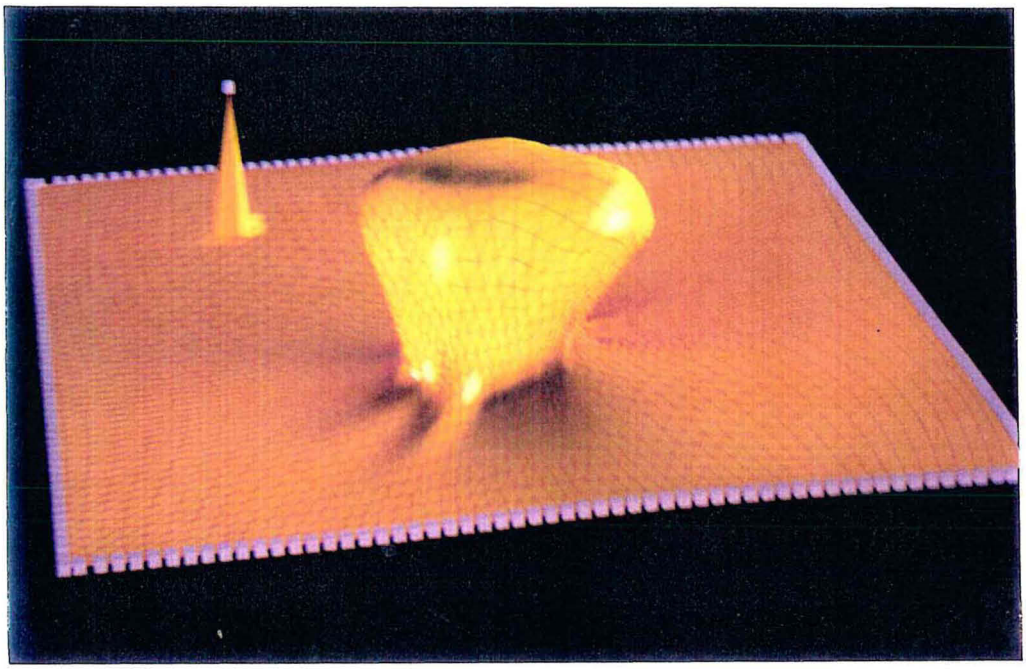


Figure 0.10 Représentation d'un dôme de sel avec GOCAD. Un point a été déplacé et fixé par un Control Node.

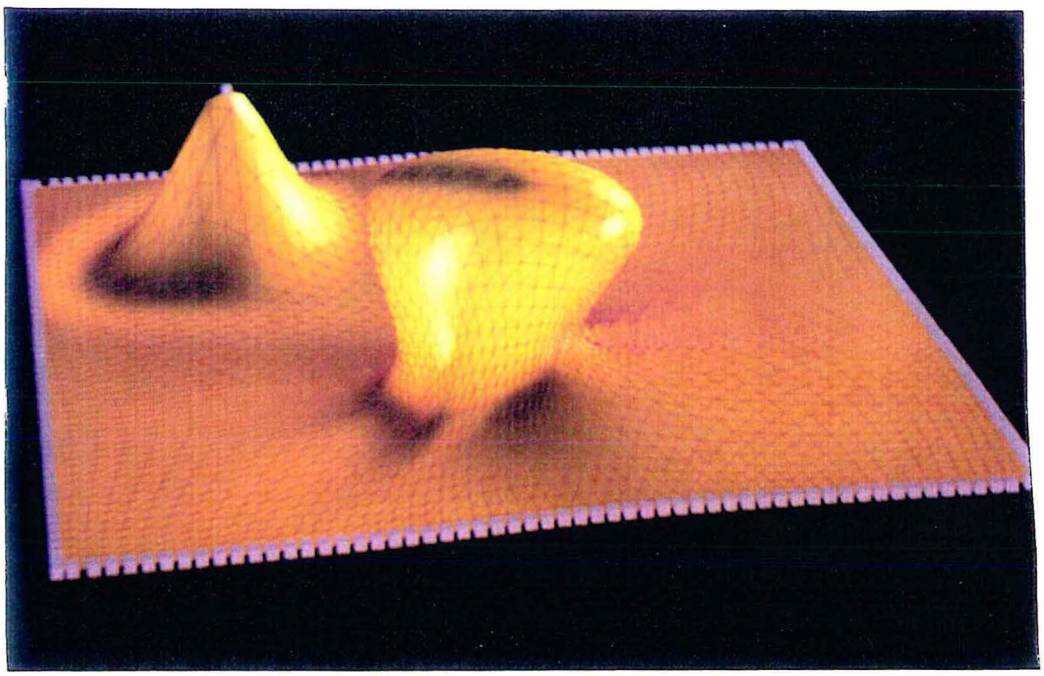


Figure 0.11 Représentation d'un dôme de sel avec GOCAD. Résultat après interpolation avec D.S.I.

PARTIE I

MODELISATION VOLUMIQUE NON-MANIFOLD

Introduction

Dans cette première partie, nous allons nous consacrer à la présentation des structures introduites pour notre système de modélisation volumique *non-manifold*.

Le premier chapitre propose une brève présentation du modèle qui a servi de base de lancement à cette recherche: le modèle de Weiler.

Le second chapitre présente les structures de haut niveau décrivant le modèle volumique, c'est-à-dire la région et la frontière d'une région (ou shell).

Le troisième chapitre expose la théorie des divers éléments introduits pour décrire la topologie des objets utilisés pour créer le modèle volumique.

Enfin le quatrième chapitre décrit les processus automatiques de création et gestion des régions, tous deux basés sur l'utilisation de la topologie des objets modélisés.

Chapitre 1

Modèle de Weiler

1.1 Introduction

Kevin Weiler nous propose la première représentation complète de modélisation par frontières d'objets *non-manifold*, qui décrit de façon explicite des adjacences topologiques: la **Radial-Edge Structure** [38]. C'est une structure de données spécifique pour la modélisation d'un objet, basée sur la représentation de la topologie par frontières *non-manifold*. Elle contient des informations sur les relations d'adjacence.

1.2 Géométrie et Topologie

La **géométrie** est utilisée ici pour représenter essentiellement les informations relatives à la forme géométrique d'un objet; ceci comprend sa position dans l'espace ainsi que la position géométrique précise de tous les aspects de ses différents éléments.

Par définition, la **topologie** est une abstraction, un sous-ensemble cohérent des informations disponibles sur la géométrie d'une forme. C'est l'ensemble des propriétés d'une forme géométrique qui ne changent pas quand la forme est modifiée [31]. Dans le contexte de la modélisation géométrique, la topologie se résume en général aux relations d'adjacence entre des éléments topologiques.

1.3 Eléments topologiques non-manifold

Il existe sept types d'élément topologique de base dans une représentation par frontières d'objets *non-manifold*. Ils sont décrits ici dans l'ordre décroissant de leur rôle dans la hiérarchie de la

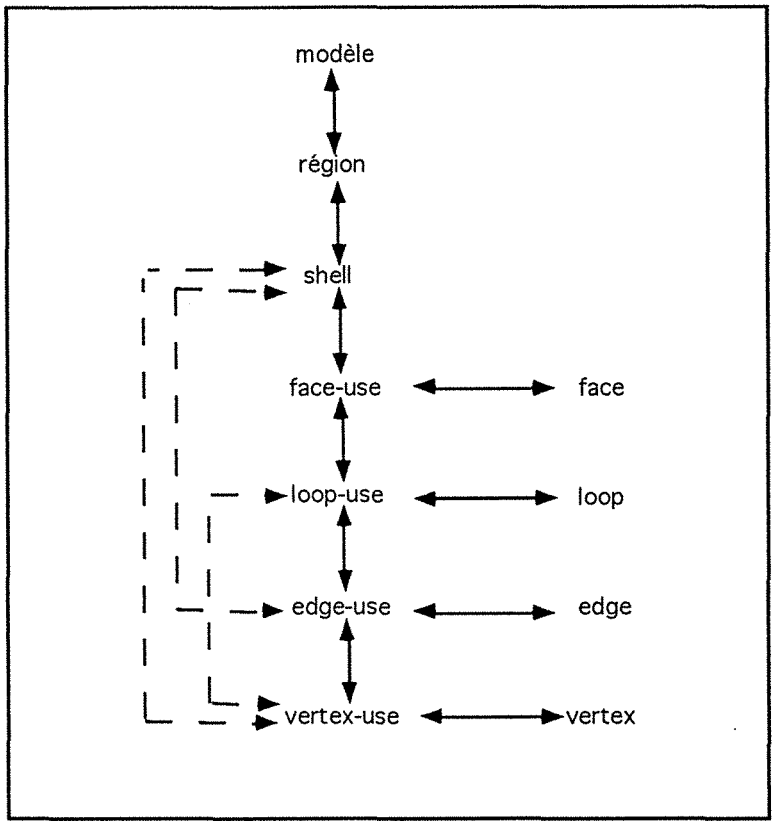


Figure 1.1 Hiérarchie de la **Radial-Edge Structure.**

Radial-Edge Structure[1] (voir Figure 1.1):

- **Le Modèle:**
  Le **modèle** est plongé dans l'espace 3D topologique unique de modélisation. Il est constitué d'une ou plusieurs **régions** distinctes de l'espace 3D et agit comme un dépôt de tous les éléments topologiques contenus dans une représentation de l'espace de modélisation.

- **La Région:**
  Une **région** est un volume de l'espace 3D. Il existe toujours au moins une **région** dans le **modèle** (l'Univers). Pour un objet solide unique, il existe deux **régions** (une à l'intérieur, l'autre à l'extérieur).

- **Le Shell:**
  Un **shell** est la surface-frontière orientée d'une **région**. Il est composé de **faces** orientées. Une **région** peut avoir de 1 à n **shells**, comme dans le cas d'une **région** contenant des sous-**régions**.

[1] Les mots spécifiques à la Radial-Edge Structure sont imprimés dans ce paragraphe en caractères gras.

- **La Face:**
  Une **face** est une partie connexe d'une surface, correspondant à une portion limitée d'un **shell**. Alors qu'une surface globale peut être *non-manifold*, les **faces** individuelles qui la composent sont *manifold*.

- **Le Loop:**
  Un **loop** est la frontière connectée d'une **face** unique. Une **face** peut avoir de 1 à n **loops**, comme dans le cas d'une **face** avec des trous.

- **L'Edge:**
  Un **edge** est un ensemble connexe de points formant une courbe spatiale limitée par deux **vertices**. C'est l'élément de base d'un **loop**.

- **Le Vertex:**
  Un **vertex** est un point unique de l'espace 3D.

Pour la **Radial-Edge Structure**, Weiler a introduit quatre types d'élément supplémentaires, orientés de façon plus spécifique vers la description des utilisations topologiques de quelques uns des éléments présentés précédemment (la **face**, le **loop**, l'**edge** et le **vertex**). Le nom de chacun de ces nouveaux éléments est obtenu en ajoutant le suffixe **use** au nom de la structure à laquelle il est rattaché. Ces **élément-uses** sont:

- **La Face-use:**
  Une **face-use** correspond à l'un des deux côtés d'une **face**. Les **face-uses** (utilisation de la **face** par un **shell**) sont orientées en respectant la géométrie de la **face**. Les **face-uses** fonctionnent par paire (une pour le côté positif de la **face**, l'autre pour son côté négatif). Chaque élément de ce couple est appelé **face-use mate** de l'autre élément.

- **Le Loop-use:**
  Le **loop-use** correspond à l'une des utilisations d'un **loop**, associée à une **face-use**. Le **loop-use** est orienté en fonction de la **face-use** à laquelle il est associé. Il existe deux **loop-uses** pour un **loop** donné.

- **L'Edge-use:**
  Un **edge-use** est l'edge orienté d'un **loop-use**. L'**edge-use** est orienté en fonction de l'**edge** auquel il est associé. Un **edge** donné peut être partagé par plusieurs **faces**. Chaque **face** ayant deux **face-uses**, on en déduit que tout **edge** peut avoir $(2 * N)$ **edge-uses**, où N est le nombre de **faces** partageant cet edge $(N \geq 1)$.

  Les **edge-uses** fonctionnent par paire (un pour chaque **loop-use** d'une **face-use**). Chaque élément de ce couple est appelé **edge-use mate** de l'autre élément.

  A un **edge-use** donné EU_1 d'un **loop-use** LU_1 peut également être associé un autre **edge-use** EU_2 du même **edge**, mais associé à un **loop-use** différent LU_2. Cette association permet de décrire les adjacences entre les **face-uses**. Le pointeur qui réalise cela s'appelle **radial**, d'où le nom de cette représentation (**Radial-Edge Structure**).

- **Le Vertex-use:**
  Le **vertex-use** représente l'utilisation d'un **vertex** par un **edge**, un **loop** ou un **shell**.

1.4 Définitions utiles

Nous donnons ici quelques définitions utiles à une bonne compréhension de la suite:

- **deux éléments topologiques sont connectés**
  s'il existe un chemin constitué d'edges et de vertices adjacents conduisant d'un élément à l'autre.

- **fermeture d'une région (avec une nouvelle face):**
  la création de la **face** a subdivisé une région existante en deux régions distinctes; ie il n'est pas possible de connecter un point interne à l'une des deux nouvelles régions et un point interne à l'autre sans traverser une **face**, un **edge** ou un **vertex**.

1.5 Exemples d'utilisation de la Radial-Edge Structure

Nous présentons ici brièvement deux exemples décrivant la façon dont les éléments topologiques présentés dans le paragraphe précédent sont utilisés par la représentation **Radial-Edge**.

1.5.1 Exemple 1

Sur la Figure 1.2, les structures suivantes sont définies:

- **Face:**
  L'exemple comporte deux **faces** partageant le même **edge**.

- **Face-use:**
  Il existe deux **face-uses** pour chaque **face** (une pour chaque côté de la **face**), soit un total de quatre **face-uses** pour le modèle.

- **Edge:**
  Plusieurs **edges** existent dans le modèle présenté. En fait, la frontière de chaque **face** est composée de quatre **edges**. Nous nous intéresserons uniquement à l'**edge** partagé par les deux **faces**.

- **Edge-use:**
  Il existe deux **edge-uses** pour chaque utilisation de l'edge commun par une **face**, soit un **edge-use** par **face-use**. Sachant que deux **faces** partagent le même **edge**, il existe donc quatre **edge-uses** pour le modèle.

De plus, le schéma décrit la façon dont ces quatre **edge-uses** sont connectés (pointeurs **mate** et **radial**).

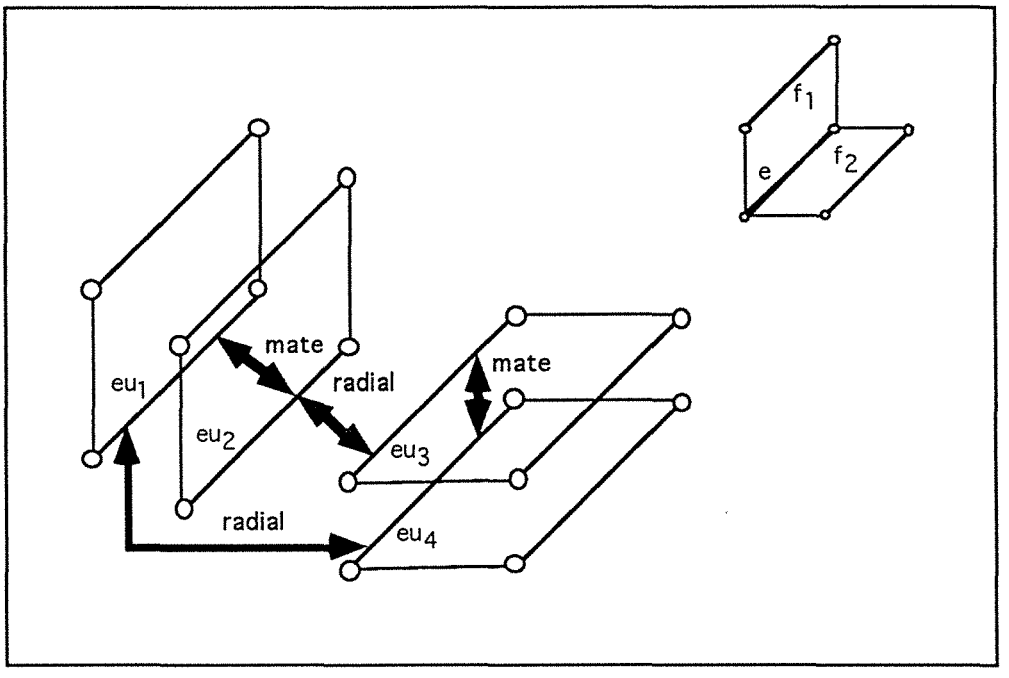


Figure 1.2 Représentation de deux faces se joignant le long d'un edge commun.

1.5.2 Exemple 2

Le second exemple, illustré par la figure 1.3, est présenté, pour plus de clarté, sous-forme d'une coupe 2D. Sur ce schéma, les noeuds correspondent donc à des **edges**, et les segments à des **faces**. On distingue diverses structures:

- **Face:**
  L'exemple comporte trois **faces** partageant le même **edge** e_1.
- **Face-use:**
  Il existe deux **face-uses** pour chaque **face** (une pour chaque côté de la **face**), soit un total de six **face-uses** pour le modèle.
- **Edge:**
  L'exemple ne présente qu'un seul **edge** e_1, partagé par les trois **faces** en présence.
- **Edge-use:**
  Il existe deux **edge-uses** pour chaque utilisation de l'**edge** e_1 par une **face**, soit un **edge-use** par **face-use**. Sachant que trois **faces** partagent cet **edge**, il existe donc un total de six **edge-uses** pour le modèle.

Le schéma décrit également les différentes relations établies entre chacune des structures topologiques présentes (pointeurs **mate** et **radial**).

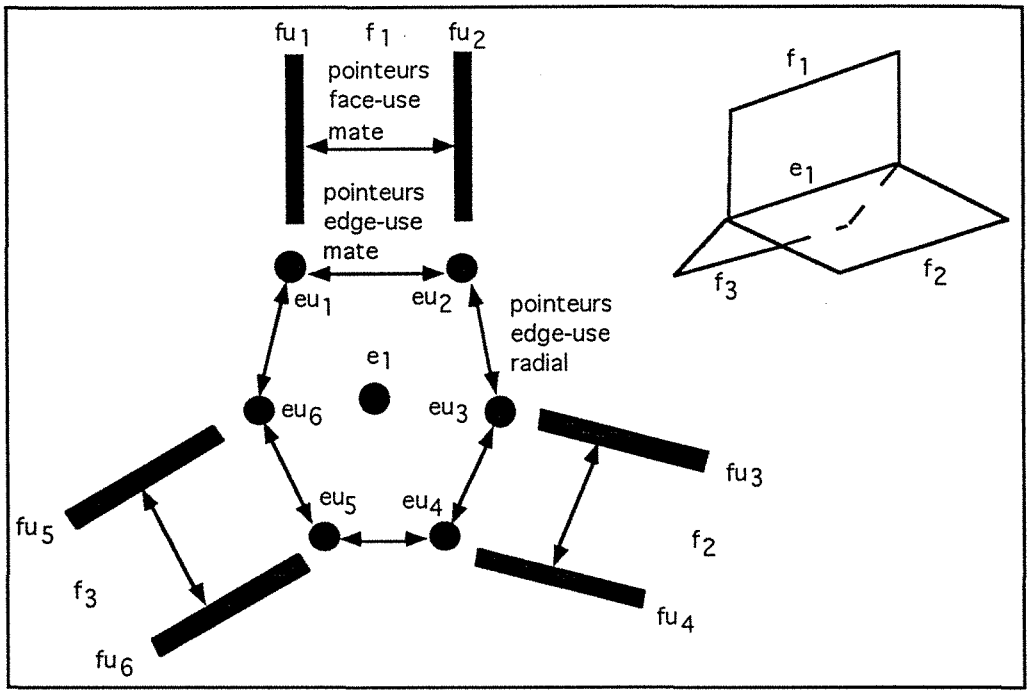


Figure 1.3 Représentation 2D de trois faces partageant un edge commun.

1.6 Opérateurs

Weiler propose des opérateurs de bas-niveau (**Non-Manifold Topology Operators**) pour la construction, la maintenance et l'utilisation des éléments topologiques d'une représentation *non-manifold* d'un objet [40]. Ces opérateurs ont été conçus de façon compatible avec un ensemble d'opérateurs *manifold* existants, les opérateurs d'Euler.

1. **Opérateurs de construction:**

   - M*suitedelettres* = Make
     Permet de créer une nouvelle instance d'un élément topologique.
   - K*suitedelettres* = Kill
     Permet de supprimer une instance existante d'un élément topologique.
   - G*suitedelettres* = Glue
     Permet d'associer deux instances d'un même type d'élément topologique. L'exemple le plus fréquent est celui de l'association de deux éléments de type **face** dans le but de fermer une région (voir Paragraphe 1.4). Cette opération de **Glue** conduit à la création de nouvelles relations d'adjacence entre les deux éléments associés.

   suitedelettres symbolise le type de l'élément sur lequel l'opérateur doit agir (**modèle, région, shell...**).

2. **Opérateurs d'accès:**

   - Queries (demandes):
     Accès uniques aux informations sur la relation d'adjacence, pour déterminer un élément du groupe adjacent à un élément spécifié.
   - Traversals (parcours):
     Queries répétées pour déterminer tous les membres appartenant à un groupe adjacent d'une relation d'adjacence.

1.7 Suffisance de la représentation

Par définition, la suffisance topologique d'une représentation est la possibilité de représenter de façon **complète** et **non-ambigüe** les adjacences topologiques d'un modèle donné.

1.7.1 Complétude

Une représentation est dite **complète** si toutes les relations d'adjacence sont dérivables des données de la structure.

1.7.2 Non-ambigüité

Un représentation est dite **non-ambigüe** si, pour un ensemble de données de la représentation, il existe un et un seul ensemble d'informations topologiques pouvant résulter de l'interprétation de la représentation. En d'autres termes, il doit exister une bijection entre la représentation et l'ensemble des informations topologiques.

1.7.3 Suffisance de la Radial-Edge Structure

La représentation fondée sur la **Radial-Edge Structure** proposée par Weiler est suffisante, la preuve en est fournie dans [39].

1.8 Conclusion

La modélisation géométrique *non-manifold* utilisant une représentation par frontières et des structures spécifiques décrivant les relations d'adjacence constitue un progrès considérable dans la technologie de la modélisation par frontières. En effet, ce type de représentation présente deux avantages principaux, à savoir:

1. Il regroupe les avantages des trois méthodes courantes de modélisation (Fil de Fer, Surfacique et Volumique), c'est-à-dire qu'il dispose d'informations sur:

   - les faces d'un objet,
   - les courbes et points qui bordent ces faces,

- la topologie des divers éléments composant cet objet.

2. L'accroissement du nombre de modèles pouvant être représentés du fait de la prise en compte des objets *manifold* et *non-manifold.*

La structure **Radial-Edge** constitue une représentation précise et complète (preuve dans [39]) de modèles géométriques *non-manifold*, vérifiant tous les critères d'une bonne modélisation volumique (voir Introduction, Paragraphe 0.1.1).

Dans sa conception première, c'est-à-dire telle qu'elle est présentée par K. Weiler dans [38], elle n'est cependant pas adaptée au cadre de notre recherche. En effet, la base de données relative à la structure **Radial-Edge** est très lourde est comporte un grand nombre d'informations redondantes. Cet inconvénient peut être contourné dans notre recherche, par le simple fait que les surfaces utilisées sont représentées à l'aide d'un maillage triangulaire.

Chapitre 2

Modélisation volumique dans GOCAD

2.1 Introduction

Notre système de modélisation volumique *non-manifold* consiste à découper l'espace de modélisation (l'Univers) en une série de volumes fermés distincts. Le modèle est donc constitué des divers éléments utilisés pour obtenir ce découpage. Le premier de ces éléments est le volume, appelé **Région**. Il est défini par l'ensemble de ses frontières, ou **Shells**. Ces volumes sont obtenus grâce à des structures décrivant les relations d'adjacence entre les divers éléments composant l'objet modélisé.

Nous proposons dans ce chapitre la définition des structures de haut niveau composant notre modèle: la *Région* et le *Shell*. Une autre structure, le **Layer**, a également été introduite pour permettre le regroupement de *Régions* partageant certaines propriétés. Le *Layer* n'est pas à proprement parler une structure essentielle dans la définition du modèle. Il doit plutôt être considéré comme un outil permettant une meilleure utilisation des *Régions* dans le but de faciliter l'interprétation du modèle par l'utilisateur.

2.2 Le volume de l'espace 3D: la Région

Une *Région* est un volume fermé de l'espace 3D. Elle est définie par ses frontières, ou *Shells*. Si une *Région* a plusieurs *Shells*, alors il en existe toujours un contenant tous les autres. Ce *Shell* est dit **externe**, les autres étant des *Shells* **internes**. Une *Région* a un seul *Shell* externe et peut avoir de 0 à n *Shells* internes. En fait, il existe autant de *Shells* internes pour une *Région* donnée que de sous-*Régions* contenues dans cette *Région*. Une *Région* est définie par (voir Figure 2.1):

- son nom,

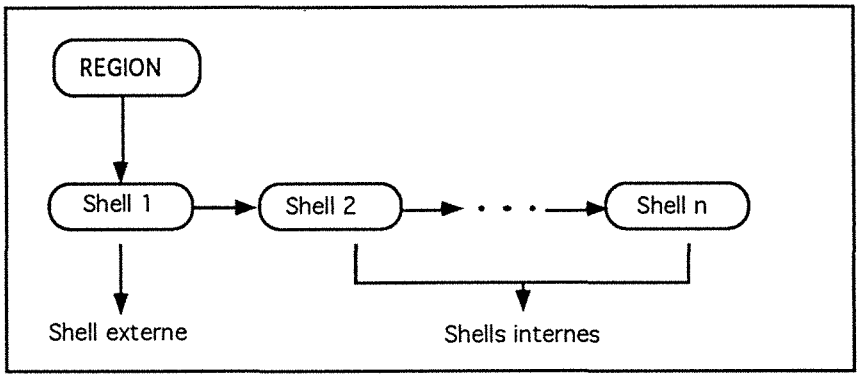


Figure 2.1 La structure *Région*.

- l'ensemble de ses *Shells*,
- une référence vers le *Layer* auquel cette *Région* appartient.

Cas Particulier: L'Univers:

Une première *Région* est créée lors de l'initialisation de notre modèle. Elle correspond à l'Univers (l'espace 3D de modélisation). On imagine très bien que cette *Région* n'a pas de *Shell* externe, l'Univers étant par définition sans limite. Cette *Région* spéciale ne comporte en fait que des *Shells* internes, correspondant aux frontières des *Régions* qui se découpent dans l'Univers.

2.3 La frontière d'un volume: le Shell

Le *Shell* est une frontière d'une *Région* donnée. On peut le considérer comme un ensemble de faces[1] adjacentes délimitant une *Région*. Un *Shell* donné ne peut être la frontière que d'une seule et unique *Région*. Or une face ayant deux côtés, elle peut délimiter deux *Régions* différentes. On introduit ici une notion fondamentale dans notre système de modélisation, à savoir que les faces sont toujours utilisées en association avec une orientation. En conséquence, un *Shell* est un ensemble de faces orientées adjacentes délimitant une *Région*. On distingue deux types de *Shells*:

1. *Shell* **externe:** (voir Figure 2.2)
   Ce *Shell* correspond à la frontière de la *Région* qui contient tous les autres *Shells* de cette même *Région*.

2. *Shell* **interne:** (voir Figure 2.3)
   Ce *Shell* correspond à une **frontière interne d'une** *Région* pouvant être:

[1] La notion de face sera développée plus en détail dans le chapitre suivant. Pour le moment, on considère une face comme un sous-ensemble connexe d'une surface.

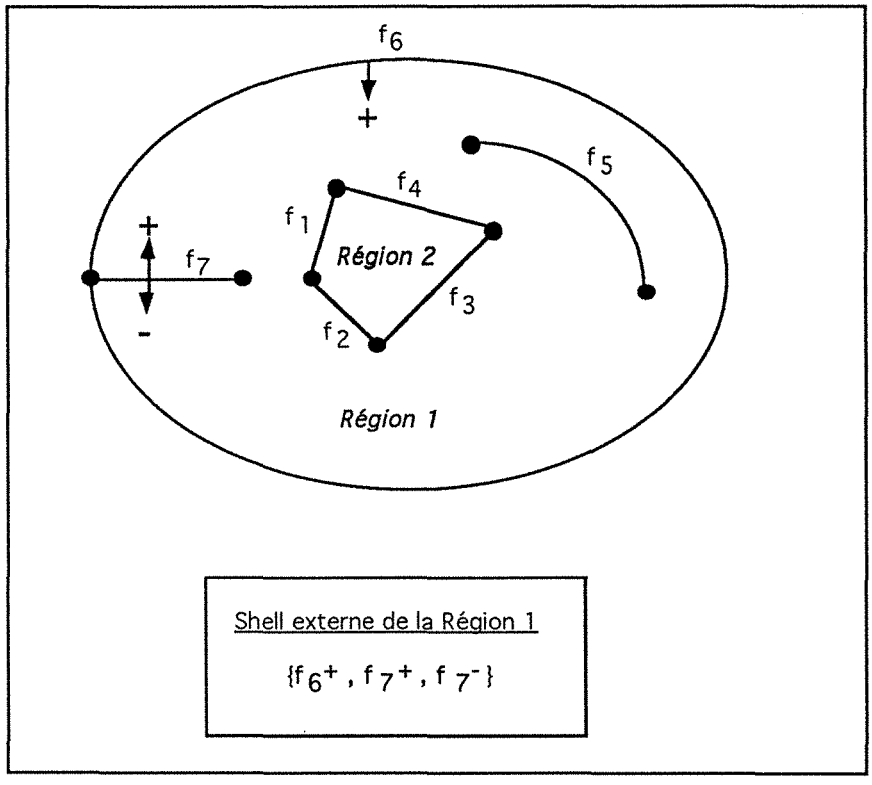


Figure 2.2 Exemple de *Shell* externe (Représentation 2D).

- la frontière d'une sous-*Région* incluse dans cette *Région* (par exemple frontière de la *Région* 2 sur la figure 2.3).
- les deux côtés d'une face isolée contenue par cette *Région* (par exemple les côtés '+' et '−' de la face f_5 sur la figure 2.3).

Un *Shell* est défini par:

- son type (1: **interne**; 0: **externe**),
- l'ensemble des faces orientées qui le composent,
- une référence vers la *Région* dont il constitue tout ou partie de la frontière.

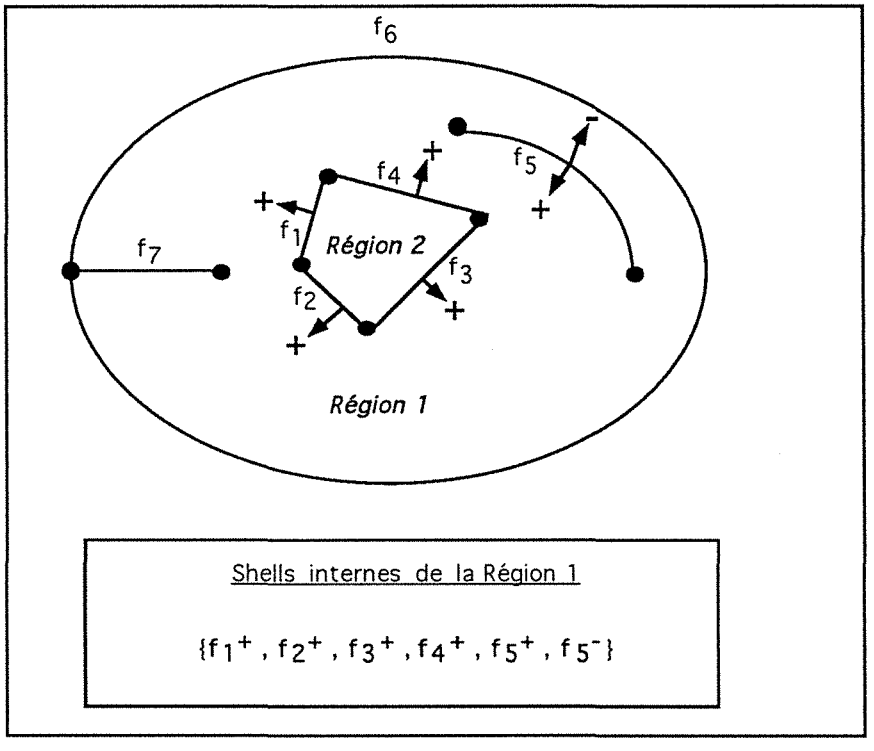


Figure 2.3 Exemple de *Shells* internes (Représentation 2D).

2.4 L'élément spécifique à une application géologique: le Layer

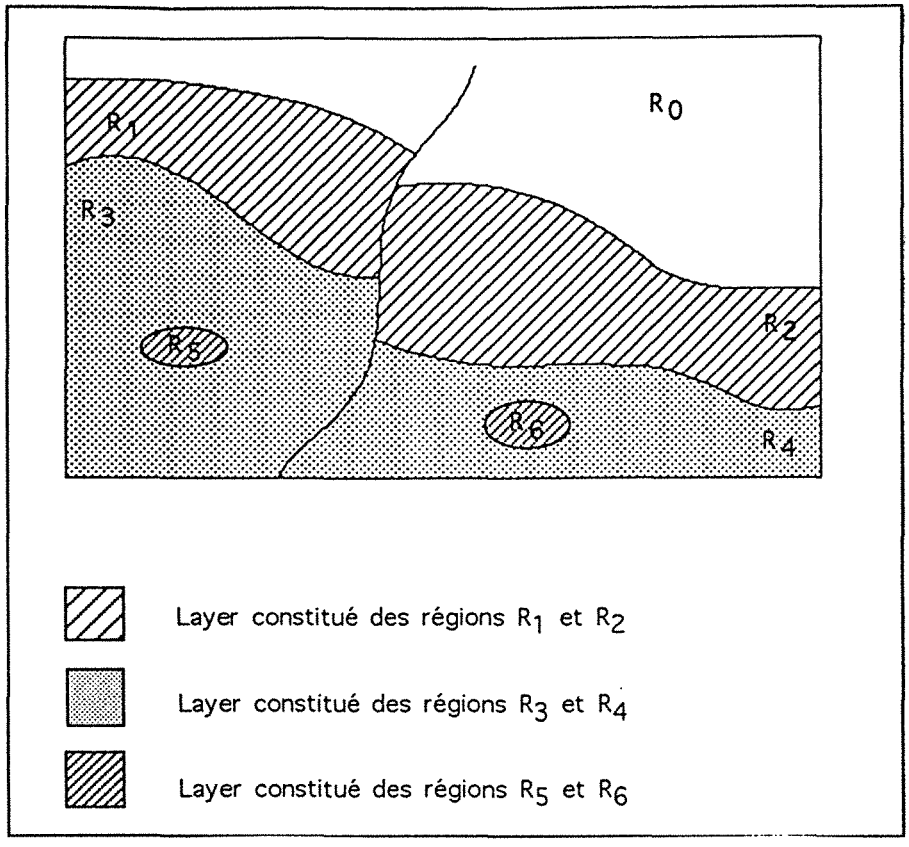


Figure 2.4 Exemple de *Layers* (Représentation 2D).

2.4.1 Définition

Comme le montre la figure 2.4, un *Layer* (ou couche géologique) est un ensemble de *Régions* partageant certaines propriétés. Celles-ci peuvent être:

1. Propriétés graphiques: couleur, texture...
2. Propriétés physiques: porosité, densité, vitesse...
3. Age géologique.
4. ...

Cette structure a été introduite pour répondre à un besoin spécifique à des applications dans le domaine de la Géologie, à savoir la notion de **couche**. Lors de la création d'une *Région*, un nouveau *Layer* lui est automatiquement attribué. C'est ensuite à l'utilisateur de décider, en

fonction de son interprétation du modèle, quelles *Régions* il désire grouper. Ce traitement ne peut être fait de façon automatique puisqu'il dépend entièrement des connaissances du géologue-utilisateur. Un *Layer* est composé de (voir Figure 2.5):

- son nom,
- l'ensemble des *Régions* qui le composent.

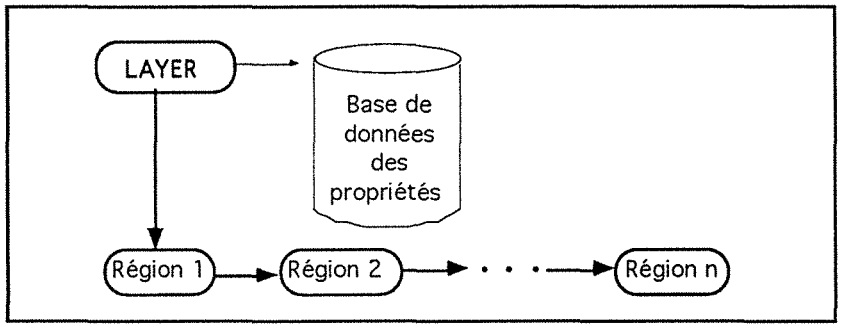


Figure 2.5 La structure *Layer*.

2.4.2 Gestion des Layers

Un procédé de gestion interactive des *Layers* a été développé par O.Mariez [29] au cours de son stage de D.E.A. Un *Layer* peut être vu à deux niveaux:

1. le regroupement de plusieurs *Régions* faisant partie d'une même couche géologique.
2. une couche géologique elle-même subdivisée en sous-couches. Grâce à cette récursivité, l'utilisateur peut choisir le niveau d'observation correspondant à son analyse.

La solution adoptée pour gérer les *Layers* est une représentation sous forme d'arbres formant une "forêt". Les opérations de manipulation des *Layers* sont alors les suivantes:

- **Ajout d'un *Layer*:**
  cette opération consiste à ajouter un noeud dans l'arbre à l'endroit désigné par l'utilisateur.
- **Retrait d'un *Layer*:**
  cette opération consiste à supprimer un noeud donné dans l'arbre.
- **Déplacement d'un *Layer*:**
  cette opération consiste à déplacer un noeud donné de l'arbre vers une position choisie par l'utilisateur.
- **Gestion automatique des *Layers* à afficher:**
  l'affichage des *Layers* est contrôlé en fonction de la sélection de l'utilisateur.

- **Modification des caractéristiques d'un *Layer*:**
  le nom et la couleur d'un *Layer*, attribués automatiquement lors de la création de celui-ci, peuvent être modifiés à tout moment. Le nom peut être toute chaine alphanumérique; la couleur est choisie dans une palette proposée.

2.5 Conclusion

Dans notre système de modélisation volumique, nous considérons une *Région* comme un ensemble de *Shells*, où chaque *Shell* est un ensemble de faces orientées. Pour définir une *Région*, il nous faut donc trouver les faces orientées adjacentes composant chacun de ses *Shells*. Pour cela, il est nécessaire de connaitre les adjacences des faces, donc la topologie de l'objet modélisé.

Chapitre

3

Description de la topologie inter- et intra-objets

3.1 Introduction

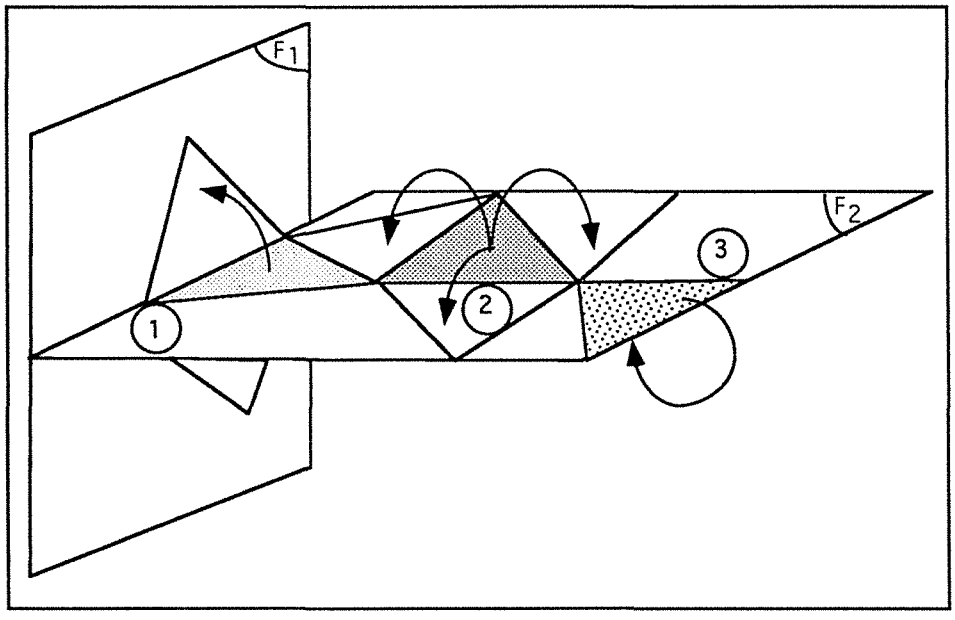


Figure 3.1 Relations d'adjacence d'un triangle.

Les surfaces utilisées dans cette recherche sont représentées sous forme d'un ensemble de triangles adjacents. Pour notre système de modélisation volumique, nous introduisons un nouveau concept: la *Face*, que nous avons défini précédemment comme un sous-ensemble connexe d'une surface[1]. Une modélisation par maillage triangulaire implique que la description de la topologie d'un modèle doit apparaitre non seulement au niveau des *Faces* (**Macro-Topologie**), mais également au niveau des triangles (**Micro-Topologie**). Les relations d'adjacence entre les triangles (voir Figure 3.1) sont décrites par une structure appelée **Tlink**, ou **Triangle-Link**. Un *Tlink* apparait à deux niveaux:

- Niveau Intra-*Face* (cas 2 et 3 sur la figure 3.1):
  Description des connexions entre deux triangles d'une même *Face*.
- Niveau Inter-*Face* (cas 1 sur la figure 3.1):
  Description des connexions entre deux triangles de deux *Faces* différentes.

Les différents éléments topologiques introduits pour résoudre notre problème de modélisation volumique *non-manifold* sont exposés dans les paragraphes suivants.

3.2 La structure Face

3.2.1 Définition

Une **Face** est un ensemble de triangles connectés d'une surface. En d'autres termes, chaque partie connexe d'une surface constitue une *Face*. Le bord d'une *Face* est déterminé par tout ou partie du bord de la surface dont elle est originaire. Lors de la présentation de la structure *Shell*, nous avions noté que les *Faces* en elles-mêmes ne sont pas fondamentales dans notre système de modélisation, mais plutôt chacun des côtés de cette *Face*. C'est pourquoi une *Face* est toujours utilisée en association avec une orientation ('+' pour le côté positif de cette *Face*; '−' pour son côté négatif). Par la suite, nous nommerons un couple (*Face*, orientation) une **Face-use**. Une *Face* donnée a deux *Face-uses* (voir Figure 3.2).

3.2.2 Orientation des Faces

Tous les triangles d'une *Face* donnée F sont supposés avoir leurs vecteurs normaux orientés de façon consistante[2]. Cette orientation est telle que:

un observateur marchant sur la *Face* voit toutes les normales orientées dans la même direction:

- sens de la normale: côté '+',
- sens oppposé au sens de la normale: côté '−'.

[1] En fait, comme vous le remarquerez par la suite, le concept de surface n'apparait plus qu'au niveau de l'utilisateur. C'est la partie "visible" du système. De façon interne, seules les *Faces* sont utilisées et gérées.

[2] L'orientation d'une *Face* est entièrement dépendante de celle des triangles qui la composent. C'est pourquoi tous les triangles d'une même *Face* doivent impérativement être orientés de façon consistante.

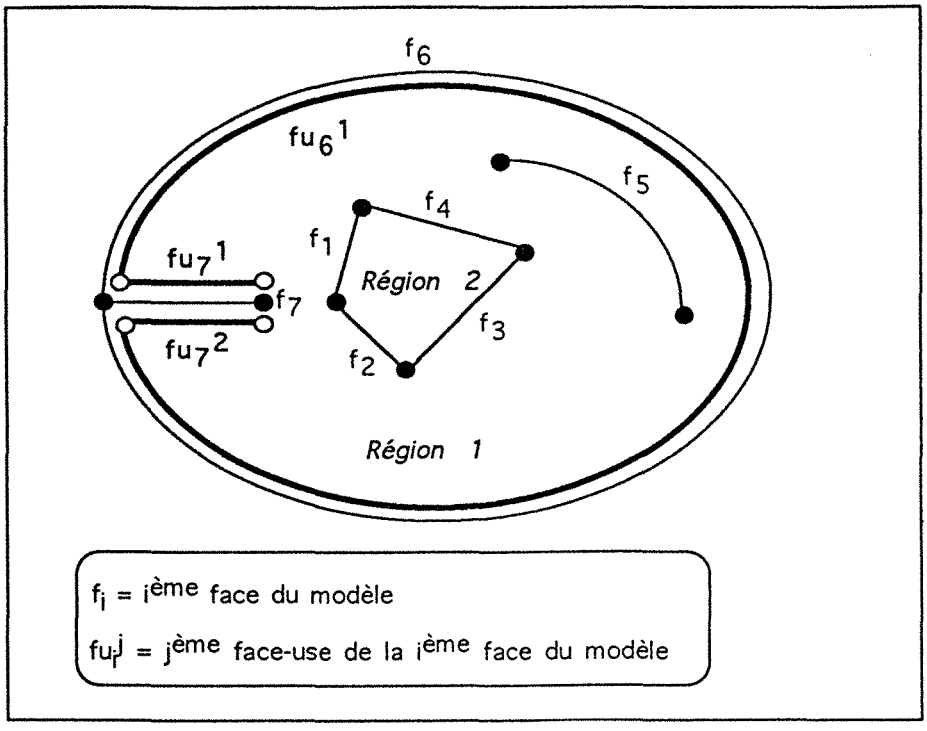


Figure 3.2 Définition d'une *Face-use* (Représentation 2D).

Ce vecteur normal est utile pour faire la différence entre les deux côtés d'une *Face* donnée (côtés positif et négatif), donc pour définir les deux *Face-uses* d'une *Face*.

3.3 La structure Tlink

Dans notre contexte de géométrie *non-manifold*, la topologie est la description des relations d'adjacence entre les *Faces*. Des informations d'adjacence doivent donc être attachées aux frontières des *Faces*. Une *Face* étant un ensemble de triangles connectés, chacun d'eux est susceptible d'avoir des relations d'adjacence. La description de cette **Micro-Topologie** est réalisée par la structure **Tlink**. Il est à noter que les *Tlinks* associés aux segments du bord de la *Face* sont définis au moment de la création de la *Face*.

3.3.1 Triangle-use

Un triangle peut avoir 3 types de relation d'adjacence (voir Figure 3.1):

- **Type 1:**
  Adjacence d'un triangle avec un autre appartenant à une *Face* différente (niveau **Inter-*Face***).

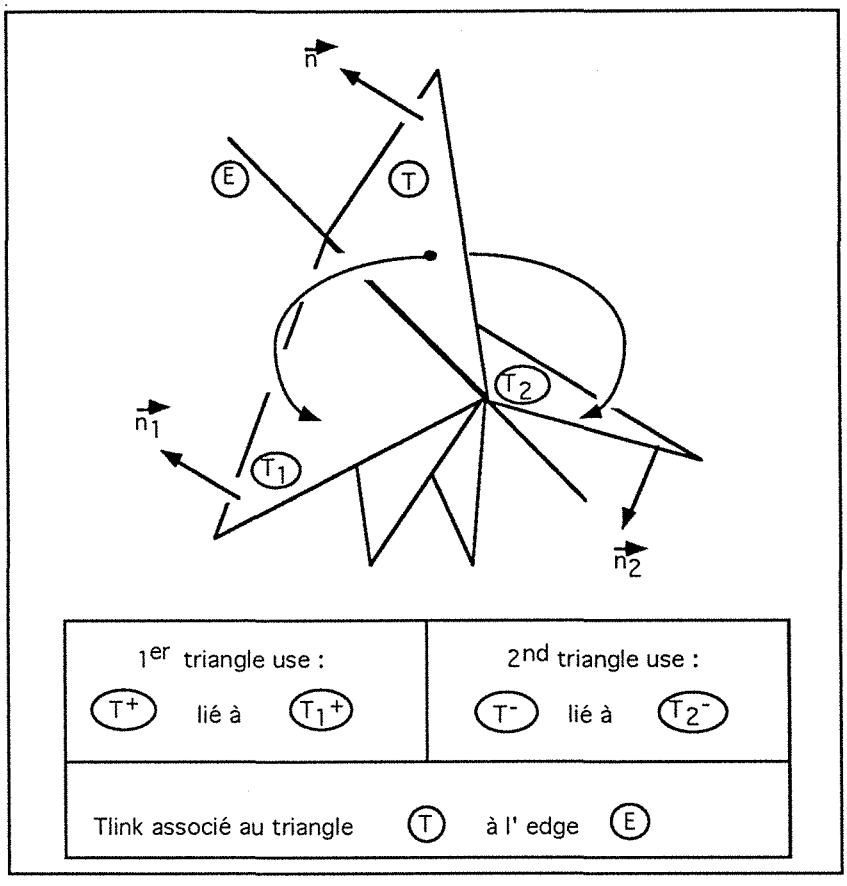


Figure 3.3 Définition du *Tlink*.

- **Type 2:**
  Adjacence d'un triangle avec un autre appartenant à la même *Face* (niveau **Intra-***Face*).

- **Type 3:**
  Adjacence d'un triangle avec lui-même (niveau **Intra-***Face*).

Dans notre cas, les seules relations d'adjacence intéressantes sont celles apparaissant au niveau des frontières des *Faces*. C'est pourquoi seuls les types d'adjacence 1 et 3 sont pris en compte. Tout triangle T a deux côtés notés '+' et '−', selon l'orientation de son vecteur normal. Par définition, on appelle *Triangle-use* d'un triangle T, et on note

$$[T, side]$$

le couple composé de T et de l'un de ses côtés ($'+'$ ou $'-'$). D'après cette définition, tout triangle T a deux *Triangle-uses*:

$$T \Rightarrow \begin{cases} triangle - use_1 = [T,'+'] \\ \\ triangle - use_2 = [T,'-'] \end{cases}$$

Les *Triangle-uses* fonctionnent toujours par paire ($[T\ ,'+']\ ,\ [T\ ,'-']$). Chaque élément de cette paire est appelé **Triangle-use mate** de l'autre élément.

3.3.2 Tlink

La structure *Tlink* est utilisée pour décrire l'adjacence d'un triangle du bord d'une *Face*. Pour chaque edge E d'un triangle T, on appelle *Triangle-Link* (ou *Tlink*), et on note

$$Tlink(T,E) = \{[T_1, side_1], [T_2, side_2]\}$$

le couple composé de deux *Triangle-uses* $[T_1\ ,\ side_1]$ et $[T_2\ ,\ side_2]$ tels que (voir Figure 3.3):

1. $E = T \cap T_1 = T \cap T_2$
2. Un observateur marchant sur le côté $'+'$ du triangle T et traversant l'edge E arrive sur le côté $side_1$ du triangle T_1.
3. Un observateur marchant sur le côté $'-'$ du triangle T et traversant l'edge E arrive sur le côté $side_2$ du triangle T_2.

On dira que:

- $[T\ ,'+']$ est lié à $[T_1\ ,\ side_1]$.
- $[T\ ,'-']$ est lié à $[T_2\ ,\ side_2]$.

Tout triangle a trois edges et chacun d'eux est susceptible d'avoir une relation d'adjacence de type 1 ou 3. Il en résulte qu'un triangle a au plus trois *Tlinks* associés. La Figure 3.4 montre divers exemples de *Tlinks*.

3.3.3 Face-use

Par analogie avec le *Triangle-use* (côté d'un triangle en fonction de son vecteur normal), on nomme *Face-use* d'une *Face* F, et on note

$$[F, side]$$

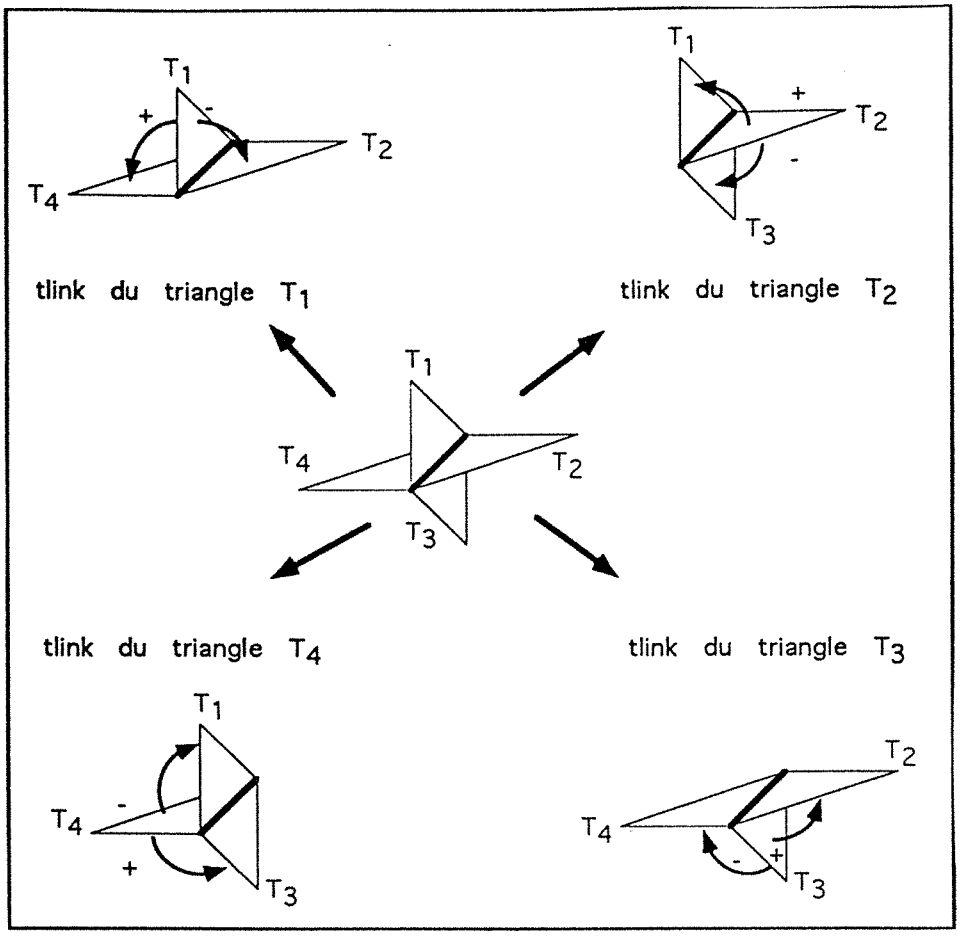


Figure 3.4 Exemple de *Tlink*s.

le couple composé de F et d'un de ses côtés ('+' ou '−'). Toute *Face* F a deux *Face-uses*:

$$F \Rightarrow \begin{cases} face-use_1 = [F,'+'] \\ \\ face-use_2 = [F,'-'] \end{cases}$$

Les *Face-uses* fonctionnent toujours par paire ($[F\ ,'+']\ ,\ [F\ ,'-']$). Chaque élément de cette paire est appelé **Face-use mate** de l'autre élément.

3.3.4 Cas d'une Face fermée isolée

Soit F une *Face* créée à partir d'une surface fermée en une seule partie. F est elle-même fermée, c'est-à-dire qu'elle n'a pas de bord. Or, comme nous l'avons dit précédemment, les relations

d'adjacence intéressantes apparaissent au niveau du bord d'une *Face*. Donc, dans le cas d'une *Face* fermée isolée, c'est-à-dire sans aucune relation d'adjacence avec d'autres *Faces*, aucune structure *Tlink* n'est nécessaire[3].

3.4 Conclusion

L'utilisation de la topologie dans une modélisation volumique par frontières permet de simplifier les algorithmes de manipulation et d'augmenter leur efficacité. Cependant, la topologie peut être encore plus utile si elle est utilisée comme le coeur d'un système de modélisation volumique, comme c'est le cas dans notre système. L'utilisation de la topologie comme centre d'un système de modélisation volumique apporte de nombreux avantages:

1. **Economie de temps:**
   Il n'est pas nécessaire de visualiser en détail toutes les données associées à un modèle.

2. **Connaissance des adjacences:**
   Lors de manipulations locales d'une petite portion d'un objet, il est utile de pouvoir trouver les portions adjacentes d'un objet sans avoir à examiner toutes les données associées à l'objet.

3. **Indépendance entre la géométrie et la topologie:**
   La topologie doit rester indépendante de la représentation géométrique, de façon à ce que toute modification de la géométrie n'affecte pas le système à base topologique.

4. **Imprécision géométrique:**
   La combinaison d'un système à base topologique et d'une technique de représentation de surfaces produit un moyen de représenter les propriétés d'un objet en dépit d'éventuelles imprécisions géométriques.

5. **Simplification des manipulations du modèle:**
   L'utilisation de la topologie simplifie la création, vérification et analyse du modèle solide.

3.4.1 Hiérarchie du modèle

On peut diviser les entités introduites dans notre système de modélisation volumique en deux groupes:

1. **Objets topologiques de base:**
   Cet ensemble regroupe toutes les entités utiles à la description précise des relations d'adjacence entre les différents objets présents. L'utilisateur ne peut agir directement sur les entités de ce groupe de par le fait que celles-ci sont créées et gérées de façon interne. Cette **"partie cachée" du modèle** comporte: la *Face*, le *Tlink*, le *Triangle-use* et la *Face-use*.

[3] Le cas particulier des *Faces* fermées sera développé dans le chapitre 4, paragraphe 4.4.

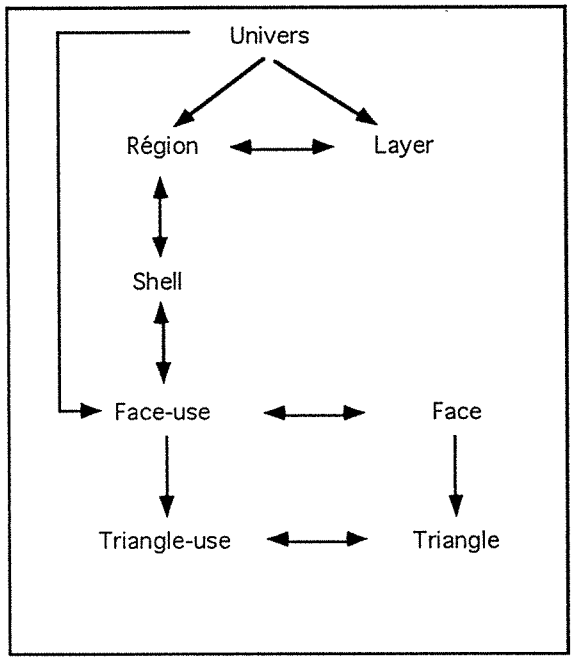


Figure 3.5 Hiérarchie des structures du modèle.

2. **Objets topologiques complexes:**
   Les éléments de ce groupe utilisent les objets topologiques de base pour leur définition. Dans cet ensemble, on retrouve les entités suivantes: le Modèle, le *Layer*, la *Région* et le *Shell*. Contrairement aux objets topologiques de base, les éléments de ce groupe peuvent être affectés directement par une action de l'utilisateur. Cette partie des structures est dite **"partie visible" du modèle.**

La Figure 3.5 montre les relations entre les différentes structures de ces deux groupes d'objets.

3.4.2 Valeur du système de modélisation

La structure *Tlink*, tout comme la **Radial-Edge Structure** développée par K. Weiler [38], a été introduite pour permettre la description complète et non-ambigüe des adjacences topologiques entre les *Faces*. Dans le modèle de Weiler, les informations topologiques sont attachées aux edges importants du modèle. Dans notre modèle, nous profitons du maillage triangulaire des objets initiaux pour attacher ces informations aux triangles du bord des *Faces*. De cette façon, les informations d'adjacence sont plus nombreuses, donc plus précises. De plus, les critères définissant une "bonne" représentation volumique exposés par Gardan [11] (voir Introduction, Paragraphe 0.1.1) sont respectés. En effet:

1. **Puissance:**
   Le domaine de représentation de notre modèle est défini par les critères suivants:

- Surfaces *non-manifold*,
- *Faces manifold*,
- Non-intersection des *Faces*, edges et vertices,
- Finitude des *Faces*, edges et vertices.

2. **Suffisance:**
   Dans une topologie d'adjacence consistant de trois éléments de base (*Face*, edge et vertex), il peut exister neuf relations d'adjacence possible. Si une représentation topologique contient suffisamment d'informations pour recréer ces neuf relations d'adjacence sans erreur, elle peut être considérée comme une représentation topologique suffisante. Dans notre système de modélisation, seule la relation Face-Face est explicitement décrite á l'aide des *Tlinks*, *Triangle-uses* et *Face-uses*. Les huit autres relations d'adjacentes peuvent être implicitement déduites du modèle et des structures GOCAD.

3. **Validité:**
   La validité topologique du modèle est assurée à l'aide des hypothèses de départ sur les données, c'est -à-dire la non-intersection des divers éléments et leur finitude. Cependant, la validité géométrique du modèle ne peut être vérifiée automatiquement. En effet, seul l'utilisateur peut s'apercevoir, au moment de son interprétation, que les données sont erronnées (par exemple un "trou" entre deux *Faces* qui devraient être jointes).

4. **Unicité:**
   Les modèles Breps valides sont non-ambigus, ils ne sont pas uniques.

Il est également à noter que la représentation proposée dans ce rapport, bien que dérivant "conceptuellement" de la **Radial-Edge Structure**, comporte des différences majeures, au niveau de sa réalisation, avec la méthode de K.Weiler. Ces divergences, qui ont permis de construire une base de données considérablement "allégée", sont en grande majorité dues au fait que les objets utilisés dans cette recherche sont des surfaces triangulées.

3.4.3 Conclusion

A ce stade, nous connaissons:

- la définition d'une *Région*.
- la définition d'une *Face*.
- la structure décrivant les adjacences entre deux *Face-uses*.

En d'autres termes, nous disposons de tous les éléments composant notre modèle volumique *non-manifold*. L'étape suivante consiste donc à créer une représentation du modèle en définissant et gérant de façon automatique les *Régions*.

Chapitre 4

Détection et gestion des Régions

4.1 Introduction

Notre objectif est de déterminer, à partir des structures topologiques *Tlink*, *Face*, *Face-use* et *Triangle-use*, les *Régions* définissant le modèle volumique. Sachant qu'une *Région* est définie par ses frontières, cette opération revient donc à définir une *Région* en déterminant toutes ses frontières (*Shells*).

4.2 Détection des Régions

4.2.1 Introduction

La structure *Tlink* présentée dans le chapitre 3, paragraphe 3.3 a été introduite pour décrire les adjacences entre les *Faces* d'un modèle donné. Dans notre système de modélisation volumique de surfaces *non-manifold*, la structure *Tlink* est essentiellement utilisée pour examiner les conséquences de l'ajout d'une *Face* sur des structures existantes du modèle. C'est ce qu'on nomme le processus de détection automatique des *Régions*.

Supposons que le modèle comporte déjà un nombre n de *Régions* (éventuellement n=0), et supposons que l'on ajoute une *Face* F. Comme le suggère la Figure 4.1, trois cas peuvent se présenter:

1. F est à l'intérieur d'une *Région* et n'a pas de relation d'adjacence avec des *Faces* existantes du modèle (cas (c) sur figure 4.1).

2. F est à l'intérieur d'une *Région* et s'appuie en partie sur la frontière de celle-ci (cas (b) sur figure 4.1).

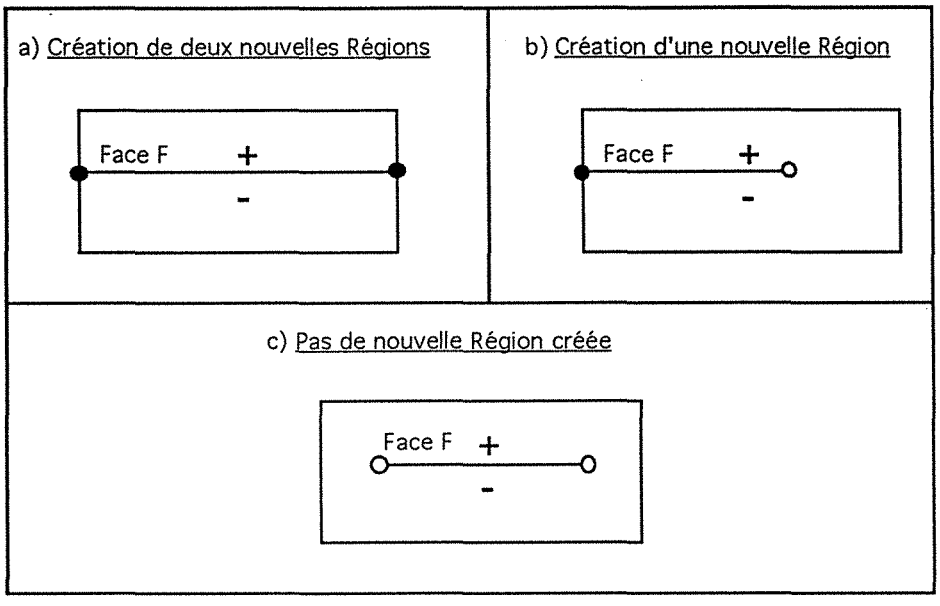


Figure 4.1 La détection de *Régions* (Représentation 2D).

3. F est à l'intérieur d'une *Région* et s'appuie entièrement sur la frontière de celle-ci (cas (a) sur figure 4.1).

Le procédé de détection automatique des *Régions* consiste à parcourir récursivement les côtés des *Faces* (donc les *Face-uses*) adjacentes à la *Face* ajoutée, afin de déterminer la frontière d'une *Région*.

4.2.2 Résumé de l'algorithme de détection des Régions

Soit F_1 la *Face* ajoutée, considérée comme la *Face* initiale du processus, et soient $[F_1,'+']$, $[F_1,'-']$ ses deux *Face-uses*[1].

A partir de la *Face-use* $[F_1,'+']$[2] de la *Face* F_1, le procédé consiste à examiner et marquer de façon récursive toutes les *Face-uses* adjacentes à $[F_1,'+']$. L'algorithme s'arrête lorsque toutes les *Face-uses* adjacentes ont été parcourues et peut avoir trois conclusions:

1. **Partage d'une *Région* en deux nouvelles *Régions*:**
   Toutes les *Face-uses* adjacentes à $[F_1,'+']$ ont été examinées et la *Face-use* $[F_1,'-']$ n'a pas été marquée. Dans ce cas, l'algorithme a permis de détecter deux volumes fermés, l'un avec le côté positif de la *Face* initiale, l'autre avec son côté négatif. De plus, il fournit la définition du *Shell* externe de chacune des deux nouvelles *Régions* (cas (a) sur figure 4.1).

[1] $[F_1,'+']$ est la *Face-use mate* de $[F_1,'-']$, et réciproquement.

[2] On a choisi arbitrairement de débuter le processus de détection des *Régions* à partir du côté positif de la *Face* initiale.

2. **Création d'une nouvelle** *Région*:
   Les deux *Face-uses* de la *Face* initiale ont été marquées et certaines *Faces* adjacentes ne sont marquées que d'un côté. Dans ce cas, un volume fermé a été détecté. Les *Face-uses* parcourues constituent le *Shell* externe de cette nouvelle *Région* (cas (b) sur figure 4.1).

3. **Pas de nouvelle** *Région*:
   Toutes les *Faces* adjacentes à la *Face* initiale, y compris la *Face* initiale elle-même, sont marquées des deux côtés. Dans ce cas, on est sûr qu'il n'existe aucune *Région* ayant la *Face-use* initiale comme frontière externe. L'algorithme fournit la définition d'un nouveau *Shell* interne qu'il faudra par la suite associer à une *Région* (cas (c) sur figure 4.1).

4.2.3 Notations

Les notations utilisées dans les algorithmes présentés dans ce chapitre sont les suivantes:

1. *Face*:
   F est la *Face* courante du processus.

2. *Face-use*:
   $[F,'+']$ et $[F,'-']$ sont les deux *Face-uses* de la *Face* F. Les *Face-uses* adjacentes seront présentées sous la forme: **[adjFace , adjSide]**.

3. *Triangle-use*:
   [Triangle , Side] représente un *Triangle-use* de la *Face-use* en cours d'examen.

4. Edge de *Triangle-use*:
   E est l'un des trois edges du *Triangle-use* courant.

4.2.4 Algorithme de détection des Régions

void Create_REGIONS (M , F)

- **Input:**
  - *Face* initiale, F.
  - Modèle M.
- **Output:** Modèle M.

{

/* *Initialisation* */

• Créer un ensemble vide $FaceUseSet_1$;

• Marquer toutes les *Face-uses* du modèle à 0.

/*

Définition du Shell à partir du côté positif de la Face initiale:
Recherche des Face-uses adjacentes à la Face-use associée au côté positif de la Face de départ.

*/

• **Create_SHELL** (M, $[F$, $'+']$, $FaceUseSet_1$) ;

si ($[F , '-']$ est marquée à 0)
alors

{

/* *Création de la première région* */

• **Create_New_REGION** (M, $FaceUseSet_1$) ;

• Créer un nouvel ensemble vide $FaceUseSet_2$;

/* *Définition du Shell de la seconde région* */

• **Create_SHELL** (M, $[F$, $'-']$, $FaceUseSet_2$) ;

/* *Création de la seconde région* */

• **Create_New_REGION** (M, $FaceUseSet_2$) ;

}

sinon

{

Modify_REGION (M, $FaceUseSet_1$) ;

}

}

Les deux fonctions **Create_New_REGION()** et **Modify_REGION()** utilisent les ensembles de *Face-uses* adjacentes constituant les *Shells* de *Régions* nouvelles ou existantes (voir Paragraphe 4.3).

void Create_Shell (*M*, *[F , S]*, *FaceUseSet*)

- **Input:**
  - *Face-use* initiale *[F , S]*.
  - Modèle *M*.
- **Output:** *FaceUseSet* = ensemble de *Face-uses* adjacentes à la *Face-use* initiale, et constituant le *Shell* de la *Région* associée à cette *Face-use*.

{

/ Initialisations */*

- Créer une pile vide *Stack* ;
- **Empiler** (*Stack* , *[F , S]*) ;

tant que(*Stack* $\neq \emptyset$)

{

- **Dépiler** (*Stack* , *[F , S]*) ;
- **Marquer** (*[F , S]* , 1) ;

/ Recherche des Face-uses adjacentes à la Face-use courante [F , S] */*

- Face_Use_**Find_Adjacencies** (*M*, *[F , S]*, *Stack*) ;
- **Ajouter** (*FaceUseSet*, *[F , S]*) ;

}

}

Les *Face-uses* stockées dans l'ensemble **FaceUseSet** définissent le nouveau *Shell* créé à partir de la *Face-use* associée au côté positif de la *Face* **initiale.**

4.2.5 Recherche des Face-uses adjacentes d'une Face-use

Sachant que les structures décrivant les adjacences entre *Faces* sont créées au niveau des triangles (voir *Tlink*), la recherche des *Face-uses* adjacentes d'une *Face-use* donnée nécessite l'examen des *Triangle-uses* de chaque *Face-use* rencontrée. Pour une *Face-use* donnée, un test est effectué sur chaque edge de chacun de ses triangles pour savoir si un *Tlink* existe.

void FACE_USE_**Find_Adjacencies** (*M*, *[F , S]*, *Stack*)

- **Input:**
  - *Face-use* courante *[F , S]*.
  - Modèle *M*.
  - *Stack* = pile contenant les *Face-uses* adjacentes à la *Face-use* initiale.
- **Output:** *Stack* = pile contenant les *Face-uses* adjacentes à la *Face-use* initiale.

```
{
  /* Parcours des Triangle-uses de la Face-use [ F , S ] */
  pour ( chaque Triangle-use [ T , S ] de la Face-use [ F , S ] )
    {
      /* Parcours des edges du Triangle-use */
      pour ( chaque edge E du triangle T )
        {
          /* Test de l'existence d'un Tlink associé à l'edge E du triangle T à l'aide
             de la fonction Tlink() définie dans le chapitre 3, paragraphe 3.3. */
          si ( ∃λ = Tlink ( T , E ) )
            {
              /* Accès au Triangle-use adjacent:
                 La fonction Tlink2Tuse() retourne le Triangle-use du Tlink λ
                 associé au côté S du triangle T. */
              [ T2 , S2 ] = Tlink2Tuse ( λ , S ) ;

              /* Recherche de la Face-use adjacente:
                 La fonction Triangle2Face() retourne, pour un triangle donné, la Face
                 le contenant. */
              [ adjFace , adjSide ] = [ Triangle2Face ( T2 ) , S2 ] ;

              si ( [ adjFace , adjSide ] est marquée à 0 )
                {
                  • Empiler ( Stack , [adjFace , adjSide] ) ;
                  • Marquer ( [ adjFace , adjSide ] , 1 ) ;
                }
            }
        }
    }
}
```

4.3 Gestion des Régions

4.3.1 Introduction

L'algorithme de détection des *Régions* a produit plusieurs ensembles de *Face-uses* définissant de nouveaux *Shells*. A partir de ces ensembles, la seconde phase du traitement automatique des *Régions* consiste à mettre à jour la base de données des *Régions*. On distingue trois cas, en fonction de la conclusion de l'algorithme de détection des *Régions*:

1. **Création d'une nouvelle *Région*:**
   Il existe un ensemble de *Face-uses* adjacentes définissant le *Shell* externe de cette *Région*.

2. **Création de deux nouvelles *Régions*:**
   Il existe deux ensembles de *Face-uses* adjacentes, chacun d'eux définissant le *Shell* externe d'une *Région*.

3. **Modification de la *Région* initiale:**
   Il existe un ensemble de *Face-uses* adjacentes définissant un *Shell* interne.

4.3.2 Définition de la Région initiale

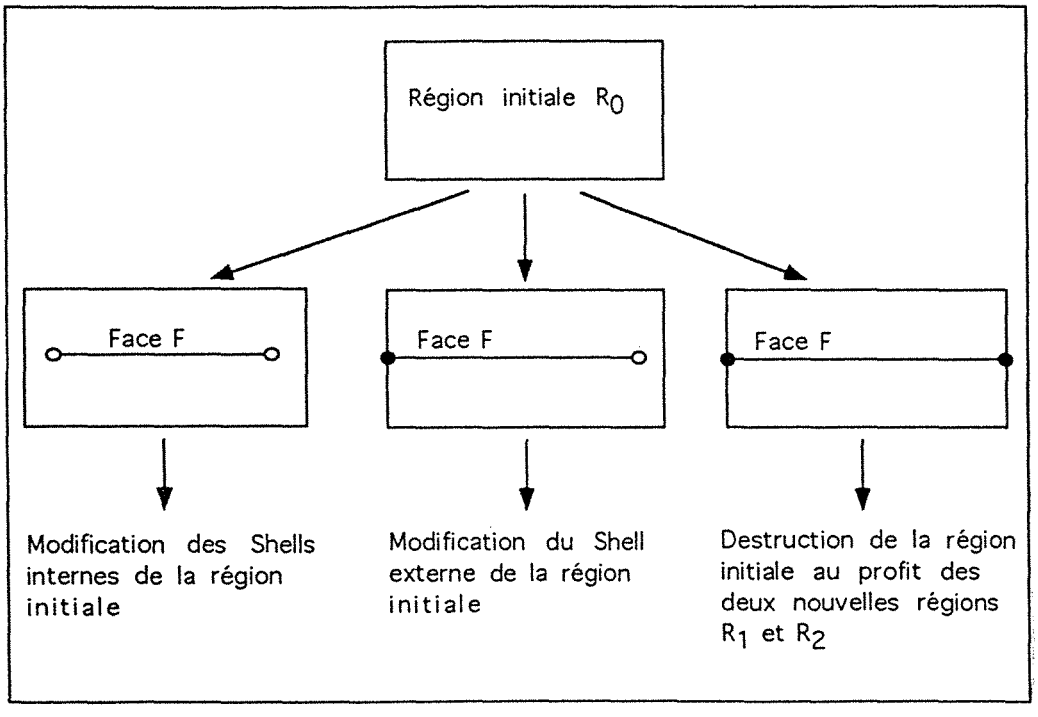


Figure 4.2 Modifications de la *Région* initiale (Représentation 2D).

Le procédé de gestion automatique des *Régions* requiert une contrainte importante que nous n'avons pas encore présentée. En effet, il est nécessaire de connaitre la *Région* contenant la *Face* de départ avant de lancer le processus. Cette *Région*, que nous appellerons la **Région initiale**, est fondamentale car c'est elle qui subira les modifications dues au résultat de l'algorithme (voir Figure 4.2). Ces modifications peuvent être:

- la destruction de la *Région*:
  Dans le cas d'une création de deux *Régions*, la *Région* initiale est intégralement remplacée par ces nouvelles *Régions*.
- la modification du *Shell* externe de la *Région*:
  La création d'une nouvelle *Région* revient à modifier le *Shell* externe de la *Région* initiale en lui ajoutant certaines *Face-uses*, dont la *Face-use* initiale. En conséquence, plutôt que de détruire cette *Région* (comme pour la création de deux *Régions*) pour en construire une nouvelle quasi-identique, il est plus simple et plus efficace de la modifier.
- la modification des *Shells* internes de la *Région*.
  L'ensemble des *Shells* internes de la *Région* initiale doit être modifié en tenant compte des deux *Face-uses* de la *Face* initiale.

Ceci signifie que toute *Face* doit être contenue par une seule et unique *Région*. En fait, pour cela, il suffit que les surfaces initiales ne s'intersectent pas. Cette contrainte est vérifiée à chaque nouvelle création des *Faces*.

La définition de la *Région* initiale s'effectue très simplement en utilisant une fonction de localisation d'un point dans le modèle. Cette fonction (voir fonction **POINT_Get_REGION()** dans Annexe C, Paragraphe C.3) consiste à déterminer de façon automatique la *Région* contenant un point (x,y,z) donné. On l'utilise ici pour déterminer la *Région* contenant la *Face* initiale de notre traitement, ceci en localisant un point quelconque interne à cette *Face*.

Un point interne à une *Face* est un point situé ailleurs que sur le bord de celle-ci. En effet, les noeuds de la frontière sont susceptibles d'être partagés par plusieurs *Faces*. Pour un point du bord, il est impossible de décider entre les *Régions* limitées par ces *Faces*, et plus exactement par leurs *Face-uses*. Quelque soit le point interne choisi, la *Région* trouvée est toujours la même en raison de la contrainte que nous nous sommes fixés, à savoir qu'une *Face* donnée est incluse dans une seule et unique *Région*.

4.3.3 Création de deux nouvelles Régions

Dans le cas d'une création de *Régions*, l'algorithme précédent a produit deux *Shells*. Le premier a été obtenu à partir de la *Face-use* associée au côté positif de la *Face* initiale. Le second a été défini à partir de la Face-use associée au côté négatif de la *Face* initiale (donc *Face-use mate* de la *Face-use* utilisée pour l'autre *Shell*). La création des deux *Régions* s'effectue en quatre étapes:

1. **Création effective de la première** *Région*
2. **Création effective de la seconde** *Région*

3. **Réaffectation des *Shells* internes de la *Région* initiale**

4. **Destruction de la *Région* initiale**

Création effective d'une Région

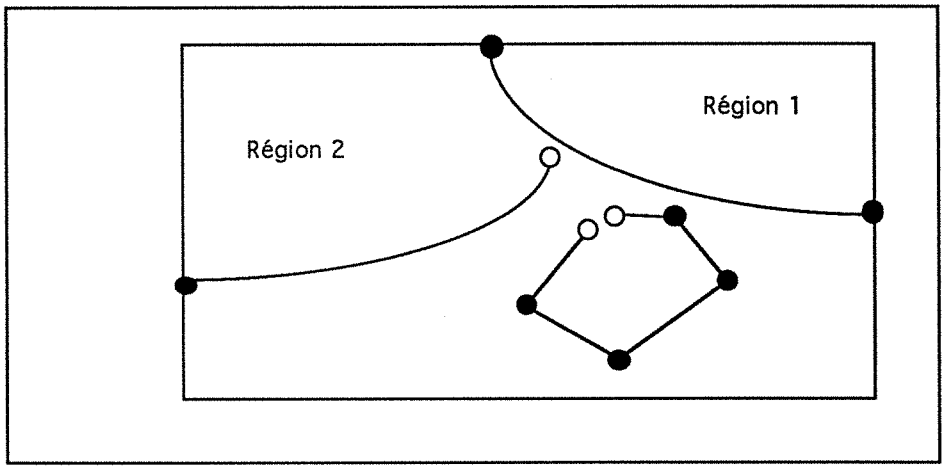


Figure 4.3 Création de *Régions* - Situation initiale (Représentation 2D).

L'algorithme de détection des *Régions* a produit deux ensembles de *Face-uses* adjacentes, donc deux *Shells*. D'après la définition du *Shell*, on sait que celui-ci peut être **interne** ou **externe** selon qu'il définit l'enveloppe globale d'une *Région*, ou la frontière d'une sous-*Région* de celle-ci. Avant d'attribuer nos deux *Shells* à leurs *Régions* respectives, on a donc besoin de les identifier en tant qu'interne ou externe. Pour cela, on peut distinguer deux cas[3] de création de *Région*:

1. Création par division d'une *Région* existante.

2. Création par fermeture d'une *Région*.

Création par division d'une Région existante

Dans ce cas (voir Figure 4.4), les deux *Shells* créés par l'algorithme de détection s'appuient sur le *Shell* externe de la *Région* initiale, et constituent eux-mêmes le *Shell* externe de leur *Région* respective.

Création par fermeture d'une Région

Dans ce cas (voir Figure 4.5), l'un des deux *Shells* construits est interne, l'autre est externe. Le processus qui a construit ces *Shells* ne sait pas qui est quoi. Il faut donc qualifier chacun

[3]Chaque cas est illustré par un schéma faisant référence à la situation initiale présentée sur la figure 4.3.

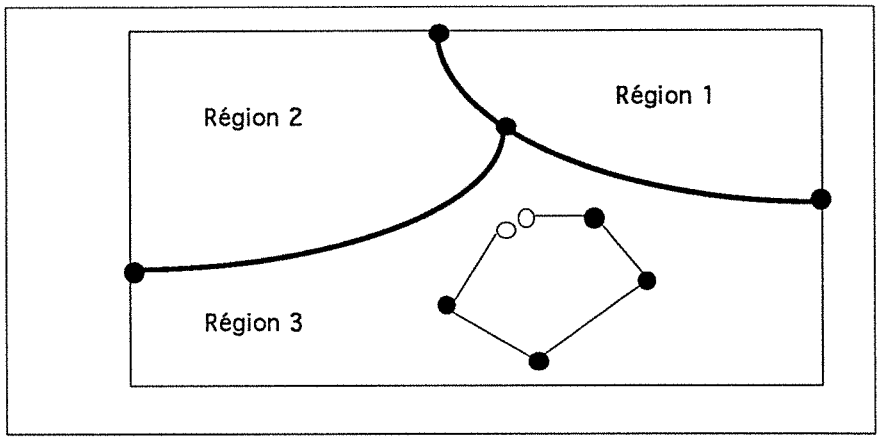


Figure 4.4 Création de *Régions* par division (Représentation 2D).

des deux *Shells* construits comme interne ou externe. Le processus permettant de prendre cette décision est le suivant:

1. Calcul du volume de chacun des deux *Shells* construits:
   Un *Shell* est un ensemble de *Face-uses* adjacentes. Chacune de ces *Face-uses* est originaire d'une *Face* à laquelle est associée une orientation. Cette orientation est obtenue à l'aide des vecteurs normaux attachés à chaque triangle composant la *Face*.

   Le volume d'un *Shell* est obtenu en additionnant le volume de chacune de ses *Face-uses*. Par définition, le volume d'une *Face* (et donc d'une *Face-use*) est nul. C'est pourquoi on considère que chaque *Face* est dupliquée (deux *Face-uses*) de façon à créer un volume. Il en est de même pour les triangles. Pour calculer le volume d'une *Face-use*, on calcule le volume de chacun de ses *Triangle-uses*. Ceci s'obtient en calculant dans un premier temps l'aire du triangle, puis en multipliant le résultat par la moyenne des coordonnées z de chaque sommet. Enfin, selon le côté du triangle utilisé par la *Face-use*, la valeur obtenue est multipliée par 1 (côté positif) ou par -1 (côté négatif). En définitive, le volume du *Shell* est obtenu en multipliant la somme des volumes de chaque *Face-use* le composant par -1.

2. Pour chaque *Shell*, tester le signe du volume calculé:

   - **si** ($volume > 0$) le *Shell* est externe.
   - **si** ($volume \leq 0$) le *Shell* est interne.

Algorithme de création d'une Région

Pour éviter de distinguer deux cas dans l'algorithme de création d'une *Région*, on a choisi de traiter les deux types de création de la même façon, c'est-à-dire selon la méthode de création par fermeture d'une *Région*.

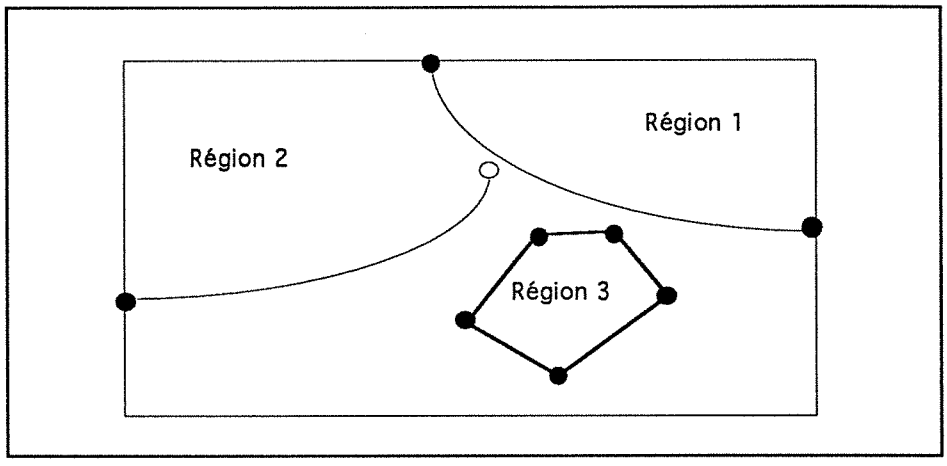


Figure 4.5 Création de *Régions* par fermeture (Représentation 2D).

D'après la définition de la structure *Région* (voir Chapitre 2, Paragraphe 2.2), la création d'une nouvelle entité *Région* dans le modèle comporte:

- **le choix du nom de cette Région:**
  Cette opération se réalise automatiquement par concaténation de la chaine de caractères "Region" et d'un compteur incrémenté à chaque nouvelle création de *Région*.

- **la création des *Shell*s de cette *Région*:**
  Comme nous l'avons précisé plus haut (voir Création par fermeture d'une *Région*), cette opération consiste à identifier chaque *Shell* obtenu grâce à l'algorithme de détection de *Région* comme **interne** ou **externe**.

- **la création d'un *Layer* contenant cette *Région*:**
  Un *Layer* est automatiquement créé à chaque nouvelle création de *Région* (voir Chapitre 2, Paragraphe 2.4). La définition du nom du *Layer* associé à la nouvelle *Région* s'effectue de la même façon que pour la *Région*, à l'aide d'un second compteur.

Ces opérations sont réalisées pour chacune des deux nouvelles *Régions*.

void Create_New_REGION (*M*, *FaceUseSet*)

- **Input:**
  - *FaceUseSet* = ensemble de *Face-uses* adjacentes définissant un *Shell* de la nouvelle *Région.*
  - Modèle *M*.
- **Output:** Modèle *M*.

```
{
  /* Recherche de la Région contenant la Face initiale. */
  • Soit point un point de la Face initiale ;
  • init_Région = POINT_Get_REGION( M , point ) ;
  /* Initialisations. */
  • Créer un Shell vide new_Shell ;
  • new_Shell → face_Set = FaceUseSet ;
  • Créer une Région vide new_Région ;
  • Créer un Layer vide new_Layer ;
  • new_Région → layer = new_Layer ;
  • Ajouter ( new_Layer→ région_Set , new_Région ) ;
  /* Calcul du volume du Shell défini par FaceUseSet */
  • volume = Compute_Volume_Of_FACE_Set ( FaceUseSet ) ;
  si ( volume ≤ 0 )
  alors    /* Cas d'un Shell interne. */
    {
      • new_Shell→ type = 1 ;
      • new_Shell→ région = new_Région ;
      • Ajouter ( new_Région→ shell_Set , new_Shell ) ;
      /* La fonction Find_External_SHELL() permet de récupérer le Shell externe de la
      Région initiale pour construire celui de la nouvelle Région. */
      • external_Shell = Find_External_SHELL( init_Région ) ;
      • external_Shell→ région = new_Région ;
      • Ajouter ( new_Région→ shell_Set , external_Shell ) ;
    }
  sinon    /* Cas d'un Shell externe. */
    {
      • new_Shell→ type = 0 ;
      • new_Shell→ région = new_Région ;
      • Ajouter ( new_Région→ shell_Set , new_Shell ) ;
    }
}
```

Réaffectation des Shells internes de la Région initiale

La *Région* initiale a été intégralement remplacée par deux nouvelles *Régions*. Elle est devenue redondante et n'a donc plus de raison d'exister. Avant de procéder à sa destruction, il nous faut répartir les éventuels *Shells* internes qui la définissent entre les deux nouvelles *Régions*.

void Reaffect_Internal_SHELLS (*M*, *init_Région*)

- **Input:**
  - *init_Région* = référence à la *Région* contenant la *Face* initiale.
  - Modèle *M*.
- **Output:** Modèle *M*.

{

- Créer un ensemble vide *shell_Set* ;
- Créer une liste vide *shell_List* ;

/* *La fonction* **Find_Internal_SHELLS**() *permet de récupérer les Shells internes de la Région initiale.* */

- *internal_Shell_Set* = **Find_Internal_SHELLS** (*init_Région*) ;

/* *Recherche du point (x,y,z) de x maximum, pour chaque Shell interne.* */

pour (chaque *Shell* $Sh_i \in$ *internal_Shell_Set*)

{

- $max\_x = -1.e30$;
- **pour** (chaque *Face-use* du *Shell* Sh_i)

{

- *vrtx* = vertex de la *Face-use* ayant le x maximum ;
- **si** ($vrtx \rightarrow x > max\_x$)

{

- $max\_x = vrtx \rightarrow x$;
- $vrtx\_max = vrtx$;

}

}

- **Ajouter** (*shell_Set* , (Sh_i , *vrtx_max*)) ;

}

/* *Tri des Shells internes par ordre décroissant de leur x-maximum.* */

shell_List = **Sort** (*shell_Set* , ($vrtx\_max_i \rightarrow x > vrtx\_max_{i+1} \rightarrow x$)) ;

/* *Associer chaque Shell interne à une Région.* */

pour (chaque couple $(Sh_i, vrtx\_max_i) \in$ *shell_List*)

{

- *région* = **POINT_Get_REGION** ($vrtx\_max_i$) ;
- $Sh_i \rightarrow$ région = *région* ;
- **Ajouter** (*région*→ shell_Set , Sh_i) ;

}

}

Destruction de la Région initiale

A présent que toutes les informations de la *Région* initiale ont été réparties entre les deux nouvelles *Régions*, la *Région* initiale n'a plus de raison d'exister. La destruction de cette *Région* s'effectue en plusieurs étapes:

1. Retrait de cette *Région* de l'ensemble des *Régions* de son *Layer*.
2. Destruction de son *Layer*, s'il n'existe pas d'autres *Régions* le composant.
3. Retrait de cette *Région* de l'ensemble des *Régions* du modèle.
4. Destruction de la Région:
   Les *Shells* ont déjà été traités, donc cette destruction se résume à la destruction du pointeur de *Région*.

4.3.4 Modification de la Région initiale

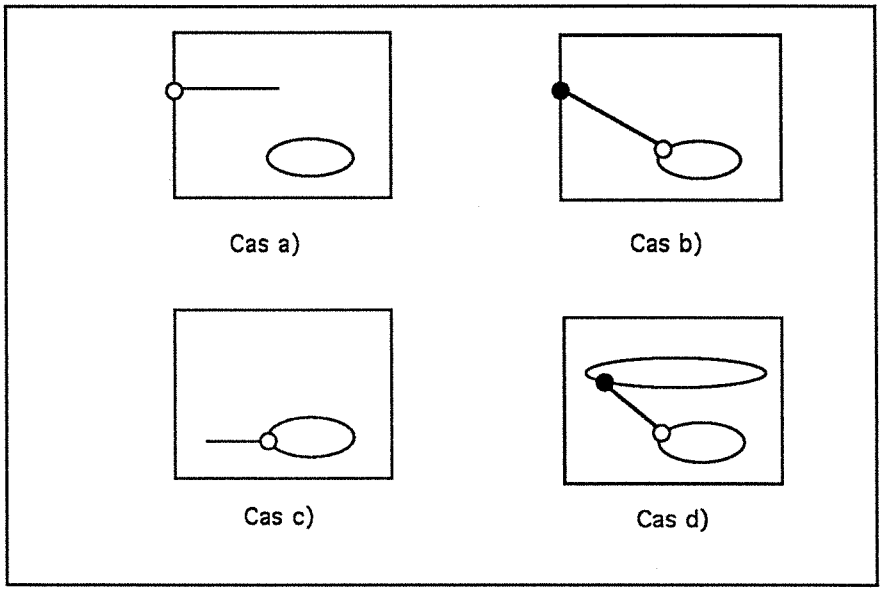


Figure 4.6 Modification de shells (Représentation 2D).

Cette situation apparait lorsque l'algorithme de détection des *Régions* a produit un seul ensemble de *Face-uses* adjacentes définissant un nouveau *Shell*. Ce dernier contient les deux *Face-uses* associées à la *Face* initiale du processus. La modification de la *Région* initiale résulte en la modification de la structure des *Shells* qui la définissent:

1. Ajout des deux *Face-uses* de la *Face* initiale à la définition du *Shell* externe (a sur Figure 4.6).

2. Regroupement du *Shell* externe et d'un *Shell* interne (b sur sur Figure 4.6).

3. Ajout des deux *Face-uses* de la *Face* initiale à la définition d'un *Shell* interne (c sur Figure 4.6).

4. Regroupement de deux *Shells* internes (d sur Figure 4.6).

Les cas 1 et 2 apparaissent lorsque l'algorithme de détection des *Régions* a conclu à la création d'une nouvelle *Région*.

Tous ces cas peuvent être résolus par un même algorithme grâce à la connaissance des *Face-uses* examinées lors du processus de détection des *Régions*.

Résumé

Soit *FaceUseSet* l'ensemble des *Face-uses* adjacentes obtenu grâce à l'algorithme de détection des *Régions*, et définissant un nouveau *Shell*. Notre problème consiste à trouver les *Shells* de la *Région* initiale devant être modifiés. Ces *Shells* sont en fait ceux dont la définition fait appel à des *Face-uses* appartenant à l'ensemble *FaceUseSet*.

Une première étape consiste donc à retrouver, pour chaque *Face-use* de l'ensemble *FaceUseSet*, le *Shell* de la *Région* initiale qui l'utilise. La seconde étape consiste ensuite à créer un nouveau *Shell* à partir de l'ensemble de *Face-uses*. Cette opération revient en fait à fusionner les *Shells* qui ont servi de support à la définition de ce nouveau *Shell*. De ce fait, ces *Shells* originaux n'ont plus de raison d'être, et sont donc supprimés au cours d'une troisième étape.

void Modify_Region (*M*, *FaceUseSet*)

- **Input:**
  - *FaceUseSet* = ensemble de *Face-uses* adjacentes définissant un *Shell* de la nouvelle *Région*.
  - Modèle *M*.
- **Output:** Modèle *M*.

{

- Créer une ensemble vide *shell_Set* ;
- *init_Région* = *Région* contenant la *Face* initiale

(voir calcul dans fonction **Create_New_Region** ()) ;

/* *Recherche des Shells existants servant de support au nouveau Shell.*
La fonction Face_Use_Is_In_Shell() *retourne VRAI si une Face-use donnée est utilisée dans la définition d'un Shell donné. Elle retourne FAUX dans le cas contraire.* */

pour (chaque *Face-use* $[F, S] \in FaceUseSet$)
{
pour (chaque *Shell* $Sh_j \in$ *init_Région*— shell_Set)
{
si (Face_Use_Is_In_Shell ($[F, S]$, Sh_j))
{
- **Ajouter** (*shell_Set* , Sh_j)

/* *Une Face-use n'appartenant, par définition, qu'à un seul et unique Shell, une fois que celui-ci est trouvé, il est totalement superflu d'examiner les autres Shells de la Région initiale. L'algorithme passe donc à la Face-use suivante du nouveau Shell.* */

- **break** ;

}
}
}

/* *Création d'une entité Shell.* */

- Créer un *Shell* vide *new_Shell* ;
- *new_Shell*→ face_Set = *FaceUseSet* ;
- *new_Shell*→ région = *init_Région* ;

/* *Destruction des anciens Shells.* */

pour (chaque *Shell* $Sh_j \in$ *shell_Set*)
{
- **Supprimer** (*init_Région*→ shell_Set , Sh_j) ;
- **FREE** (Sh_j) ;

}

}

4.4 Cas des Faces fermées

4.4.1 Introduction

Imaginons une *Face* initiale F fermée, c'est-à-dire sans bord. Le système reconnait une telle *Face* au moment de la création des *Tlink*s qui lui sont associés. En effet, une *Face* fermée n'ayant pas de bord, aucun *Tlink* n'est défini. C'est le traitement suivant la détection de ce cas particulier qui est présenté dans ce paragraphe.

L'intérieur de toute *Face* fermée constitue à lui seul une *Région*. De plus, l'extérieur de la *Face* définit une nouvelle frontière pour la *Région* contenant la *Face*. En conséquence, il nous faut attribuer les deux *Face-use*s de cette *Face* fermée à deux *Régions*. Cette opération est réalisée au moment de la création de la *Face* et se déroule de la façon suivante:

1. Rechercher la *Région* contenant la *Face*, c'est-à-dire la *Région* initiale.
2. Déterminer, parmi les deux *Face-use*s de la *Face*, celle constituant un *Shell* interne (l'extérieur de la *Face*), et celle définissant un *Shell* externe (l'intérieur de la *Face*).
3. Créer une nouvelle *Région* définie par le *Shell* externe.
4. Associer le *Shell* interne à la *Région* initiale.

4.4.2 Recherche de la Région initiale

La définition de la *Région* contenant la *Face* traitée s'effectue de la même façon que pour toute autre *Face*, à l'aide de la fonction de localisation d'un point (voir Paragraphe 4.3.2).

4.4.3 Définition des Shells interne et externe

Il est indispensable, pour la cohérence du modèle, de définir avec précision quelle *Face-use* de la *Face* délimite l'intérieur de cette *Face*. Pour cela, les opérations suivantes sont exécutées:

1. Calcul du volume de la *Face-use* associée au côté positif de la *Face*:
   Par définition, le volume d'une *Face* (et donc d'une *Face-use*) est nul. C'est pourquoi on considère que chaque *Face* est dupliquée (deux *Face-use*s) de façon à créer un volume. Il en est de même pour les triangles.

   Une *Face* est composée de triangles adjacents. Pour calculer le volume d'une *Face-use* fermée, il suffit d'additionner le volume de chaque *Triangle-use* la composant. Ceci s'obtient en calculant dans un premier temps l'aire du triangle, puis en multipliant le résultat par la moyenne des coordonnées z de chaque sommet. En définitive, le volume de la *Face-use* est obtenu en multipliant la somme des volumes de chaque *Triangle-use* la composant par -1.

2. Tester le signe du volume:

- si (*volume* > 0)
  les vecteurs normaux associés aux triangles définissant la *Face* pointent vers l'intérieur de la *Face*. Donc la *Face-use* associée au côté positif de la *Face* constitue le *Shell* externe de la *Région* interne à la *Face*. Et réciproquement, l'autre *Face-use* constitue un *Shell* interne de la *Région* initiale.
- si (*volume* ≤ 0)
  les vecteurs normaux des triangles de la *Face* pointent vers l'extérieur de la *Face*. Donc la *Face-use* associée au côté positif de la *Face* est un *Shell* interne pour la *Région* initiale. Réciproquement, l'autre *Face-use* constitue le *Shell* externe de la *Région* définissant l'intérieur de la *Face*.

4.4.4 Création de la Région définissant l'intérieur de la Face

La création de la *Région* définissant l'intérieur de la *Face* fermée s'effectue exactement de la même façon que pour toute autre *Région* (voir Paragraphe 4.3.2), c'est-à-dire:

- Choix du nom de la nouvelle *Région*, de façon automatique, à l'aide du compteur de *Régions*.
- Association du *Shell* externe défini précédemment à cette nouvelle *Région*.
- Création d'un *Layer* contenant cette *Région*.

4.4.5 Association du Shell interne à la Région initiale

L'ajout de la *Face* fermée dans le modèle agit comme une division de la *Région* initiale en deux nouvelles *Régions*. La première, correspondant à l'intérieur de la *Face*, est créée par le processus défini dans le paragraphe précédent. La seconde *Région* correspond à la *Région* initiale modifiée par l'ajout d'une nouvelle frontière. En conséquence, le *Shell* interne défini dans le paragraphe 4.4.3 est affecté à la *Région* initiale.

4.5 Conclusion

La définition et la gestion des *Régions* d'un modèle donné est entièrement automatique et repose sur les informations des structures *Tlinks*. Ces dernières étant parfaitement précises et complètes, les éventuelles erreurs obtenues ne peuvent être dues qu'à un défaut des données initiales. Les opérations de manipulation du modèle (voir Annexe B) et les opérations graphiques (voir Annexe C) permettent de détecter ces erreurs, comme par exemple des "trous" entre deux surfaces que l'on croyait parfaitement collées. Ces opérations ne permettent cependant pas de corriger les problèmes éventuels. En effet, si l'erreur nécessite la modification des données initiales, c'est à l'utilisateur de l'effectuer.

Dans la représentation par frontières *non-manifold* obtenue à partir des données initiales, il n'existe aucune relation entre les divers éléments de l'objet à modéliser. De ce fait, la topologie

inter-*Face*s n'existe pas, donc ne peut être décrite. Les deux parties qui vont à présent vous être proposées décrivent en détail les deux méthodes mises au point pour créer des adjacences entre les *Face*s de l'objet modélisé.

PARTIE II

CONSTRUCTION INTERACTIVE DU MODELE

Introduction

Afin de créer une topologie inter-*Faces*, il faut créer des relations d'adjacence entre les données initiales, c'est-à-dire entre les surfaces composant l'objet modélisé. La première méthode mise au point pour créer ces liens est une méthode interactive reposant sur l'utilisation de lignes spéciales appelées **Glue-Lines**.

Le chapitre 5 donne une définition détaillée de la Glue-Line, en précisant sa raison d'être.

Le chapitre 6 est consacré à la présentation des divers processus interactifs mis au point pour la création de Glue-Lines.

Le chapitre 7 décrit en détail l'utilisation des Glue-Lines lors du processus de collage de surfaces.

Chapitre

5

La structure Glue-Line

5.1 Introduction

La **Glue-Line** ("ligne de colle") est une structure introduite pour permettre la modification interactive de la topologie d'un modèle donné. Cette modification consiste en la création de **liens physiques intelligents** entre les surfaces choisies, encore appelée *collage de deux surfaces*:

- **Lien Physique:**
  La création d'un lien entre deux surfaces s'effectue en *fusionnant une Glue-Line* définie sur l'une des deux surfaces à une autre *Glue-Line* définie sur l'autre surface. De cette façon les deux surfaces sont physiquement liées.

- **Lien Intelligent:**
  Chaque *Glue-Line* possède des informations sur la topologie locale (le long de la ligne) de la surface sur laquelle elle est définie. Plus qu'une simple ligne polygonale définie sur une surface, une *Glue-Line* est donc une structure topologique contenant des informations complémentaires sur les triangles de la surface qui lui sont adjacents. Le lien créé entre deux surfaces par l'intermédiaire de deux *Glue-Lines* est qualifié d'intelligent car il est non seulement réalisé par une fusion géométrique, mais également par la création de structures décrivant les adjacences des triangles fusionnés, c'est-à-dire des structures *Tlink*.

D'après cette définition, il est logique de dire qu'une *Glue-Line* n'a pas d'existence (et pas de signification !!) en dehors du domaine de la surface. La surface sur laquelle une *Glue-Line* donnée est définie est appelée **Surface-Mère** de la *Glue-Line*. Plusieurs *Glue-Lines* peuvent être créées sur une même surface, donc une même surface peut être la *Surface-Mère* de plusieurs *Glue-Lines*.

5.2 Définition de la Glue-Line

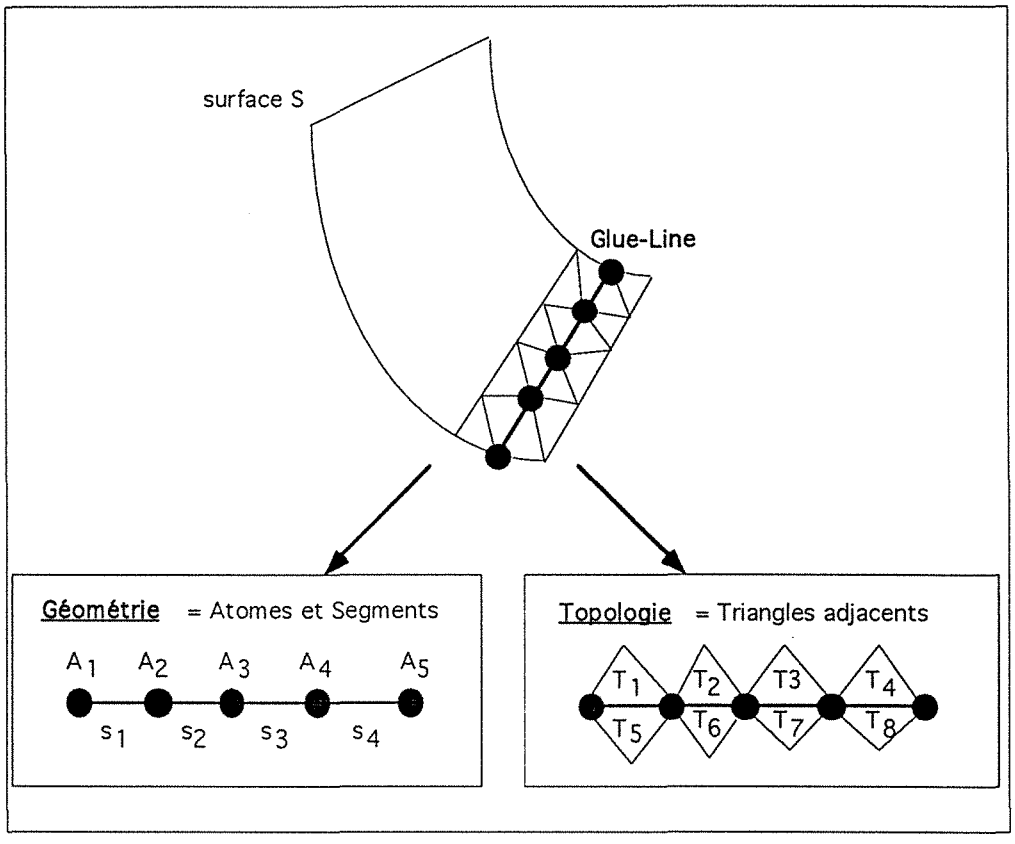


Figure 5.1 Définition de la *Glue-Line*.

La définition d'une *Glue-Line* est entièrement liée à celle de sa *Surface-Mère*. Cette définition comporte deux types d'information (voir Figure 5.1):

- **Géométrie:**
  La *Glue-Line* partage cette information avec sa *Surface-Mère*. La géométrie de la *Glue-Line* est composée de:

  - atomes correspondant aux noeuds de la surface,
  - segments correspondant aux arêtes de la surface, c'est-à-dire aux edges des triangles composant la surface.

  La *Glue-Line* n'a donc pas de structure propre pour décrire sa géométrie. Ceci renforce le fait qu'une *Glue-Line* n'a pas d'existence en dehors du domaine de la surface.

- **Topologie:**
  Cette information est liée à la *Surface-Mère* de la *Glue-Line*. L'information topologique d'une *Glue-Line* est attachée à chacun des segments qui la composent. Elle correspond aux triangles de la *Surface-Mère* adjacents à la *Glue-Line*. En d'autres termes, à chaque segment de la *Glue-Line* sont associés les deux triangles de la surface ayant deux sommets correspondant aux extrémités dudit segment.

5.3 Rôle de la Glue-Line dans la construction des Faces

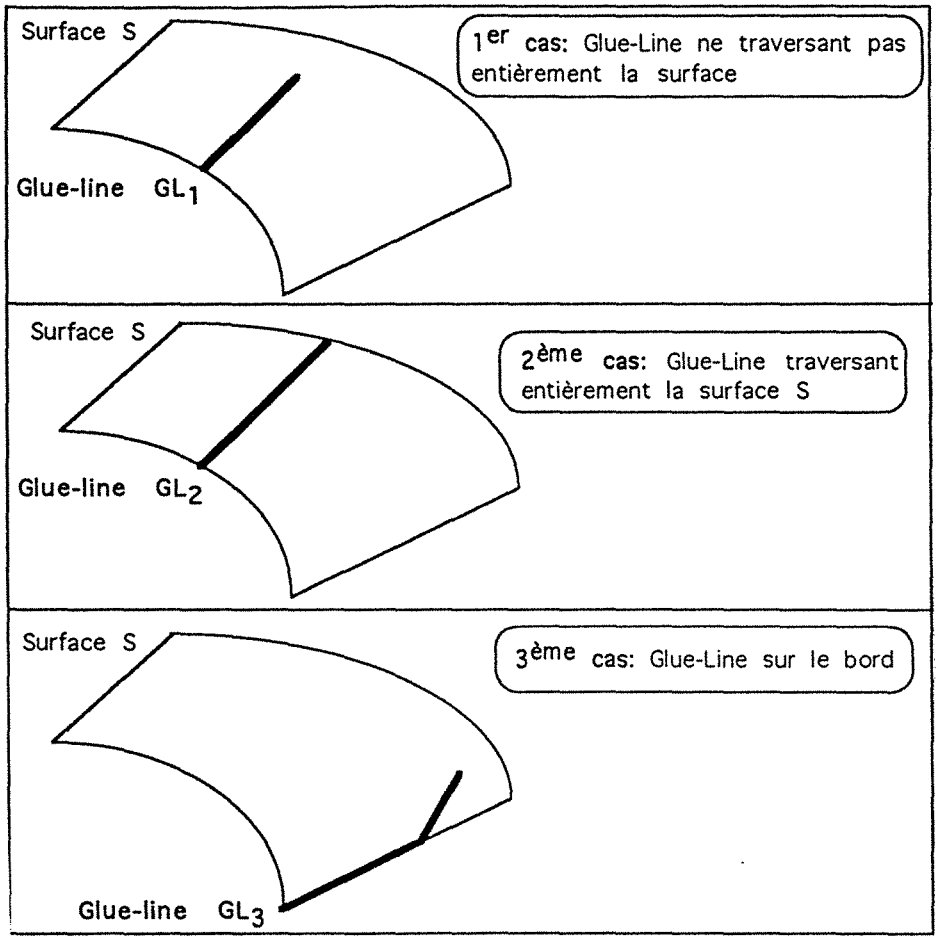


Figure 5.2 Rôle de la *Glue-Line* dans le découpage d'une surface en *Faces*.

Une *Glue-Line* peut exister partout sur la surface et joue un rôle essentiel dans le découpage d'une surface en *Faces*. En effet, selon sa position géométrique, la *Glue-Line* peut modifier la structure des *Faces* associées à sa *Surface-Mère*. L'intervention de la *Glue-Line* dans la définition

des *Faces* peut s'effectuer de trois façons différentes[1]:

1. **La *Glue-Line* GL_1 ne traverse pas complètement la surface S:**
   Dans ce cas, F demeure l'unique *Face* associée à la surface S. Cependant, la *Glue-Line* GL_1 est prise en compte dans la définition du bord de cette *Face*. En d'autres termes, la géométrie de la *Glue-Line* GL_1 s'ajoute au bord existant de la *Face* F, c'est-à-dire le bord de la surface S.

2. **La *Glue-Line* GL_2 traverse complètement la surface S:**
   La *Glue-Line* découpe la *Face* F en deux nouvelles *Faces*. Chacune d'elles prend en compte la *Glue-Line* GL_2 pour la définition de son bord.

3. **La *Glue-Line* GL_3 est sur le bord de la surface S:**
   La *Glue-Line* GL_3 se confond entièrement -ou en partie- avec le bord de sa *Surface-Mère*. Or, comme nous l'avons précisé lors de la définition de la *Face* (voir Chapitre 3, Paragraphe 3.2), le bord de la surface constitue le bord de ses *Faces* associées. Dans ce cas, la portion de la *Glue-Line* reposant sur le bord de la surface n'apporte aucune information complémentaire sur le bord des *Faces*. En conclusion, dans notre cas, cette portion de la *Glue-Line* GL_3 n'est pas prise en compte dans la définition du bord de la *Face* F.

On peut déduire de cette étude de cas que le *collage de deux surfaces* par l'intermédiaire de deux *Glue-Lines* résulte en fait en l'*association des Faces* de chacune des deux surfaces dont le bord est défini en partie par l'une des deux *Glue-Lines*. En d'autres termes, les *Glue-Lines* permettent de créer des connexions entre les *Faces* du modèle. Pour éviter toute confusion, nous parlerons par la suite de:

- *fusion de deux Glue-Lines* pour exprimer l'action permettant d'obtenir le *collage de deux surfaces.*
- *collage de deux surfaces* pour exprimer au niveau de l'utilisateur la création d'un lien entre deux surfaces.
- *association de Faces* pour exprimer au niveau interne au modèle le résultat du *collage de deux surfaces.*

5.4 Utilisation de la Glue-Line

Comme nous l'avons dit dans l'introduction de ce paragraphe, la notion de *Glue-Line* a été introduite dans un but précis, à savoir le *collage de deux surfaces.* Il existe deux opérations de manipulation d'une *Glue-Line*:

1. Définition d'une *Glue-Line* sur une surface donnée.

[1]Pour chaque cas, illustré par la figure 5.2,on supposera qu'il existe au départ une première *Face* F associée à la surface S; cette dernière est tour à tour la *Surface-Mère* de trois *Glue-Lines*: GL_1, GL_2 et GL_3.

2. Fusion de cette *Glue-Line* avec une autre.

Dans le cadre du *collage d'une surface* S_1 à une surface S_2, ces opérations s'enchainent de la façon suivante:

1. Définir une *Glue-Line* sur la surface S_1.
2. Définir une *Glue-Line* sur la surface S_2.
3. *Coller les surfaces* S_1 *et* S_2 en fusionnant les deux *Glue-Line*s précédemment définies.

Les deux opérations composant ce processus de *collage de deux surfaces* (ie définition et fusion) seront définies respectivement dans les chapitres 6 et 7.

5.5 Statut d'une Glue-Line

Selon son niveau d'utilisation, c'est-à-dire en fonction des opérations qui lui ont été appliquées, une *Glue-Line* a différents statuts (voir Figure 5.3):

- **EXISTING:**
  Etat de la *Glue-Line* après sa définition et avant son utilisation dans une opération de *fusion.*
- **MASTER:**
  Etat de la *Glue-Line* lors de son utilisation pour une opération de *fusion.*
- **SLAVE:**
  Etat de la *Glue-Line* lors de son utilisation pour une opération de *fusion.*
- **USED:**
  Etat d'une *Glue-Line* SLAVE après une opération de *fusion.*

Dans l'expression "coller une surface sur une autre" intervient la notion de **destination du** *collage.* Les *Glue-Line*s "source" et "destination" sont choisies par l'utilisateur. Elles prennent alors respectivement les statuts SLAVE et MASTER.

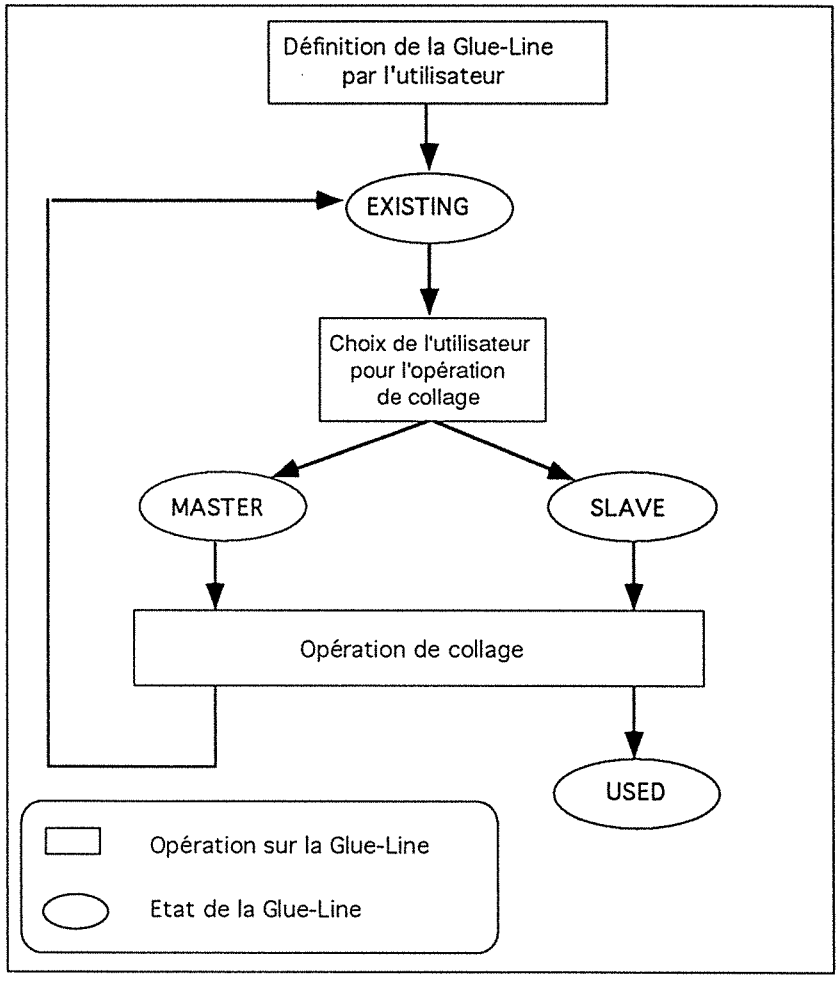


Figure 5.3 Diagramme des états d'une *Glue-Line*.

5.6 Conclusion

L'idée principale sur laquelle repose la méthode interactive de construction du modèle est de créer un lien entre deux surfaces en collant celles-ci par l'intermédiaire de deux lignes de colle. Ces *Glue-Lines* sont définies et utilisées par l'utilisateur; c'est la raison pour laquelle cette méthode est dite interactive.

Comme nous l'avons précisé plus haut, la création d'un lien entre deux surfaces se fait en deux étapes. Dans un premier temps, l'utilisateur doit définir une *Glue-Line* sur chaque surface à coller, à l'endroit où il désire faire la jonction entre les deux surfaces. Ensuite, les deux *Glue-Lines* désignées sont fusionnées pour réaliser effectivement le *collage des deux surfaces.*

Chapitre 6

Création des Glue-Lines

6.1 Introduction

Un *Glue-Line* est une ligne polygonale possédant des informations complémentaires sur la topologie de la surface sur laquelle elle a été créée (ie sa *Surface-Mère*) (voir définition dans Chapitre 5). Deux étapes sont nécessaires à la création d'une *Glue-Line*:

1. **Création de la géométrie de la *Glue-Line*:**
   Cette première étape nécessite l'intervention de l'utilisateur et permet de définir la forme et la position géométrique de la *Glue-Line*.

2. **Création des informations topologiques de la *Glue-Line*:**
   Cette seconde étape est réalisée de façon automatique et consiste à créer les informations topologiques attachées à chaque segment de la *Glue-Line*.

6.2 Création de la géométrie de la Glue-Line

Trois méthodes ont été imaginées pour créer la géométrie d'une *Glue-Line* définie sur une surface donnée [5]:

1. Méthode interactive

2. Méthode du plus court chemin

3. Méthode conditionnée par le collage

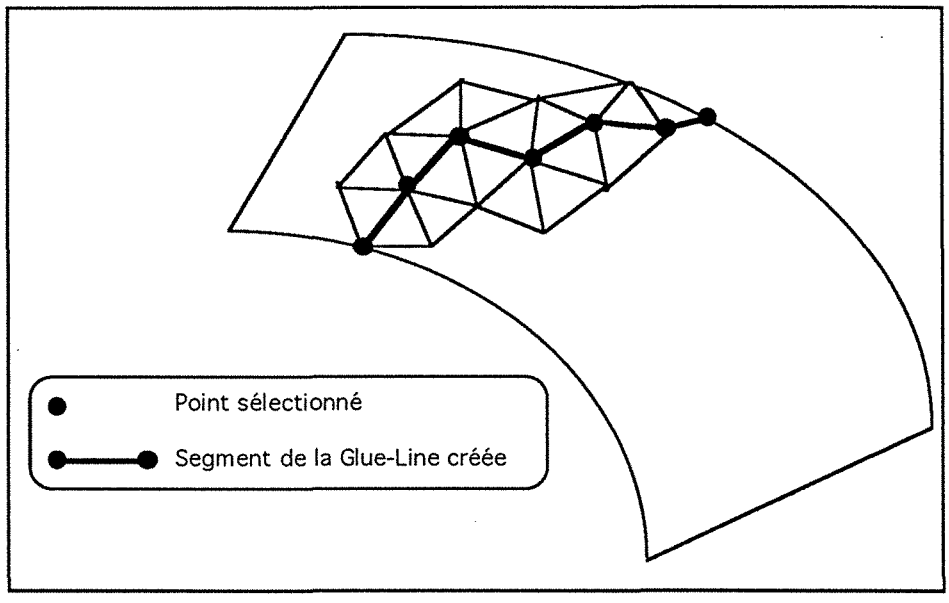


Figure 6.1 Création d'une *Glue-Line* par la méthode interactive.

6.2.1 Méthode interactive

Cette méthode consiste à créer la *Glue-Line* de façon interactive et exhaustive en sélectionnant l'un après l'autre les points de la *Surface-Mère* qui composeront la future *Glue-Line* (voir Figure 6.1). Cette méthode interactive permet de créer une *Glue-Line*, à l'endroit précis où celle-ci doit se trouver. On peut également imaginer que l'utilisateur dispose d'un fichier contenant la description géométrique de la *Glue-Line* (vertices et segments correspondant à des arêtes de la *Surface-Mère*). L'avantage de cette méthode réside donc dans le fait que l'utilisateur peut créer une *Glue-Line* là où il le désire, sans approximation, dans le cas où il a la connaissance exacte de la géométrie de cette ligne.

Cette méthode, qui est la première que nous ayons implantée, présente cependant un inconvénient majeur, à savoir sa lourdeur d'utilisation. En effet, elle s'avère pratiquement inutilisable dans le cas d'un modèle complexe comportant de nombreuses surfaces pour lesquelles des liens multiples doivent être créés.

6.2.2 Méthode du plus court chemin

Cette méthode est basée sur le principe de la méthode du plus court chemin entre deux points [3] (voir Figure 6.2). Elle consiste à sélectionner trois points de la *Surface-Mère*:

- deux points A et B correspondant aux extrémités de la future *Glue-Line*,

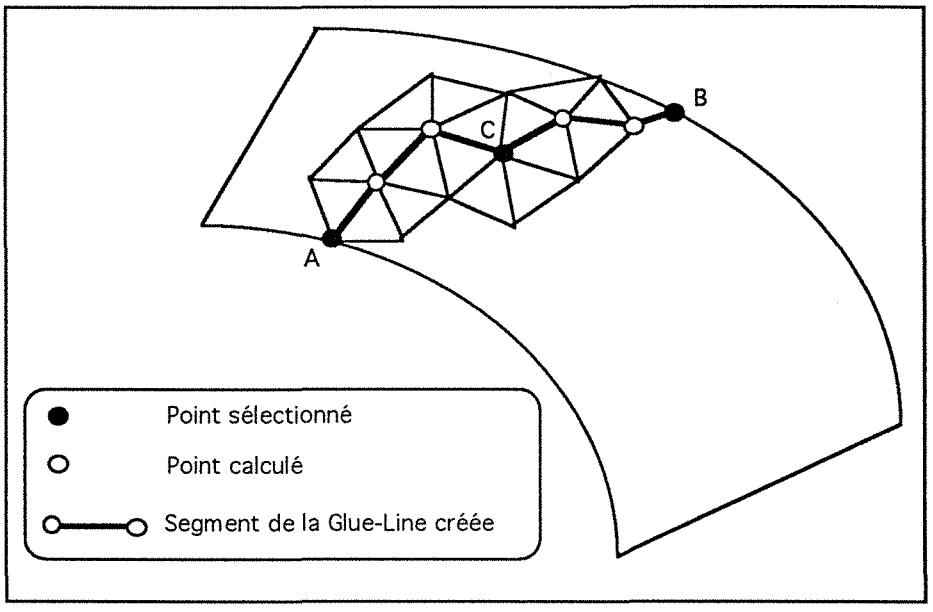


Figure 6.2 Création d'une *Glue-Line* par la méthode du plus court chemin.

- un point intermédiaire C par lequel la *Glue-Line* doit obligatoirement passer[1].

L'idée est de définir un chemin entre les points A et B en utilisant l'étape C de telle façon que la somme des distances (AC) et (CB) soit minimale. Le calcul du plus court chemin entre les points A et B s'effectue de façon récursive en examinant les arêtes de la *Surface-Mère*. La géométrie de la ligne ainsi définie correspond parfaitement à celle de la *Surface-Mère*.

Cette méthode s'avère moins lourde que la précédente, mais également moins précise puisque c'est, de part son principe, une méthode d'approximation dépendant entièrement du maillage de la *Surface-Mère*. En effet, plus le maillage de cette surface est fin, plus la définition -et donc la forme globale- de la *Glue-Line* est précise. En situation réelle, où de nombreux liens doivent être créés entre les surfaces, ce type de création de *Glue-Line* devient particulièrement impossible à utiliser. En fait, c'est le principe même des deux méthodes précédentes qui doit être modifié, à savoir la création indépendante de chaque *Glue-Line*.

6.2.3 Méthode conditionnée par le collage

Une *Glue-Line* est une structure créée par l'utilisateur dans l'optique d'être fusionnée à une autre. La méthode conditionnée par le collage, plus évoluée que les deux précédentes, permet de créer une nouvelle *Glue-Line* à partir d'une *Glue-Line* existante, en sachant que ces deux *Glue-Lines* doivent être fusionnées par la suite (voir Figure 6.3).

[1] Entre deux points A et B, il existe plusieurs chemins possibles. C'est pour être sûr de choisir le bon que l'on introduit l'étape C.

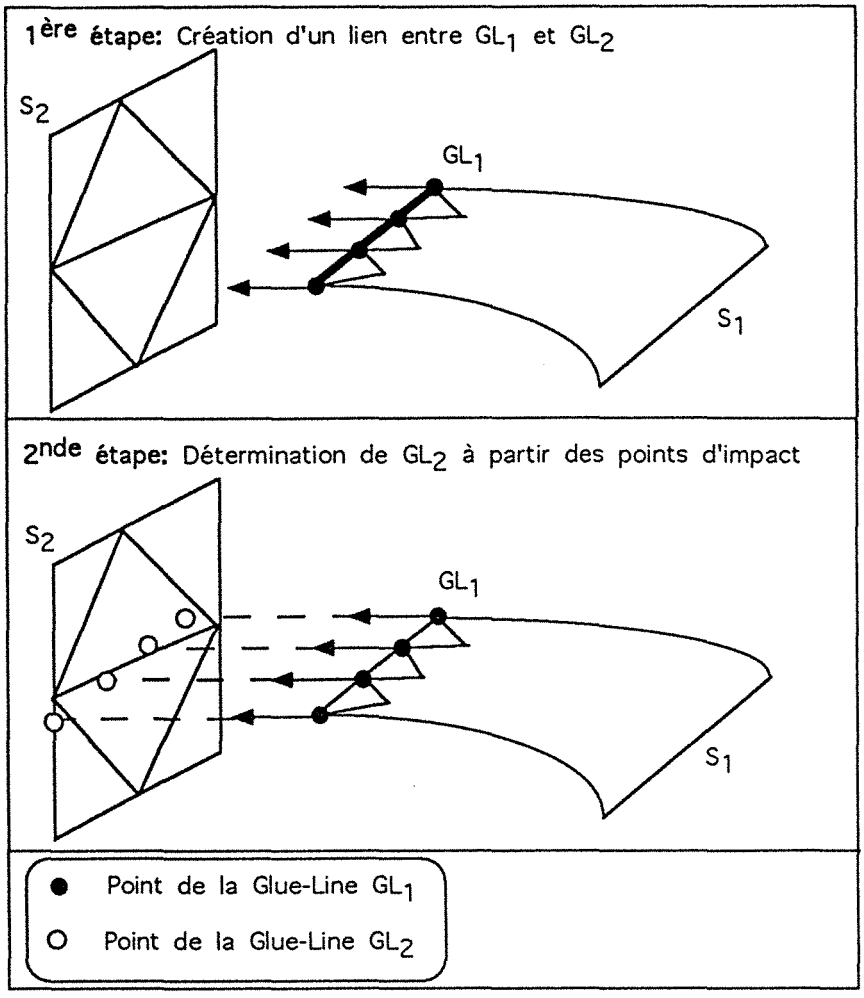


Figure 6.3 Création d'une *Glue-Line* par la méthode conditionnée par le collage.

Imaginons que l'on désire *coller deux surfaces* S_1 et S_2.
Soit GL_1 une *Glue-Line* située sur le bord de la surface S_1 et créée à l'aide de l'une des deux méthodes présentées précédemment.
Soit GL_2 la *Glue-Line* définie sur S_2 et avec laquelle GL_1 doit être fusionnée pour *coller les deux surfaces*. GL_2 n'existe pas encore.

1ère étape: Création d'un lien entre la *Glue-Line* GL_1 et la surface S_2

Dans un premier temps, un vecteur de tir[2] en direction de la surface S_2 est associé à chaque point de la *Glue-Line* GL_1. Ceci permet de définir pour chaque point de la *Glue-Line* une droite

[2]La notion de vecteur de tir a été introduite lors de la mise en place de la contrainte On-Tsurf [22] dont un bref résumé sera proposé dans la partie suivante (voir Chapitre 8, Paragraphe 8.2).

ayant pour origine le point lui-même et pour vecteur directeur le vecteur de tir. Ces vecteurs peuvent être définis de trois façons:

1. **Interactivement:**
   L'utilisateur a la possibilité de définir pour tout point du bord d'une surface un vecteur de tir. La direction initiale de ce vecteur est calculée automatiquement en trouvant le point de la surface-destination S_2 qui minimise la distance entre l'origine du vecteur et S_2. Cette direction peut ensuite être modifiée de façon interactive.

2. **Automatiquement:**
   Pour chaque point A_i de la *Glue-Line* GL_1, la direction du vecteur de tir qui lui est associé est calculée automatiquement en trouvant le point B_j de la surface S_2 qui minimise la distance $(A_i\ ,\ B_j)$.

3. **Semi-automatiquement:**
   L'utilisateur définit de façon interactive un vecteur de tir à chaque extrémité du bord choisi. Les vecteurs associés aux points du bord compris entre ces deux extrémités sont calculés automatiquement par interpolation des deux vecteurs-extrémité. Ce calcul s'effectue à l'aide de l'algorithme D.S.I.

2ème étape: Détermination de la *Glue-Line* GL_2 à partir des points d'impact

Cette étape consiste à calculer l'intersection entre la droite associée à chaque point de la *Glue-Line* GL_1 et la surface S_2. Le résultat est une suite de points (autant que de points de la *Glue-Line* GL_1) sur la surface S_2[3]. Ces points ne sont pas toujours (et même très rarement) des noeuds de la surface S_2. Or, pour la construction des structures topologiques de la *Glue-Line* GL_2, il est nécessaire que celle-ci soit composée de noeuds de sa *Surface-Mère*. Pour répondre à cette exigence, il nous faut modifier la topologie de la surface S_2 en lui ajoutant des noeuds et des triangles.

3ème étape: Modification locale de la topologie de la surface S_2

A ce stade, il est à noter que:

- Il n'est pas obligatoire d'avoir deux points d'impact successifs dans deux triangles adjacents.

- Il peut y avoir plusieurs points d'impact dans un même triangle.

On peut envisager différentes méthodes pour résoudre ce problème[4].

[3] Le calcul des points d'impact permet également de connaitre, pour chaque point, le triangle de la surface S_2 qui le contient: le triangle-impact.

[4] Chacune des solutions présentées ci-dessous est illustrée par une figure pour la situation initiale présentée sur la Figure 6.4. Chaque méthode exposée ci-après y fait référence.

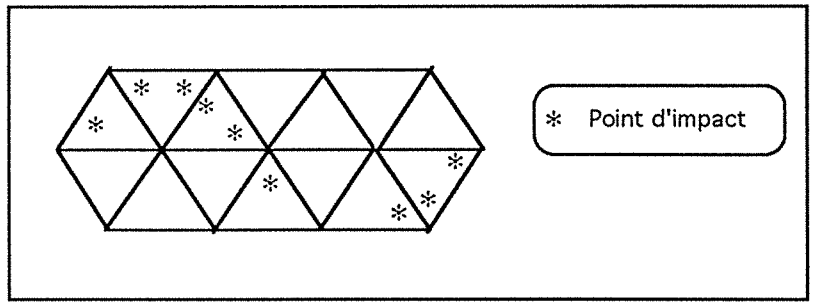


Figure 6.4 Situation de départ de l'exemple étudié.

Méthode 1: Remplacer le point d'impact par le sommet de triangle le plus proche

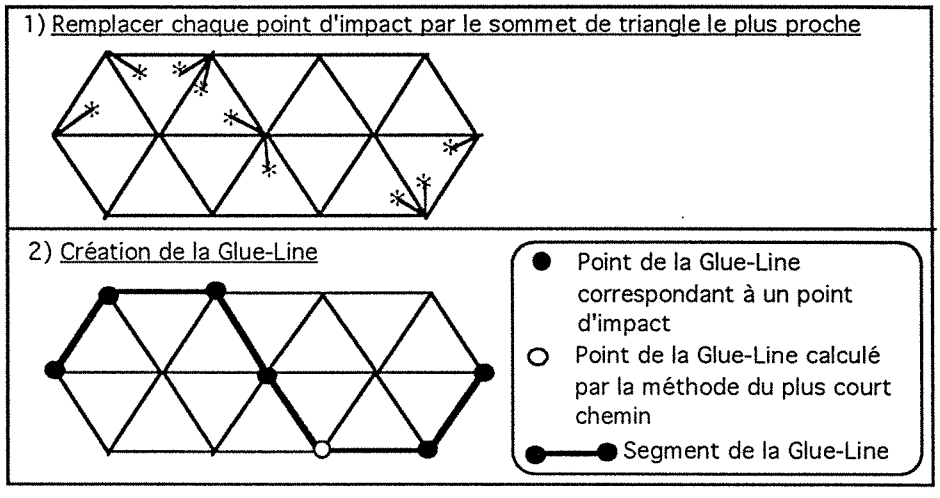


Figure 6.5 Retriangulation de la surface lors de la création d'une *Glue-Line* - Méthode 1.

Dans cette méthode (voir Figure 6.5), chaque point d'impact est déplacé vers le sommet le plus proche de son triangle-impact. Cette solution permet d'éviter la retriangulation, même locale, de la surface S_2. Les points d'impact étant fusionnés avec des noeuds de la surface, il suffit ensuite de calculer le plus court chemin entre deux points consécutifs pour obtenir les points et les segments définissant la géométrie de la *Glue-Line* GL_2.

Malgré l'avantage que cette solution présente, à savoir la conservation de la topologie de la surface initiale, elle s'avère inutilisable de part le peu de précision du résultat obtenu. En effet, plusieurs points d'impact peuvent se retrouver au même sommet de triangle, produisant ainsi une définition très grossière de la *Glue-Line* GL_2.

Méthode 2: Diviser chaque triangle-impact

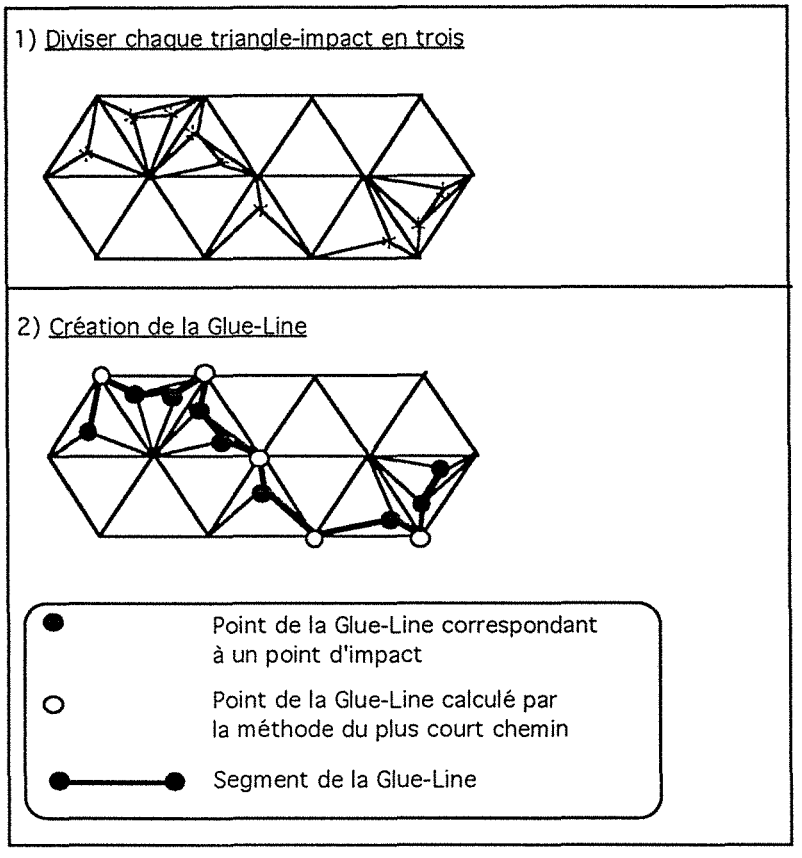


Figure 6.6 Retriangulation de la surface lors de la création d'une *Glue-Line* - Méthode 2.

Cette solution (voir Figure 6.6) s'inspire de l'opération de **Split** existant dans GOCAD [3]. Chaque triangle-impact est divisé en trois nouveaux triangles ayant pour sommet commun le point d'impact contenu par le triangle initial. Les points d'impact étant devenus des sommets de triangle, il suffit ensuite de calculer le plus court chemin entre deux points consécutifs pour obtenir la géométrie de la *Glue-Line* GL_2. Dans le cas d'un triangle-impact contenant plusieurs points d'impact, l'opération de division d'un triangle est répétée pour chacun des points.

Cette solution permet de respecter parfaitement les données, à savoir les points d'impact, mais produit une *Glue-Line* à l'aspect "hâché". De plus, dans le cas d'un triangle-impact contenant plusieurs points d'impact (ce qui peut être le cas si le maillage de la surface initiale n'est pas très fin), cette solution conduit à des modifications importantes de la topologie de la surface initiale.

Méthode 3: Remplacer les points d'impact par des points sur les edges

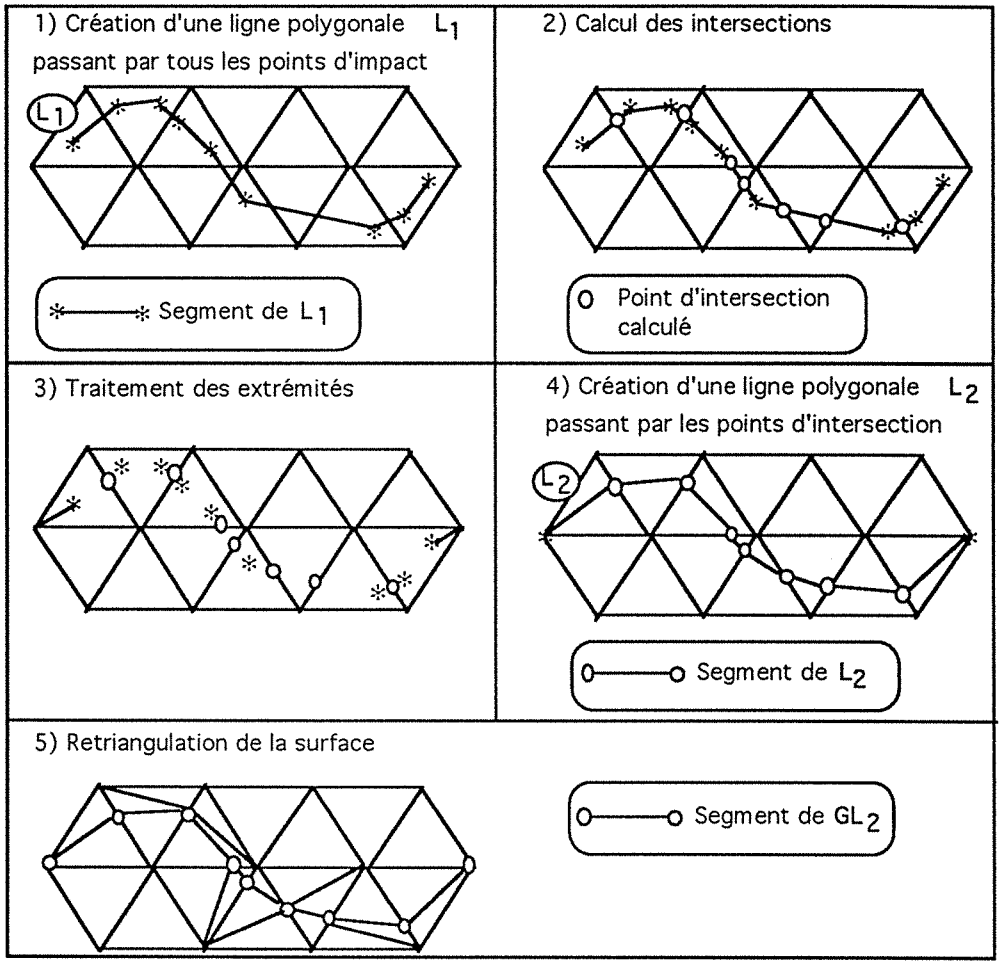


Figure 6.7 Retriangulation de la surface lors de la création d'une *Glue-Line* - Méthode 3.

Cette solution (voir Figure 6.7) se déroule en cinq phases:

1. Création d'une ligne polygonale L_1 passant par tous les points d'impact. Cette ligne est définie par des points A_i et des segments s_j (1 sur Figure 6.7).

2. Pour chaque segment s_j de la ligne L_1, calcul de l'intersection entre ce segment et le plan passant par l'edge d'un triangle de la surface S_2 et orthogonal au segment s_j. Chaque point-intersection se trouve sur un edge de triangle (2 sur Figure 6.7).

3. Traitement des extrémités de la ligne.
   Les deux points d'impact situés aux extrémités de la ligne L_1 sont déplacés vers le sommet

de triangle le plus proche (3 sur Figure 6.7). Dans le cas d'une *Glue-Line* fermée, donc sans extrémité, ce traitement n'est pas effectué.

4. Création d'une nouvelle ligne L_2 passant par les points d'intersection calculés. Les extrémités de L_2 sont identiques à celle de L_1, après déplacement de chacune d'elles vers le noeud le plus proche (4 sur Figure 6.7).

5. Retriangulation de la surface, le long de la ligne L_2 (5 sur Figure 6.7).
Le seul cas de figure rencontré dans cette méthode est celui d'un triangle initial traversé par un segment de la ligne L_2 (voir Figure 6.8). La méthode de retriangulation consiste à choisir entre les deux possibilités proposées sur le schéma, en fonction d'un "critère de beauté" exposé dans [3]. Ce critère est calculé de façon à éviter les triangles plats, donc à choisir les triangles les plus équilatéraux.

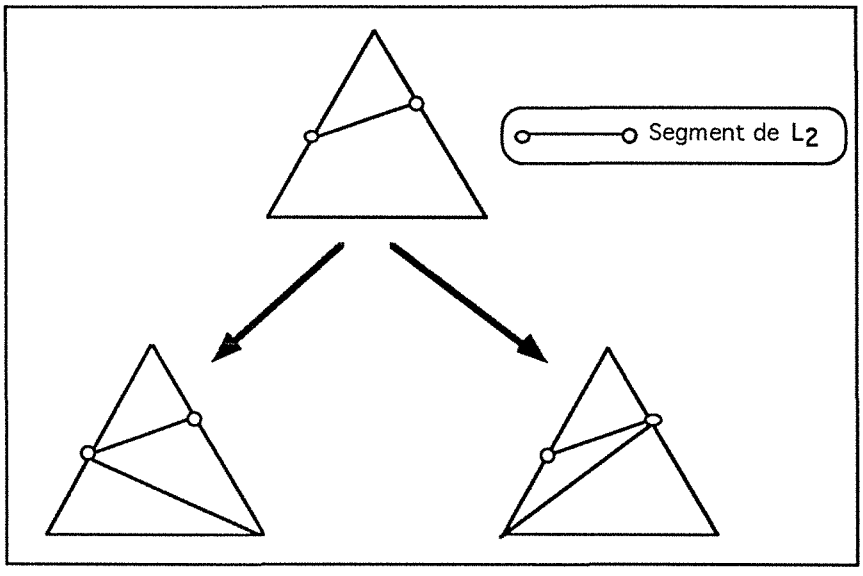


Figure 6.8 Division d'un triangle par la Méthode 3.

La géométrie de la ligne L_2 donne la définition de la *Glue-Line* GL_2.

6.3 Création des informations topologiques de la Glue-Line

Après création de la géométrie de la *Glue-Line* par l'utilisateur, à l'aide de l'une des trois méthodes présentées précédemment, la définition de la *Glue-Line* doit être complétée par la création des informations topologiques qui lui sont associées. Cette seconde phase dans la définition d'une *Glue-Line* est réalisée de façon automatique. Comme nous l'avons précisé lors de la définition de la structure *Glue-Line* (voir Chapitre 5), l'information topologique d'une

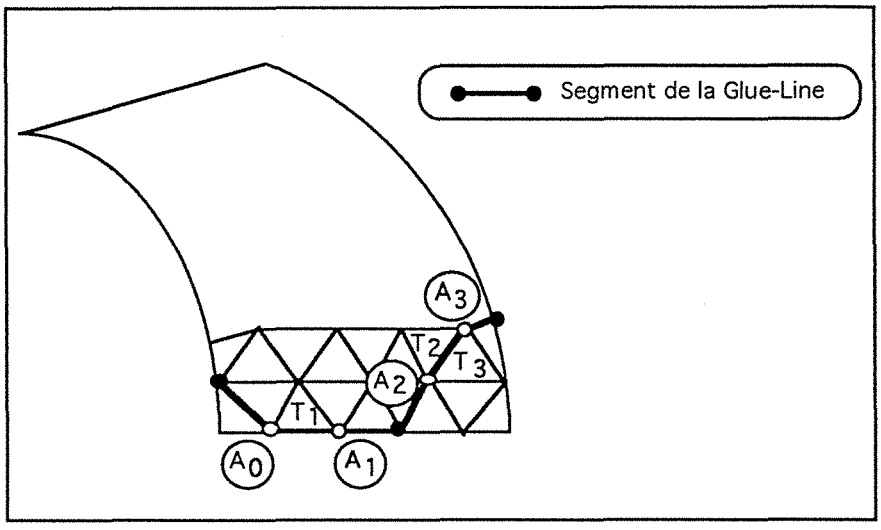


Figure 6.9 Création de la topologie d'une *Glue-Line*.

Glue-Line est attachée à chacun des segments qui la composent, et correspond aux triangles de la surface adjacents à la ligne.

La contrainte majeure lors de la création de la géométrie de la *Glue-Line* était de faire correspondre tous les points de cette ligne avec des noeuds de sa *Surface-Mère*. Cette contrainte étant respectée par chacune des trois méthodes disponibles, tous les segments composant la *Glue-Line* correspondent à des edges de triangles de la surface. La création de la topologie de la *Glue-Line* s'avère donc aisée et consiste à retrouver, pour chaque segment de la *Glue-Line*, les deux triangles de la surface ayant un edge commun avec ce segment. Deux cas peuvent se présenter (voir Figure 6.9):

1. **le segment de la *Glue-Line* est sur le bord de la surface:**
   Dans ce cas, un seul triangle de la surface a un edge commun avec le segment ($[A_0,A_1]$ sur la figure 6.9). Les deux triangles associés au segment de la *Glue-Line* sont donc identiques (triangle T_1).

2. **le segment de la *Glue-Line* est interne à la surface:**
   Dans ce cas, il existe deux triangles de la surface ayant un edge commun avec le segment ($[A_2,A_3]$ sur la figure 6.9). Ces deux triangles (T_2 et T_3) sont associés au segment de la *Glue-Line*.

6.4 Création des Tlinks

Nous avons dit précédemment que la *Glue-Line* intervient dans le découpage d'une surface en *Faces*. En conséquence, toute nouvelle *Glue-Line* constitue une nouvelle frontière pour les *Faces*

auxquelles elle est associée. Sachant que la description des adjacences s'effectue au niveau des bords des *Faces*, il en résulte que de nouvelles structures *Tlinks* doivent être créées suite à chaque création d'une *Glue-Line*. Cette opération est réalisée pour chaque segment de la nouvelle *Glue-Line* et consiste à créer, pour chaque triangle associé au segment courant de la *Glue-Line*, une structure *Tlink* (voir Figures ?? et ??).
Soit s_i le $i^{ème}$ segment de la nouvelle *Glue-Line*.
Soient T_1 et T_2 les deux triangles adjacents à ce segment (T_1 et T_2 peuvent être identiques dans le cas où le segment est situé sur le bord de la *Face*).

T_1 et T_2 appartiennent à la même *Face*. Donc, si la contrainte de consistance de l'orientation des triangles d'une *Face* est respectée, ils sont tous deux orientés de la même façon. D'autre part, T_1 et T_2 sont adjacents par l'intermédiaire du segment s_i. Les deux relations d'adjacence du triangle T_1, à l'edge s_i, sont donc:

- $[T_1\ ,\ '+']$ est lié à $[T_2\ ,\ '+']$.
- $[T_1\ ,\ '-']$ est lié à $[T_2\ ,\ '-']$.

D'où le *Tlink* associé au triangle T_1:

$$Tlink(T_1, s_i) = \{[T_2,'+'], [T_2,'-']\}$$

De la même façon, si le second triangle associé au segment s_i de la *Glue-Line*, T_2, est différent de T_1, le *Tlink* suivant lui est associé:

$$Tlink(T_2, s_i) = \{[T_1,'+'], [T_1,'-']\}$$

6.5 Conclusion

La création des *Glue-Lines* est entièrement interactive. Ceci se révèle à la fois un avantage et un inconvénient. L'avantage réside dans le fait que l'interactivité offre à l'utilisateur la possibilité de modifier à tout moment la topologie de son modèle jusqu'à obtention du résultat attendu. L'inconvénient de cette interactivité trop importante se révèle lors de la manipulation d'un modèle composé d'une multitude de surfaces. La création des *Glue-Lines* devient alors quasiment inutilisable pour tout utilisateur normalement constitué. C'est la raison pour laquelle nous nous sommes orientés par la suite vers une méthode automatique de création du modèle, laquelle sera exposée dans la partie suivante.

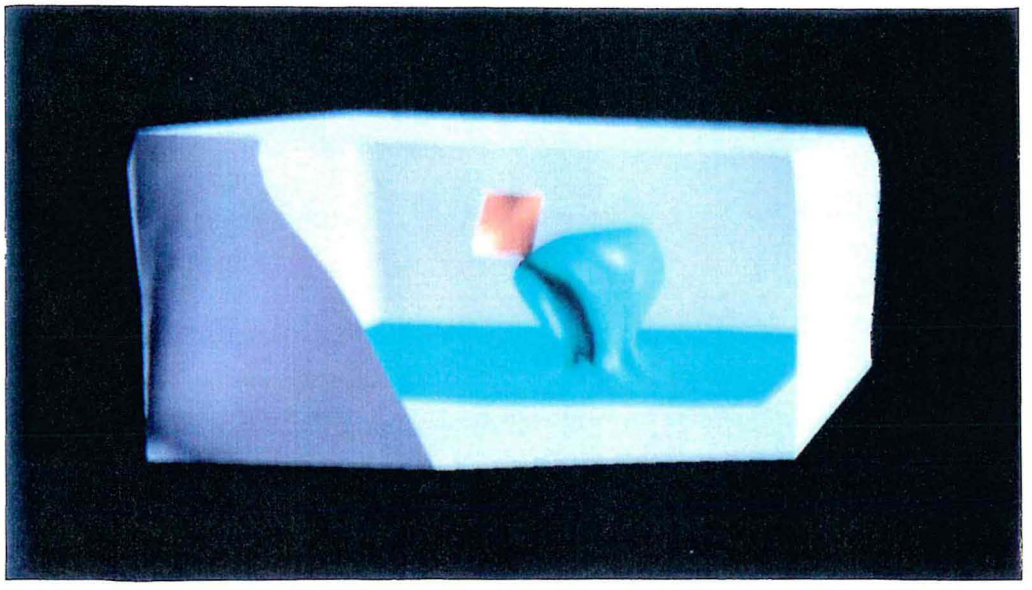


Figure 6.10 Modèle Surfacique utilisé pour la construction interactive du modèle volumique. On distingue un dôme de sel et une lentille, le tout compris dans une boite limitant l'espace de modélisation (le plan frontal de la caméra a été avancé pour permettre de visualiser l'intérieur de la boite).

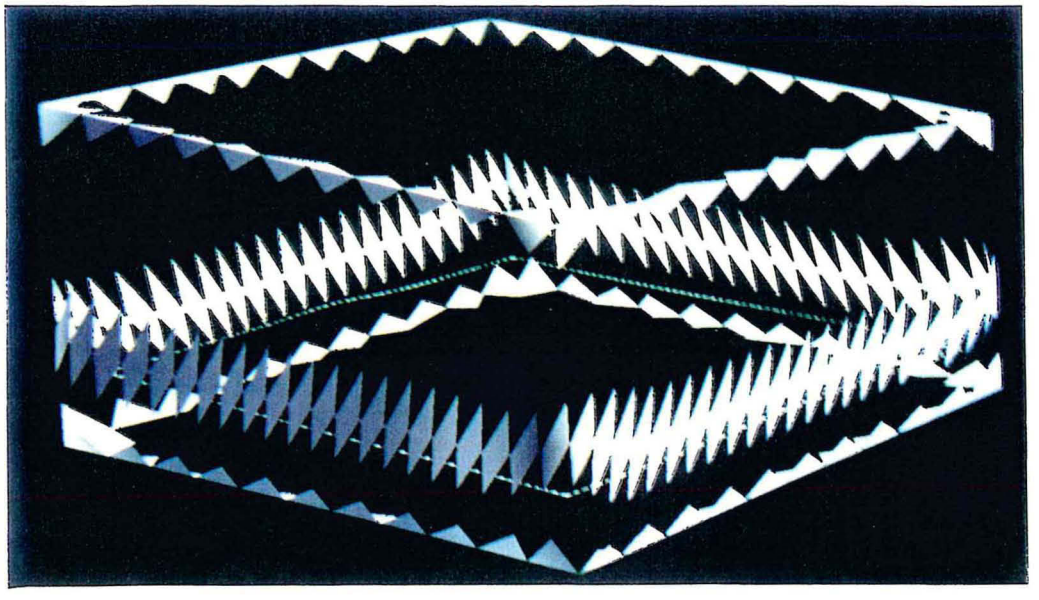


Figure 6.11 Représentation des *Tlink*s du modèle volumique.

Chapitre 7

Utilisation des Glue-Lines

7.1 Introduction

La notion de *Glue-Line* a été introduite dans un seul but qui est la fusion avec une autre *Glue-Line*. Nous avons dit précédemment que la *Glue-Line* n'a pas d'existence (et pas de signification!) en dehors du domaine de la surface. Il faudrait ajouter qu'une *Glue-Line* n'a pas de sens si elle n'est pas utilisée pour une fusion avec une autre *Glue-Line*.

Cette fusion de deux *Glue-Lines* (définies sur des surfaces différentes) permet de créer un lien physique, et par là des liens topologiques, entre deux surfaces; c'est ce que l'on nomme le *collage de deux surfaces*. Cette opération s'effectue à deux niveaux:

- **Association géométrique:**
  la *fusion de deux Glue-Lines* se traduit par le partage de leurs géométries.
- **Association topologique:**
  le *collage de deux surfaces* par l'intermédiaire de deux *Glue-Lines* crée un lien topologique entre les *Faces* respectives des surfaces.

7.2 Association géométrique

Deux procédés ont été mis au point pour *coller deux surfaces* par l'intermédiaire de *Glue-Lines* (voir Figure 7.1). Le premier, appelé **Collage Direct** (voir Paragraphe 7.2.2) nécessite de modifier la topologie de l'une des deux *Glue-Lines* à fusionner. Cette méthode fait appel aux notions de *Glue-Lines* SLAVE et MASTER exposées brièvement dans le chapitre 5 et que nous développerons de façon plus détaillée dans le paragraphe suivant.

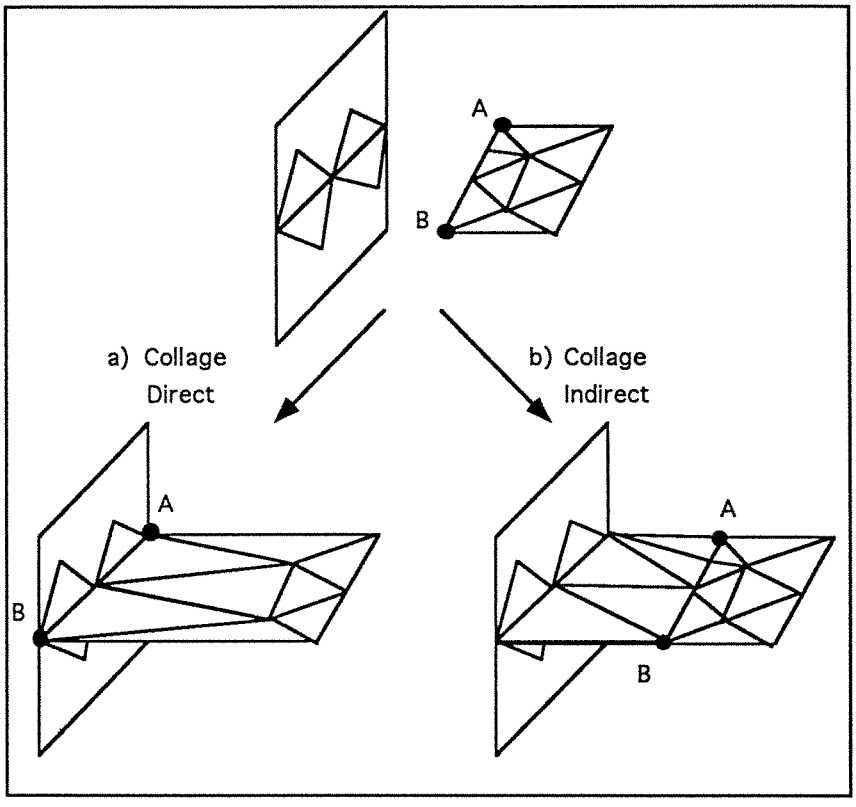


Figure 7.1 *Collage* direct ou indirect de deux surfaces.

La modification de la structure d'une *Glue-Line* provoque automatiquement la modification locale de la structure de sa *Surface-Mère*, d'après la définition-même d'une *Glue-Line* (voir Chapitre 5). C'est pourquoi un second procédé, visant à éviter toute modification des surfaces initiales, a été mis au point; c'est le **Collage Indirect** (voir Paragraphe 7.2.3).

7.2.1 Glue-Line SLAVE et Glue-Line MASTER

La méthode de *collage* direct de deux surfaces, présentée dans le paragraphe suivant, consiste à fusionner une *Glue-Line* définie sur l'une des deux surfaces et une autre *Glue-Line* définie sur l'autre surface. Pour cela, les deux *Glue-Lines* doivent avoir la même structure, c'est-à-dire le même nombre d'atomes et de segments. Pour répondre à cette exigence, nous avons choisi d'adapter la géométrie de l'une des deux *Glue-Lines* à celle de l'autre. En d'autres termes, l'une des deux *Glue-Lines* est choisie comme support de l'autre: c'est la **Géométrie de Référence**. Le choix de cette **Géométrie de Référence** ne se fait pas de façon aléatoire mais dépend entièrement de l'utilisateur. En effet, c'est l'ordre dans lequel ce dernier fait intervenir les deux *Glue-Lines* dans le lancement de l'opération de *collage* qui détermine la **Géométrie de**

Référence.

Par exemple, considérons une *Glue-Line* GL_1 que l'utilisateur désire fusionner à une seconde *Glue-Line* GL_2. Le *collage* direct peut être lancé de deux façons:

1. "Coller GL_1 sur GL_2":
   Dans ce cas, GL_2 est choisie comme **Géométrie de Référence.**

2. "Coller GL_2 sur GL_1":
   Dans ce cas, GL_1 est choisie comme **Géométrie de Référence.**

La *Glue-Line* choisie comme **Géométrie de Référence** passe du statut EXISTING au statut MASTER. L'autre *Glue-Line*, devant se "soumettre" à la *Glue-Line* MASTER, passe du statut EXISTING au statut SLAVE. Après l'opération de *fusion*, les statuts des *Glue-Lines* (voir Figure 5.3 dans Chapitre 5, Paragraphe 5.5) deviennent les suivants:

- **pour la** *Glue-Line* MASTER:
  La *Glue-Line* MASTER a servi de "support" à une *Glue-Line* SLAVE. En d'autres termes, la structure de la *Glue-Line* SLAVE ainsi que celle de sa *Surface-Mère* ont été modifiées en fonction de la *Glue-Line* MASTER. En conséquence, on peut imaginer deux philosophies différentes concernant le "devenir" d'une *Glue-Line* MASTER après une opération de *fusion.*

  La première consiste à autoriser la *Glue-Line* MASTER à devenir elle-même une *Glue-Line* SLAVE. Cette solution implique que la *Glue-Line* subisse des modifications géométriques et topologiques afin de s'adapter à une nouvelle *Glue-Line* MASTER. Donc, les *Faces* précédemment associées à cette *Glue-Line* doivent elles-aussi subir ces modifications. Cette solution est réalisable, mais s'avère très lourde à gérer.

  La seconde solution consiste à "geler" toute *Glue-Line* MASTER dans cet état. Cette méthode est beaucoup plus simple à gérer, mais restreint le champ des modèles constructibles.
  Nous avons choisi de mettre en oeuvre la première solution. En conséquence, une *Glue-Line* MASTER passe à l'état EXISTING après une opération de *fusion.* Elle peut alors prendre, selon le choix de l'utilisateur, le statut de SLAVE ou de MASTER.

- **pour la** *Glue-Line* SLAVE:
  Une *Glue-Line* SLAVE perd toute utilité une fois la fusion réalisée. Elle est toutefois conservée pour pouvoir être restaurée lors d'une opération de *décollage.*

7.2.2 Collage Direct

Soit GL_1 une *Glue-Line* définie sur la surface S_1.
Soit GL_2 une *Glue-Line* définie sur la surface S_2.
GL_1 a été choisie comme *Glue-Line* SLAVE[1] (voir Figure 7.2).

[1] La *Surface-Mère* de la *Glue-Line* SLAVE subit des modifications topologiques lors du *collage* direct. Pour minimiser ces modifications, une *Glue-Line* SLAVE sera toujours choisie sur le bord de sa *Surface-Mère.*

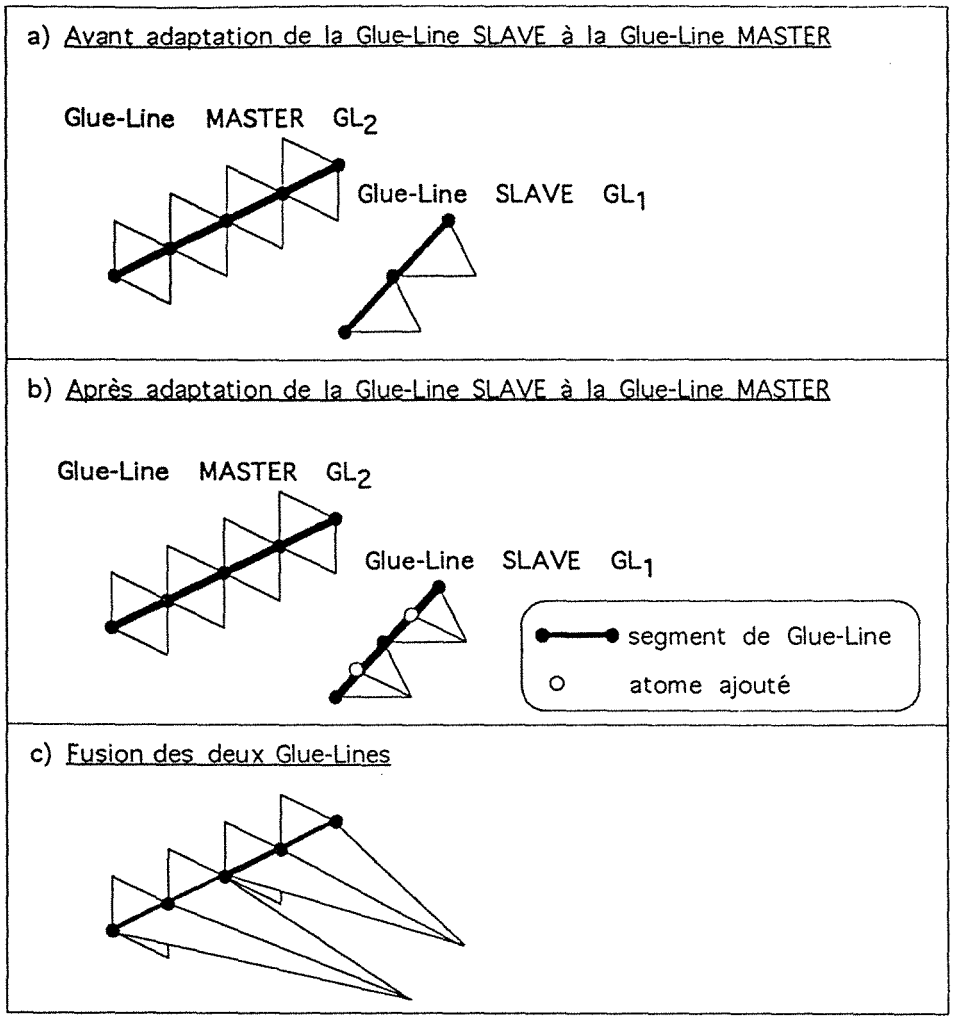


Figure 7.2 *Collage* direct de deux surfaces.

COLLAGE DIRECT: 1ère étape

La première étape dans le *collage* direct de S_1 sur S_2 par l'intermédiaire de leurs *Glue-Lines* respectives consiste à rendre identique le maillage de ces deux *Glue-Lines*. En d'autres termes, dans le cas où les deux *Glue-Lines* en présence n'ont pas le même nombre de points, la géométrie de la *Glue-Line* SLAVE doit être modifiée par l'ajout où la suppression de points, en fonction de la géométrie de la *Glue-Line* MASTER (**Géométrie de Référence**). Or, par définition, la *Glue-Line* partage sa géométrie avec celle de sa *Surface-Mère*. En conséquence, la modification de la *Glue-Line* par l'ajout ou la suppression de points provoque des modifications importantes au niveau de sa *Surface-Mère*.

Le nombre de points à ajouter ou à supprimer est obtenu en calculant la différence entre

le nombre de points définissant la *Glue-Line* MASTER et celui de la *Glue-Line* SLAVE. Si cette différence est positive, autant de points sont ajoutés sur la *Glue-Line* SLAVE. Si la différence est négative, ces points sont supprimés de la *Glue-Line* SLAVE.

Adjonction et Suppression de points: Théorie

L'adjonction et la suppression de points sur la *Glue-Line* SLAVE s'effectuent de la manière suivante [5]:

- **Ajout de n points sur une** *Glue-Line* SLAVE
  Soit N_1 le nombre de segments de la *Glue-Line*.
  - Si $(n \leq N_1)$
    un point est créé au milieu de chacun des n premiers segments de la *Glue-Line*. En d'autres termes, chacun des n premiers segments de la *Glue-Line* est divisé en deux nouveaux segments partageant le point ajouté.
  - Si $(n > N_1)$
    un point est créé au milieu de chacun des N_1 segments de la *Glue-Line*. L'opération est ensuite répétée en utilisant les nouveaux segments de la *Glue-Line*, jusqu'à ce que tous les points nécessaires aient été ajoutés.

  L'ajout de points sur la *Glue-Line* résulte en l'ajout de nouveaux noeuds à sa *Surface-Mère*. Qui dit ajout de noeuds, dit également ajout de triangles, donc modification de la topologie de la surface. Ceci est réalisé en divisant le triangle existant en deux triangles ayant pour sommet commun le noeud ajouté.

- **Suppression de n points sur une** *Glue-Line* SLAVE
  Soit N_2 le nombre de points de la *Glue-Line*.
  - Si $(n \geq N_2)$
    l'opération est impossible, car il est impossible de supprimer plus de points qu'il n'en existe. De même il serait incongru de supprimer tous les points d'une *Glue-Line*, car cette dernière n'aurait plus de points pour la définir, donc plus de points à fusionner à la *Glue-Line* MASTER. Dans ce cas, le problème vient du fait que la différence de maillage entre les deux *Glue-Lines* à fusionner est trop importante. La solution consiste à modifier le maillage de l'une des deux *Surface-Mères*:
    * **Split**, donc ajout de triangles sur la *Surface-Mère* de la *Glue-Line* MASTER.
    * **Unsplit**, donc suppression de triangles sur la *Surface-Mère* de la *Glue-Line* SLAVE.

    Après modification du maillage de l'une des deux *Surface-Mères*, la *Glue-Line* correspondante doit être reconstruite.
  - Si $(n < N_2)$
    le choix des n points à supprimer est délicat, car la topologie et la géométrie de la

Surface-Mère de la *Glue-Line* sont touchés par cette modification. En conséquence, il faut faire en sorte de perdre le moins d'informations possible. Les critères de choix des points à supprimer sont les suivants:

1. Choisir les atomes les moins contraints (au sens de D.S.I.).
2. Choisir les atomes du bord qui apportent le moins d'information géométrique à la surface, c'est-à-dire les atomes qui ne jouent pas un rôle capital dans la définition de la forme générale de la surface. Ces atomes sont ceux partagés par deux segments quasi-colinéaires. En d'autres termes, un atome A_1 partagé par deux segments adjacents s_1 et s_2 est sélectionné si:

$$|\cos(s_1, s_2)| > 0.8$$

La modification de la géométrie de la *Glue-Line*, par l'adjonction ou la suppression de points, provoque une modification du maillage de celle-ci. Sachant que des informations topologiques relatives au triangles adjacents sont attachées à chaque segment d'une *Glue-Line*, cette modification doit être suivie d'un recalcul des triangles adjacents aux segments modifiés afin de conserver la cohérence de la structure *Glue-Line*.

Exemple d'adjonction de points

Dans notre exemple (b sur Figure 7.2), GL_1 est composée de trois points et GL_2 de cinq points. Deux points doivent donc être ajoutés à la *Glue-Line* GL_1. Ces points sont ajoutés sur les segments existants de GL_1.

COLLAGE DIRECT: 2nde étape

Les deux *Glue-Lines* à fusionner ont exactement le même nombre de points. Le *collage* effectif des deux surfaces S_1 et S_2 peut alors se réaliser. Cette opération est effectuée en déplaçant les points de la *Glue-Line* SLAVE vers ceux de la *Glue-Line* MASTER. Sur notre exemple (c sur Figure 7.2), les points de la *Glue-Line* GL_1 sont "étirés" vers ceux de la *Glue-Line* GL_2.

COLLAGE DIRECT: Conclusion

L'inconvénient majeur de cette méthode est qu'elle modifie la structure initiale de la surface associée à la *Glue-Line* SLAVE. C'est pourquoi nous avons imaginé une seconde méthode de *collage* permettant d'éviter toutes modification de la géométrie et de la topologie des surfaces originales: le **collage indirect**.

7.2.3 Collage Indirect

Soit GL_1 une *Glue-Line* définie sur la surface S_1.
Soit GL_2 une *Glue-Line* définie sur la surface S_2.
La méthode de *collage* indirect consiste à créer une nouvelle surface s'appuyant sur les deux

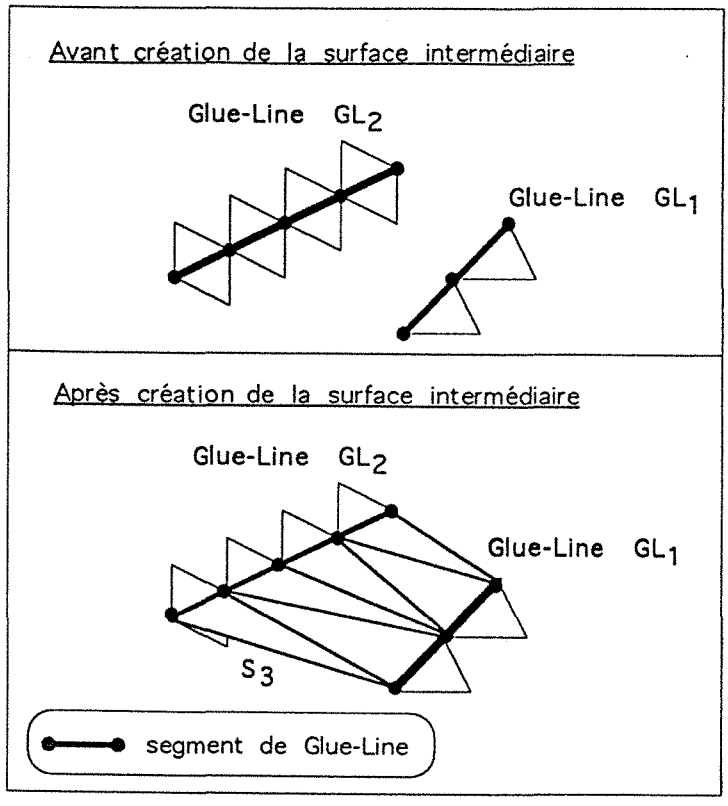


Figure 7.3 *Collage* indirect de deux surfaces.

Glue-Lines à fusionner. Les notions de SLAVE et MASTER ne sont pas fondamentales ici car la géométrie des *Glue-Lines* n'est pas modifiée. La surface créée, que nous nommons **Skin**, sera ensuite considérée comme les autres surfaces du modèle. Du fait que cette surface s'appuie sur les deux *Glue-Lines* de départ, les adjacences sont parfaitement respectées. En fait, cette méthode revient à créer une structure intermédiaire entre deux *Faces*, lesquelles deviennent "adjacentes de façon indirecte".

Sur notre exemple (voir Figure 7.3), S_1 et S_2 sont *collées* par l'intermédiaire de la surface S_3 dont deux des bords correspondent aux *Glue-Lines* GL_1 et GL_2.

7.3 Association topologique

Dans le paragraphe précédent, nous avons présenté les deux méthodes existantes de fusion de deux *Glue-Lines*, au niveau de la géométrie de celles-ci. Comme nous l'avons précisé dans le chapitre 5, paragraphe 5.3, le *collage de deux surfaces* par l'intermédiaire de deux *Glue-Lines* résulte en l'*association* de certaines *Faces* des *Surfaces-Mère* respectives de chaque *Glue-Line*. En conséquence, les adjacences de chaque *Face* mise en cause dans le processus de *fusion des deux Glue-Lines* doivent être mises à jour. Cette modification des adjacences est traitée au

niveau des triangles du bord de chaque *Face* sélectionnée.

Dans la méthode de *collage* direct, les adjacences modifiées se situent au niveau de la *Glue-Line* MASTER, laquelle a servi de support à la fusion. Dans la méthode de *collage* indirect, les adjacences modifiées se situent au niveau des deux *Glue-Lines* ayant servi de support à la création de la surface intermédiaire. La modification des adjacences se traduit par la modification des structures *Tlink* existantes. Pour expliquer ce processus de modification des adjacences lors de la fusion de deux *Glue-Lines*, nous avons choisi de présenter un exemple de *collage* direct[2].

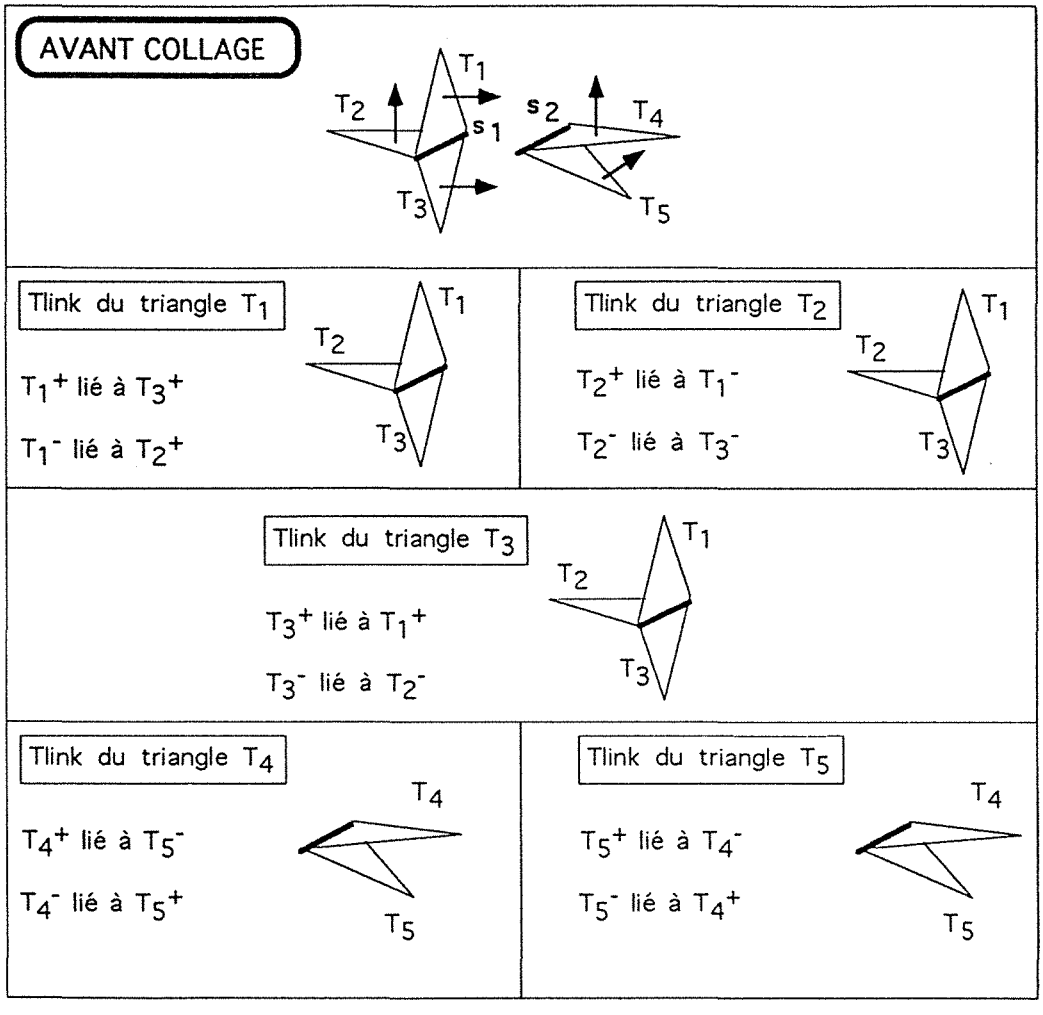


Figure 7.4 Modification des relations d'adjacence - Etat avant *collage*.

[2]Dans le cas du *collage* indirect, le processus ci-dessous doit être exécuté pour les deux *Glue-Lines*.

7.3.1 Notations

Soit s_1 un segment d'une *Glue-Line* MASTER GL_1 (voir Figure 7.4). Trois triangles sont associés au segment s_1: T_1, T_2 et T_3. Pour décrire les relations d'adjacence de ces trois triangles, il existe un *Tlink* pour chacun d'eux:

$$\begin{aligned} tlink_1 &= Tlink(T_1, s_1) \\ tlink_2 &= Tlink(T_2, s_1) \\ tlink_3 &= Tlink(T_3, s_1) \end{aligned}$$

Soit s_2 un segment d'une *Glue-Line* SLAVE GL_2. Deux triangles sont associés au segment s_2: T_4 et T_5. Un *Tlink* est associé à chacun de ces triangles pour décrire ses relations d'adjacence:

$$\begin{aligned} tlink_4 &= Tlink(T_4, s_2) \\ tlink_5 &= Tlink(T_5, s_2) \end{aligned}$$

7.3.2 Recherche des relations d'adjacence modifiées

Examinons la *fusion des deux Glue-Lines* GL_2 et GL_1 au niveau des segments s_1 et s_2.
La fusion du segment s_1 et du segment s_2 conduit à lier les deux triangles de s_2 au segment s_1. En d'autres termes, les triangles de s_2 doivent être insérés entre deux triangles de s_1. Cette insertion détruit certaines relations d'adjacence et en crée d'autres. Pour connaitre les adjacences détruites, et donc pour connaitre les deux triangles entre lesquels les triangles de s_2 doivent être insérés, la méthode choisie est la suivante (voir Figure 7.5):

Calcul des vecteurs unitaires

- Calculer un vecteur unitaire $U_i, i \in [1..3]$ associé à chaque edge de triangle attaché au segment s_1.
- Calculer un vecteur unitaire $V_i, i \in [1..2]$ associé à chaque edge de triangle attaché au segment s_2.

Le vecteur unitaire $\overrightarrow{U}$ d'un triangle T associé à un segment s est calculé de façon à ce que:

- $\overrightarrow{U} \perp s$,
- $\overrightarrow{U}$ contenu dans le plan de T.

La méthode de calcul d'un vecteur unitaire $\overrightarrow{U}$ d'un triangle T associé à un segment s est la suivante (voir Figure 7.6):

1. Soient A et B les deux sommets du triangle T confondus avec le segment s.

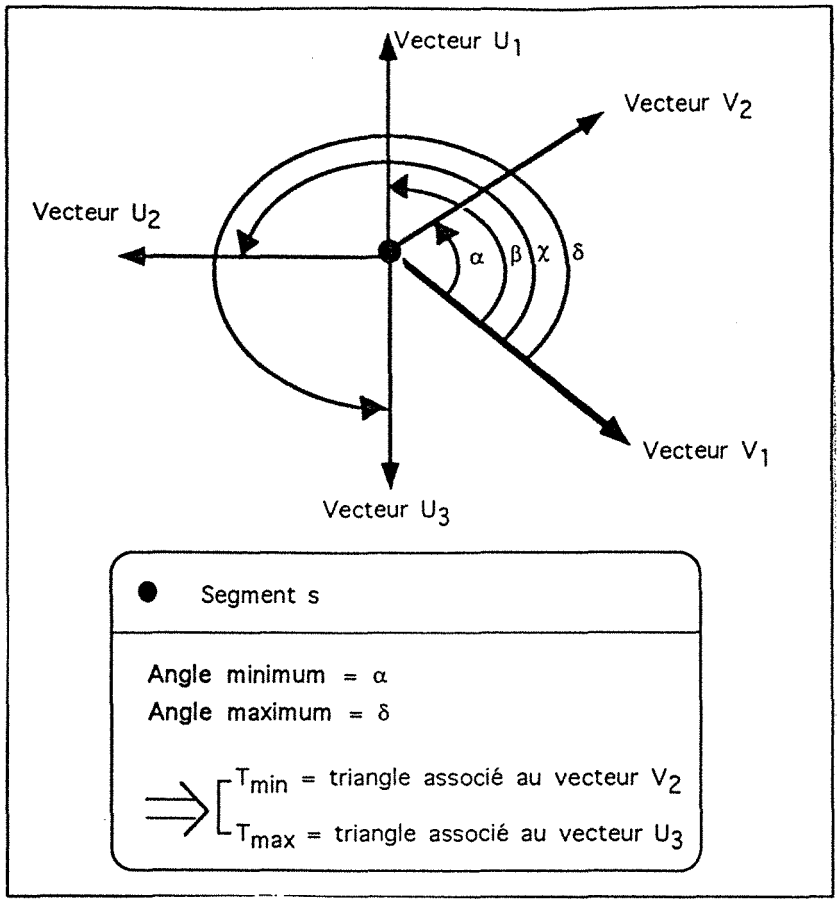


Figure 7.5 Recherche des relations d'adjacence modifiées (Représentation 2D).

2. Soit C le troisième sommet de T.

3. Calcul du vecteur $\overline{AB}$.
   Si la norme de AB est nulle, cela signifie que les deux sommets A et B sont confondus, donc que le triangle T est plat. Ce triangle aurait dû être supprimé si les données initiales avaient été "embellies". Puisque ce n'est pas le cas, ce triangle est supprimé et remplacé par un de ces triangles adjacents.

4. Calcul du vecteur $\overline{AC}$.
   Même remarque que précédemment à propos de la norme du vecteur $\overline{AC}$.

5. Calcul du vecteur $\overline{W}$, égal au produit vectoriel $\overline{AB} \wedge \overline{AC}$.

6. Calcul du vecteur $\overline{U}$, égal au produit vectoriel $\overline{W} \wedge \overline{AB}$.

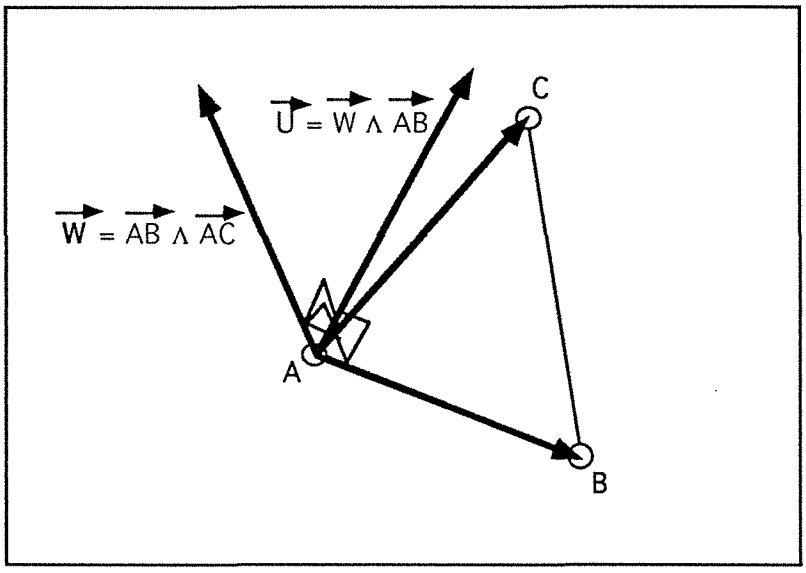


Figure 7.6 Calcul du vecteur unitaire $\overline{U}$ associé au côté (AB) d'un triangle.

7. Diviser le vecteur $\overline{U}$ par sa norme, afin d'obtenir un vecteur unitaire normé.

Les vecteurs calculés sont tous dans le même plan. La suite du problème se résoud ainsi comme un problème 2D.

Calcul des angles algébriques

- Calculer les angles algébriques entre le vecteur V_1 et tous les autres vecteurs unitaires, c'est-à-dire les vecteurs $U_i, i \in [1..3]$ et les vecteurs $V_i, i \in [2..2]$.
- Retrouver les triangles associés aux deux vecteurs formant respectivement le plus petit et le plus grand angle avec le vecteur V_1. Ces triangles (T_{min} et T_{max}) correspondent aux triangles pour lesquels l'insertion d'un nouveau triangle brise une relation d'adjacence.

Exemple

En ce qui concerne notre exemple (voir Figure 7.4), les deux triangles trouvés par la méthode précédente, sachant que le triangle T_5 a été pris arbitrairement comme référence, sont T_4 et T_3[3].

7.3.3 Modification des structures Tlink

Toute opération de *collage* modifie trois relations d'adjacence:

[3]Au lieu de deux triangles, un seul peut être trouvé dans le cas où le segment de la *Glue-Line* MASTER n'a qu'un triangle associé (segment du bord de la surface).

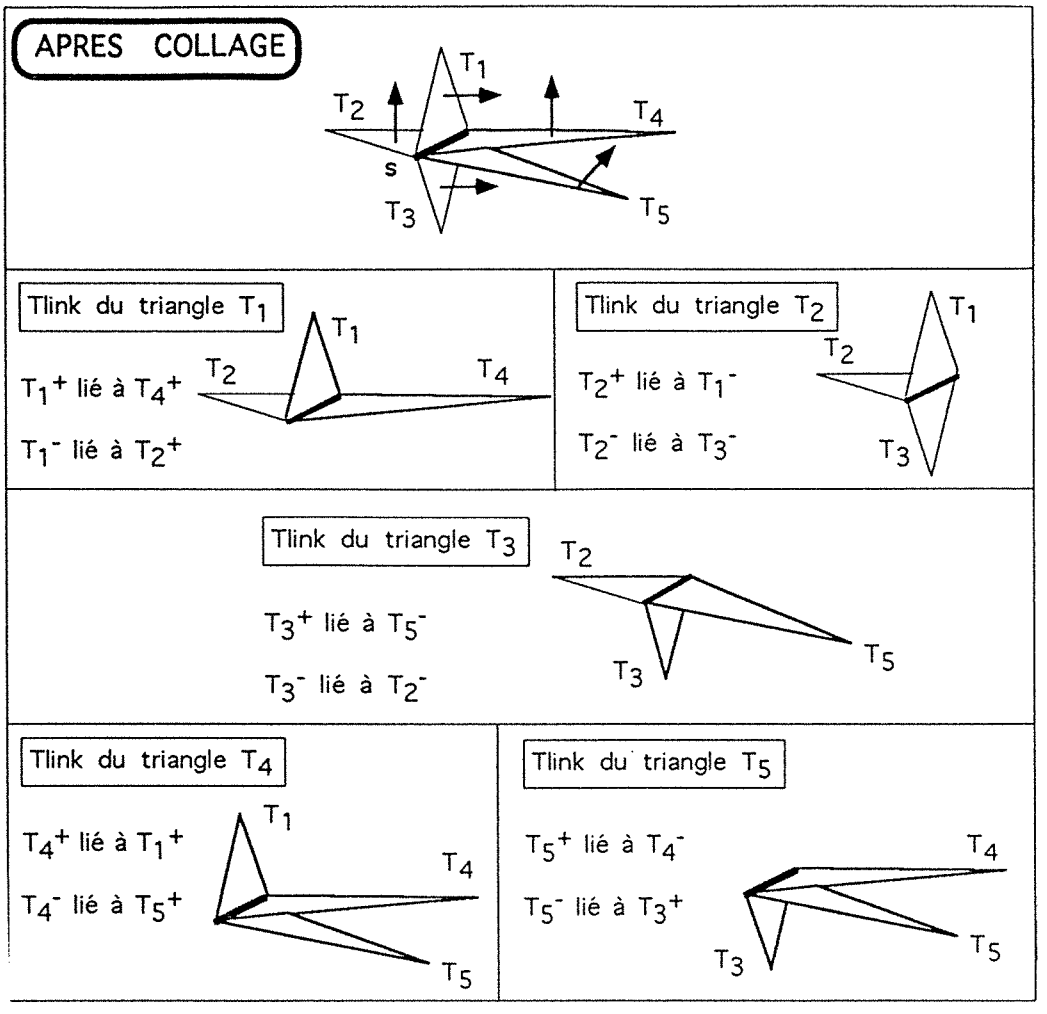


Figure 7.7 Modification des relations d'adjacence - Etat après *collage*.

1. Adjacence de T_{min} avec T_{max}.
2. Adjacence de T_{max} avec T_{min}.
3. Adjacence des triangles associés au segment de la *Glue-Line* SLAVE.

Donc, pour chaque segment résultant de la fusion de deux *Glue-Lines*, plusieurs *Tlinks* doivent être modifiés. Pour cela, il nous faut non seulement connaitre les deux triangles adjacents au triangle de référence, mais également les orientations de ces deux triangles. Ceci est obtenu en testant le signe du produit scalaire de la normale au triangle adjacent et du vecteur unitaire du triangle de référence.

Dans notre exemple (voir Figure 7.7), la fusion des deux *Glue-Lines* modifie les relations d'adjacence des triangles suivants:

- Triangle T_1:
  Le *Triangle-use* $[T_1,'+']$ est adjacent au *Triangle-use* $[T_4,'+']$.
- Triangle T_3:
  Le *Triangle-use* $[T_3,'+']$ est adjacent au *Triangle-use* $[T_5,'-']$.
- Triangle T_4:
  Le *Triangle-use* $[T_4,'+']$ est adjacent au *Triangle-use* $[T_1,'+']$.
- Triangle T_5:
  Le *Triangle-use* $[T_5,'-']$ est adjacent au *Triangle-use* $[T_3,'+']$.

Pour définir les nouvelles relations d'adjacence, on a besoin de quatre nouveaux *tlinks*:

$$\begin{aligned} tlink_1^* &= Tlink(T_1, s) \\ tlink_3^* &= Tlink(T_3, s) \\ tlink_4^* &= Tlink(T_4, s) \\ tlink_5^* &= Tlink(T_5, s) \end{aligned}$$

où s est le segment résultant de la fusion des segments s_1 et s_2, c'est-à-dire le segment de la *Glue-Line* MASTER (donc s_1).

7.4 Exemple de modèle volumique

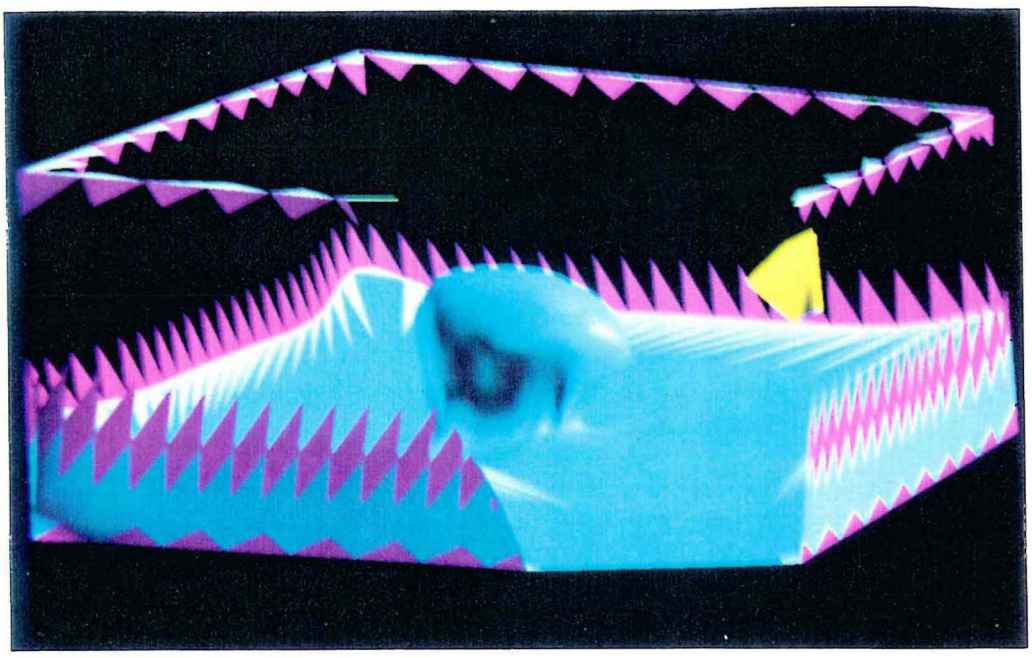


Figure 7.8 Représentation des *Tlinks* du modèle volumique, après fusion de toutes les *Glue-Lines*. Les *Tlinks* associés aux *Glue-Lines* SLAVE sont de couleur rose.

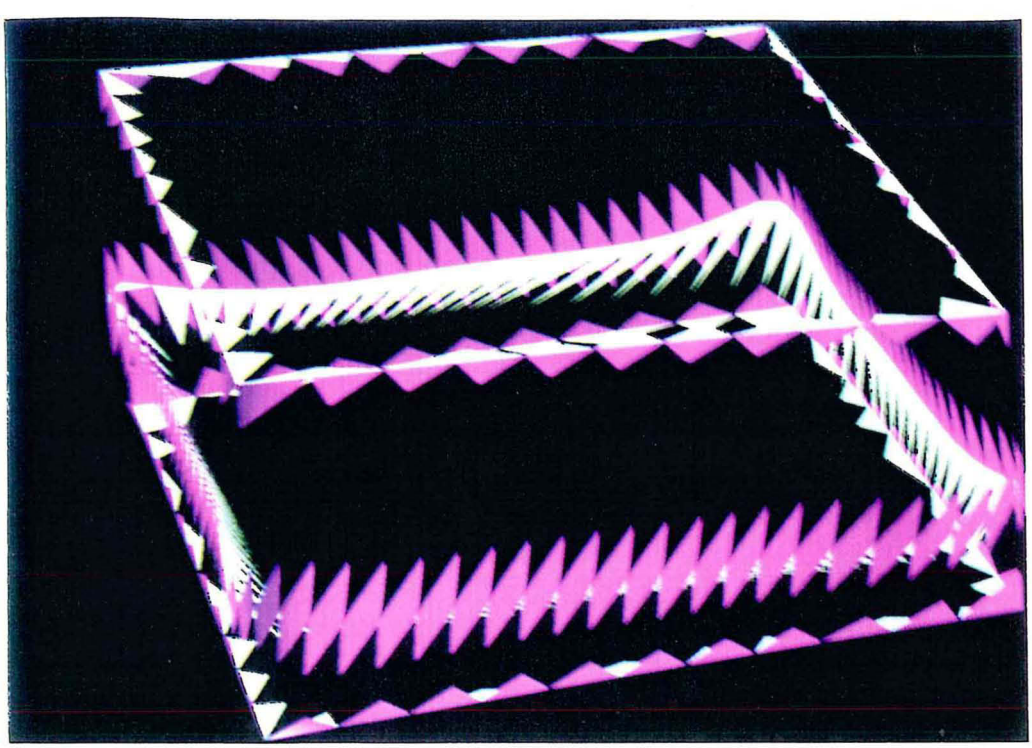


Figure 7.9 Représentation des *Régions* du modèle volumique. On distingue la *Région* correspondant à la couche inférieure du modèle, et celle correspondant à l'intérieur de la lentille (le plan frontal de la caméra a été avancé pour permettre de visualiser l'intérieur de la couche inférieure du modèle).

7.5 Conclusion

Nous avons dit dans le chapitre précédent que le principal avantage à l'utilisation des *Glue-Lines* est que celles-ci permettent la création pas-à-pas du modèle volumique *non-manifold*. De cette façon, l'utilisateur dispose à tout moment de la possibilité de faire marche arrière quand le résultat ne lui convient pas. Ceci est un avantage, mais peut également devenir un inconvénient majeur quand le modèle initial est composé d'un nombre important de surfaces complexes. D'une part la création de chaque *Glue-Line* s'avère très lourd à réaliser. D'autre part, l'utilisateur ne connait que très rarement avec précision la position géométrique exacte de chaque *Glue-Line*. Ces inconvénients, présentés dans le chapitre précédent, sont relatifs à la définition des *Glue-Lines*.

L'utilisation des *Glue-Lines* dans une opération de *collage de deux surfaces*, si elle possède le même avantage que la création d'une *Glue-Line*, c'est-à-dire l'interactivité, présente elle aussi des inconvénients majeurs:

1. Les données initiales subissent de nombreuses modifications. En effet, dans un premier temps la création d'une *Glue-Line* provoque la modification du maillage de sa *Surface-Mère*. Dans un second temps, la *fusion de deux Glue-Lines* par *collage* direct provoque elle-aussi des modifications importantes de la topologie de la *Surface-Mère*. Il est certain que, pour créer un lien physique entre deux surfaces, celles-ci ne peuvent conserver leur état initial. Cependant des modifications trop importantes et répétées conduisent à un modèle ne correspondant pas parfaitement aux données initiales.

2. Cette implantation du modèle travaille directement sur les surfaces initiales. Or chaque surface subit d'importantes modifications lors de la création des liens entre les objets. En conséquence, les surfaces initiales étant modifiées pour permettre la création d'un modèle donné, il ne peut être envisagé de créer un second modèle à partir de ces mêmes surfaces. Ceci conduirait en effet à de nouvelles modifications des surfaces, pour le second modèle, modifications inconsistantes avec le premier modèle.

Pour contourner ces inconvénients, une seconde méthode de construction du modèle, entièrement automatique, a été mise au point.

PARTIE III

CONSTRUCTION AUTOMATIQUE DU MODELE

Introduction

Les processus de création et d'association des *Glue-Lines* sont lourds et fastidieux pour l'utilisateur. Il nous fallait donc trouver un moyen de réduire l'interactivité de ce système, tout en conservant ses performances. L'objectif était de passer d'une méthode interactive de construction du modèle à une méthode automatique. C'est cette méthode qui vous est proposée dans cette troisième partie.

Le chapitre 8 est consacré plus spécifiquement au pré-traitement réalisé sur les données de départ, afin d'accélérer et de simplifier la construction du modèle.

Le chapitre 9 décrit toutes les étapes de la construction du modèle.

Chapitre 8

Préparation des données

8.1 Introduction

Dans la méthode de construction automatique du modèle volumique, la seule tâche de l'utilisateur est de fournir au système un ensemble de surfaces triangulées composant son modèle de base. Certaines de ces surfaces peuvent éventuellement être liées par l'intermédiaire de contraintes On-Tsurf (voir Paragraphe 8.2).
On suppose, avant d'entamer la construction du modèle, que les surfaces présentes sont:

- **"Embellies":**
  Cela correspond à une optimisation du maillage en fonction de divers "critères de beauté" [3]. Une attention toute particulière doit être portée sur le maillage du bord de chaque surface. En effet, comme nous l'avons dit tout au long de ce rapport, le bord d'une surface est un élément-clé de notre système de modélisation car c'est lui qui supporte les adjacences topologiques entre les *Faces*.

- **Interpolées:**
  Chaque surface du modèle doit être interpolée en utilisant l'interpolateur de GOCAD (D.S.I.). De cette façon, les surfaces sont lissées et les éventuelles contraintes On-Tsurf sont prises en compte.

8.2 Présentation de la contrainte On-Tsurf (OTS)

Dans ce paragraphe, nous proposons un bref résumé d'une contrainte D.S.I. existant dans GOCAD, la contrainte On-Tsurf (notée OTS). Cette dernière est présentée de façon beaucoup

plus précise et détaillée dans [22]. Dans le cadre de notre recherche, la contrainte OTS est utilisée comme un guide, une ligne directrice, dans la construction automatique du modèle volumique *non-manifold*.

8.2.1 Introduction

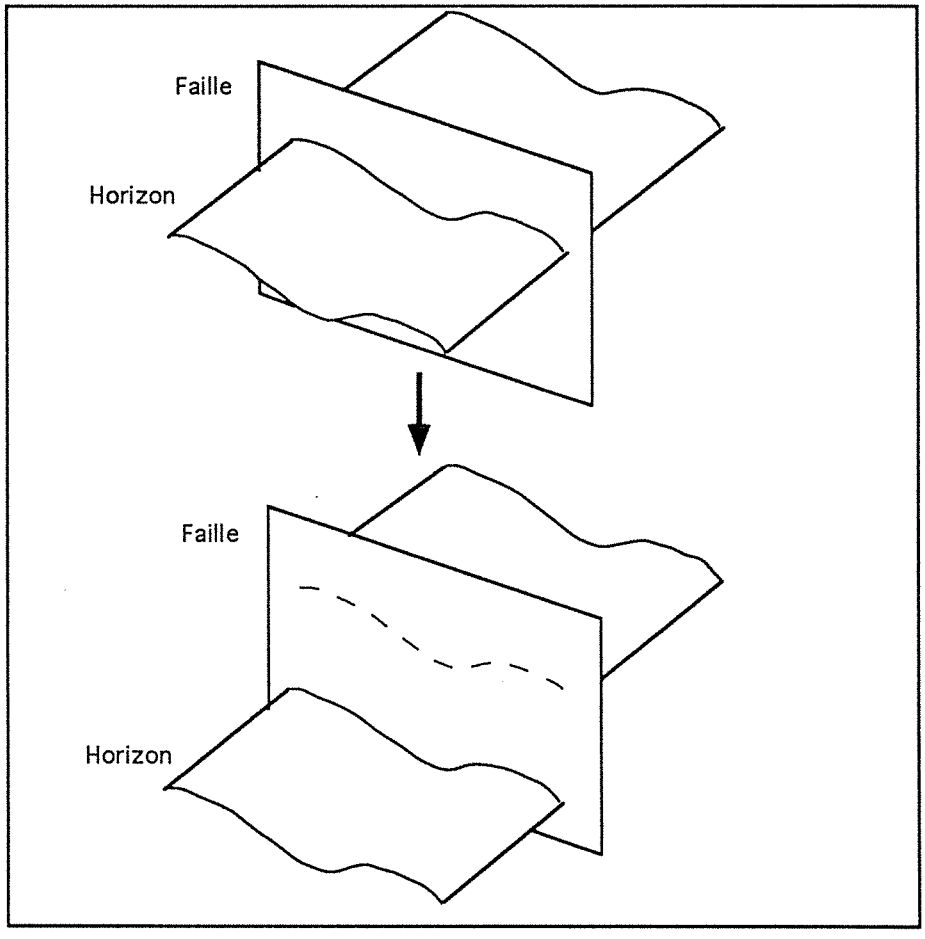


Figure 8.1 Glissement d'un horizon sur une faille.

L'idée de départ était de créer un contact, une soudure entre deux surfaces, permettant le glissement de l'une d'elles sur l'autre. Ce type de soudure est appelé soudure "soft", par opposition à la soudure "hard" réalisée par les procédés d'importation et d'exportation de vertices [27] . Cette soudure "soft" est mise en place à l'aide d'une contrainte D.S.I.: la contrainte **On-Tsurf**, notée OTS. Dans le domaine de la Géologie, cette soudure présente un intérêt particulier, car elle simule parfaitement le comportement d'un horizon traversé par une faille. Dans ce genre de situation (voir Figure 8.1), l'horizon doit glisser le long de la faille sans jamais s'en détacher.

8.2.2 Définition

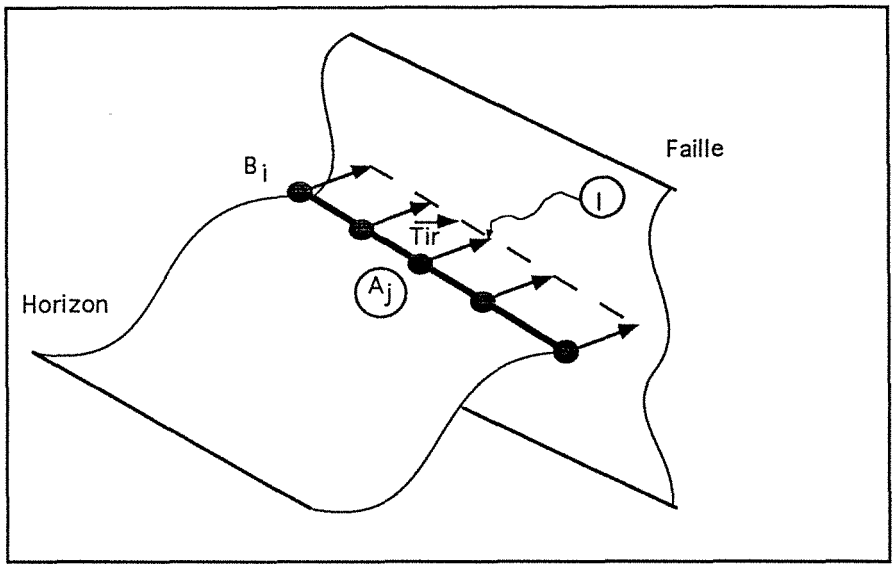


Figure 8.2 La contrainte On-Tsurf.

L'objectif est de créer une dépendance entre les deux objets *Horizon* et *Faille*, le long d'un bord B_i donné de l'*Horizon*. Cette dépendance est par la suite prise en compte par l'interpolateur D.S.I. pour restreindre les déplacements de B_i à une zone précise. En effet, la soudure "soft" consiste à contraindre le bord B_i de l'*Horizon* à se déplacer uniquement sur la *Faille*.

Le principe adopté (voir Figure 8.2) consiste à contraindre chaque atome A_j du bord B_i de l'*Horizon*. Ainsi, pour chacun de ces atomes, un vecteur de tir $\overrightarrow{Tir}$ et une cible sont définies; la cible correspond à la *Faille*. L'intersection entre la droite de vecteur directeur $\overrightarrow{Tir}$ et la *Faille* donne un point I. Ce point constitue la position que l'atome A_j doit occuper pour établir le contact entre l'*Horizon* et la *Faille*. Les vecteurs de tir sont définis dans un premier temps de façon automatique, en minimisant la distance entre l'origine du vecteur (ie un atome du bord B_i) et la *Faille*. L'utilisateur a ensuite la possibilité de modifier chacun des vecteurs de façon interactive, provoquant l'interpolation des autres vecteurs de tir (à l'aide de l'algorithme D.S.I.) et donc le recalcul des points d'intersection entre chaque ligne de tir et la *Faille*.

Lors de l'interpolation de l'*Horizon*, les points du bord B_i sont déplacés vers la faille, pour occuper les positions définies par les contraintes OTS (voir Figures ?? et ??).

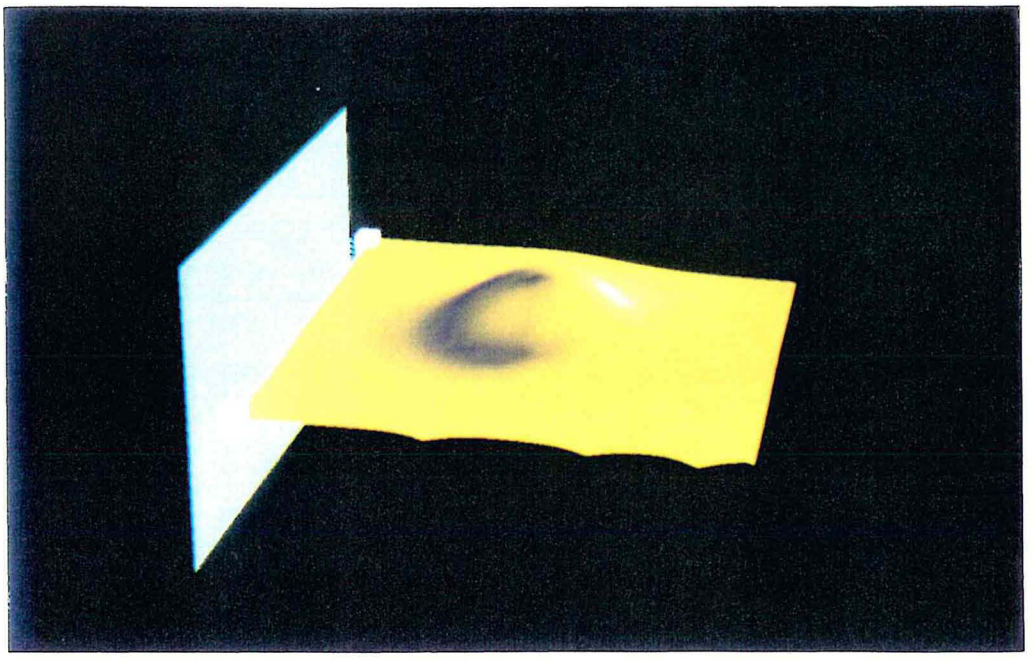


Figure 8.3 Définition de contraintes OTS sur un bord de l'horizon, en direction de la faille.

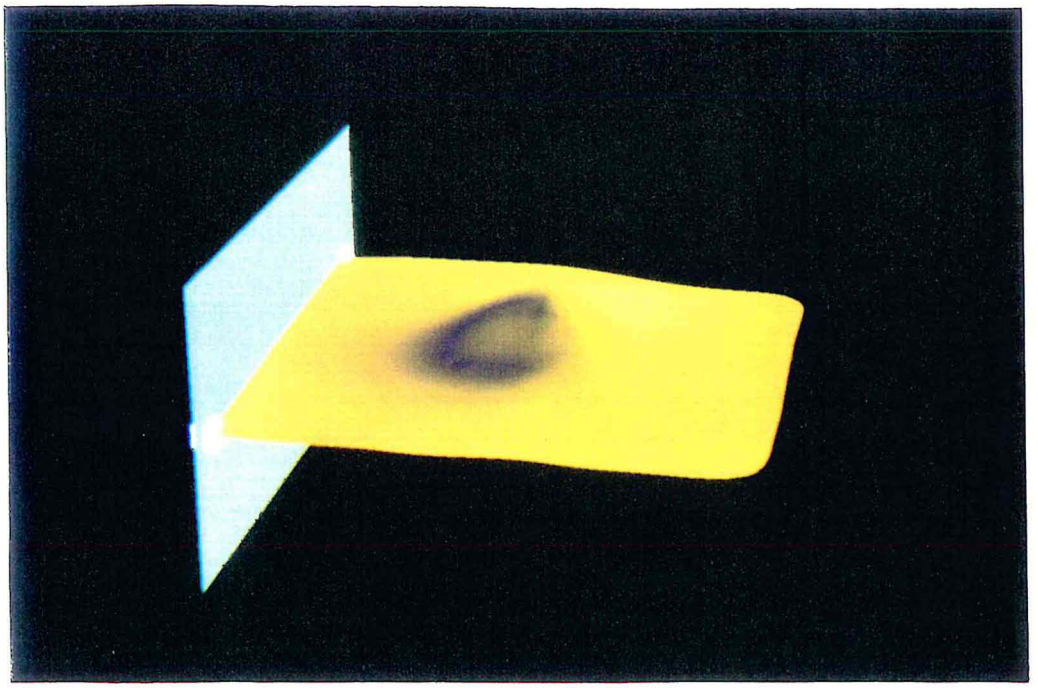


Figure 8.4 Prise en compte des contraintes OTS de l'horizon vers la faille, après interpolation de l'horizon. L'horizon est collé à la faille, et le restera suite à toute interpolation.

8.3 Préparation des données

8.3.1 Introduction

Quand l'utilisateur considère que les données dont il dispose sont prêtes pour l'étape suivante, il ne lui reste plus qu'à lancer le procédé de construction automatique du modèle. Lorsque ce dernier sera terminé, le modèle volumique sera entièrement construit et prêt à être utilisé à l'aide des outils proposés (voir Annexes B et C).

Le principe de cette méthode de construction automatique repose sur l'utilisation des intersections entre les différentes surfaces en présence. En effet, une ligne d'intersection entre deux surfaces peut être considérée comme une "ligne de colle" permettant de souder ces deux surfaces. On retrouve donc dans cette méthode l'idée qui nous avait conduit à créer les *Glue-Lines*, à savoir la **création de liens physiques entre les surfaces**. A la différence de la méthode interactive, où ces liens devaient être construits "manuellement" à l'aide des *Glue-Lines*, la méthode automatique utilise chaque ligne d'intersection entre deux surfaces pour créer un lien réel entre elles.

La construction automatique d'un modèle volumique nécessite un pré-traitement des données qui comporte les étapes suivantes:

1. Duplication des surfaces.
2. Calcul des intersections.

8.3.2 Duplication des surfaces

Cette étape a été introduite afin d'éviter toute altération des surfaces originales. De cette façon, si le modèle solide final ne correspond pas à l'interprétation de l'utilisateur, ce dernier peut revenir aux données initiales, les modifier, puis reconstruire un nouveau modèle solide. En d'autres termes, cette duplication des surfaces initiales permet de "geler" toute modification topologique de ces mêmes surfaces. L'inconvénient qui existait dans la méthode interactive, à savoir la possibilité de ne créer qu'un seul modèle à la fois, est supprimé grâce à cette opération. La duplication d'une surface consiste à:

- dupliquer la géométrie et la topologie de la surface (vertices, atomes et triangles). Suite à cette opération, l'unique chose que la surface originale partage avec sa copie est la position spatiale de chacun des points qui la compose.
- dupliquer les structures définissant le bord de la surface, en particulier les Border-Stones servant à décomposer un bord donné en plusieurs morceaux indépendants (voir Introduction, Paragraphe 0.2.2).
- dupliquer les contraintes OTS, partout où elles existent.

Pour éviter d'alourdir la présentation des développements qui vont suivre, le terme "surface" fera référence à la copie de la surface originale.

8.3.3 Calcul des intersections

Cette opération consiste à calculer toutes les intersections existantes entre les surfaces données. C'est à ce stade qu'intervient la contrainte **On-Tsurf** présentée dans le paragraphe 8.2. On a précisé que cette contrainte permettait de "souder" deux surfaces. Qui dit soudure, dit relation d'adjacence en ce qui concerne notre modèle. En conséquence, il nous faut utiliser ces contraintes dans la construction du modèle.

Par définition, la contrainte OTS permet de coller un bord de surface sur une autre. Cependant, la précision numérique n'est pas toujours suffisante pour que l'intersection entre ces deux surfaces en relation corresponde exactement à la "ligne de soudure". En conséquence, nous avons mis au point un traitement spécial des contraintes OTS, visant à accentuer l'intersection entre deux surfaces qui sont en relation par l'intermédiaire de contraintes OTS.

Pré-traitement

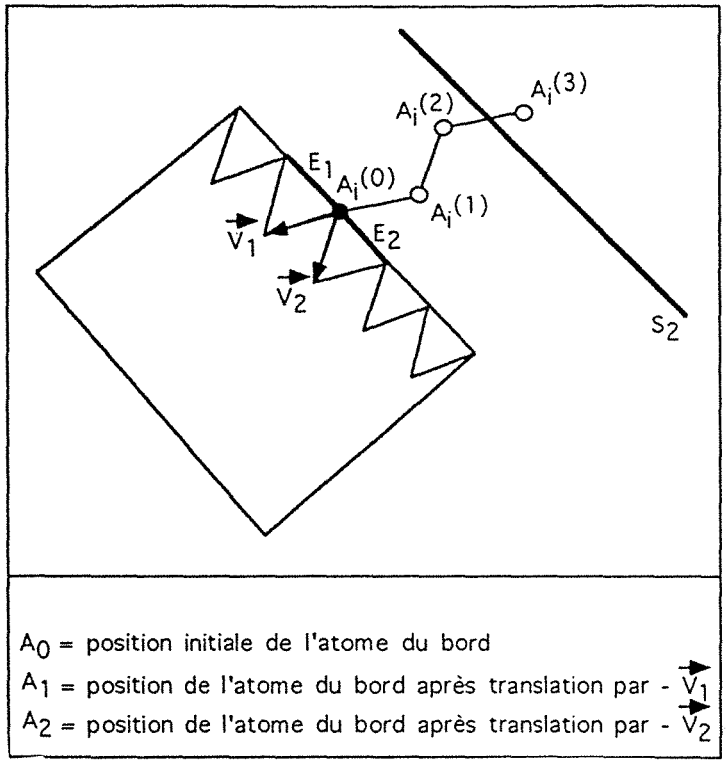


Figure 8.3 Pré-traitement de l'opération Cut.

L'idée sur laquelle repose tout ce traitement est d'obliger tous les triangles du bord d'une surface S_1 ayant des contraintes OTS vers une autre surface S_2 à intersecter la surface S_2. Pour cela, chaque point A_i du bord de S_1 ayant une contrainte OTS vers une autre surface S_2 doit être

"déplacé" vers la surface S_2 de façon à ce que A_i passe de "l'autre côté" de la surface S_2. Le déplacement du point A_i s'effectue par des translations successives jusqu'au respect de deux conditions données. Cette opération s'effectue pour chaque point du modèle ayant une contrainte OTS.

La méthode de transformation des contraintes OTS est présentée sur la figure 8.3. Ce procédé est exécuté pour chaque surface du modèle et se divise en plusieurs phases:

1. Mémoriser dans un ensemble tous les atomes du bord de la surface ayant des contraintes OTS.

2. Calculer des vecteurs de translation $\overline{V_i}, i \in [1..n]$ pour chaque atome précédemment sélectionné.
   A chaque atome du bord possédant une contrainte OTS sont associés des vecteurs de translation qui permettront de déplacer cet atome. Ces vecteurs correspondent aux edges internes de la surface courante, dont l'atome est une extrémité. En d'autres termes, chaque edge de triangle de la surface tel que

   - l'atome courant est une extrémité de cet edge,
   - cet edge n'est pas un edge du bord,

   constitue un vecteur de translation pour l'atome courant.

3. Déplacer chaque atome sélectionné en fonction des vecteurs de translation.
   Examinons par exemple le traitement réalisé sur l'atome A_i.

   - Soit $A_i^{(0)}$ l'atome de départ.
   - Soit $A_i^{(1)}$ le point obtenu par translation de A_i par le vecteur V_1.
   - Soit $A_i^{(2)}$ le point obtenu par translation de $A_i^{(1)}$ par le vecteur V_2.
   - ...
   - Soit $A_i^{(n)}$ le point obtenu par translation de $A_i^{(n-1)}$ par le vecteur V_n.

   Chaque opération de translation est suivie d'un test destiné à savoir si le point calculé est passé de "l'autre côté" de la surface-destination. Dans ce cas, aucune translation supplémentaire n'est nécessaire pour ce point. Formellement, les conditions d'arrêt des translations pour un atome donné A_i s'expriment de la façon suivante:

   - Soient $A_i^{(n-1)}$ l'atome de départ et $A_i^{(n)}$ le point calculé[1].
     Soit S_2 la surface-destination de la contrainte OTS de l'atome A_i.
     si (
     $$((A_i^{(n-1)}, A_i^{(n)}) \cap S_2 \neq \emptyset)$$
)
     la première condition est respectée.

[1] n indique que l'opération de translation qui vient d'être réalisée est la $n^{ème}$.

- Soient E_1 et E_2 les deux segments du bord de la surface ayant l'atome A_i comme extrémité commune.
  Soit S_2 la surface-destination de la contrainte OTS de l'atome A_i.
  si (

$$(E_1 \cap S_2 = \emptyset) et$$
$$(E_2 \cap S_2 = \emptyset)$$

)
  la seconde condition est respectée.

Si l'on considère notre hypothèse de départ, à savoir que les surfaces initiales doivent être interpolées, on en déduit que les contraintes OTS ont été prises en compte avant cette opération. En conséquence, les bords de surface avec des contraintes OTS sont déjà, avec plus ou moins de précision, en contact physiquement avec leurs surfaces-destination. Donc les opérations de translation que nous effectuons dans ce pré-traitement au calcul des intersections sont quasi-immédiates (un déplacement infinitésimal suffit pour respecter les conditions). Le résultat de ce pré-traitement sur un exemple simple est présenté sur la figure 8.4.

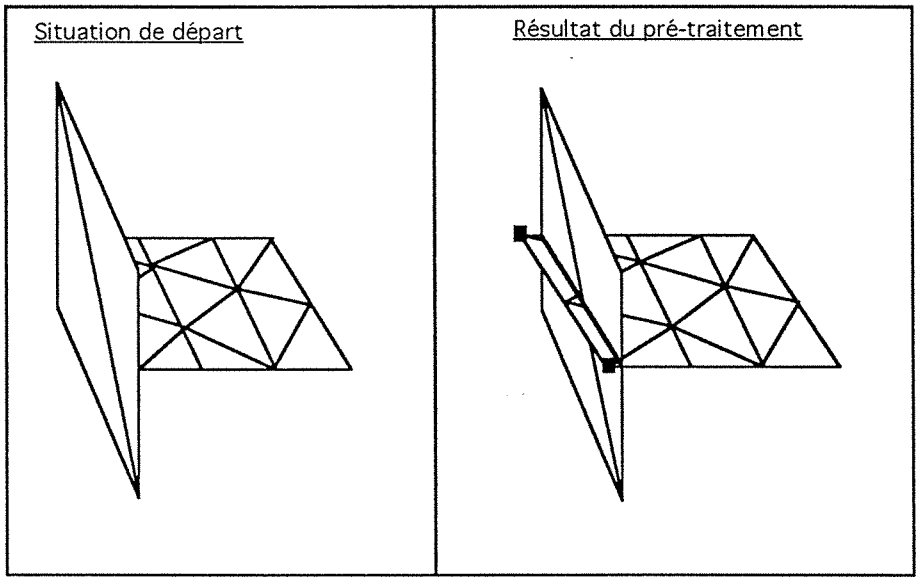


Figure 8.4 Résultat du pré-traitement de l'opération Cut.

Calcul des intersections: Opération Cut

L'ensemble des copies des surfaces initiales est fournie à la fonction réalisant l'opération de calcul des intersections: TSURF_List_Cut() [27]. Cette dernière calcule l'intersection entre chaque surface de cet ensemble et toutes les autres puis coupe effectivement les surfaces en les retriangulant localement le long de la ligne d'intersection. Cette opération produit donc de nouveaux triangles et de nouveaux bords de surfaces (voir Figure 8.5).

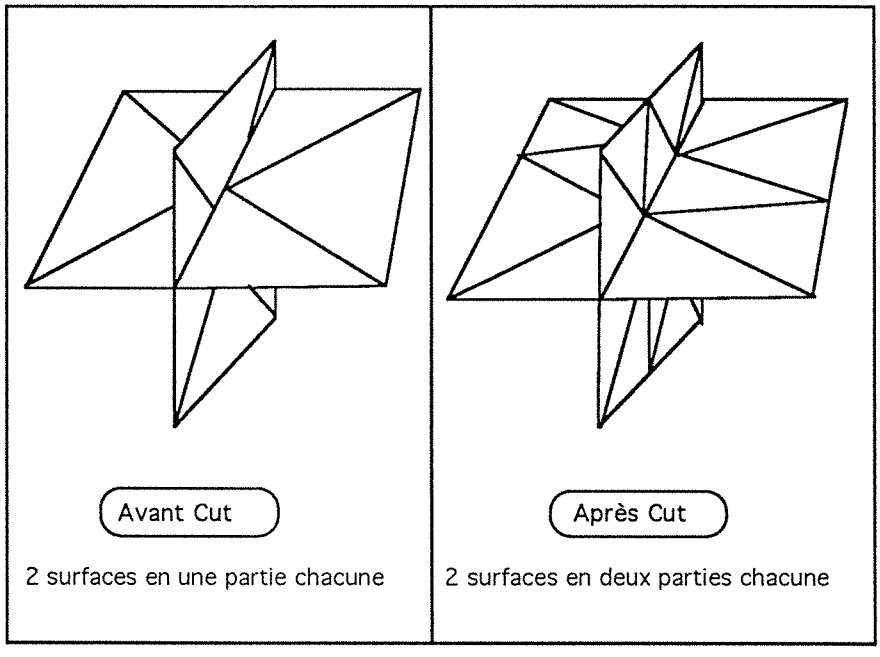


Figure 8.5 L'opération Cut.

8.3.4 Post-traitement

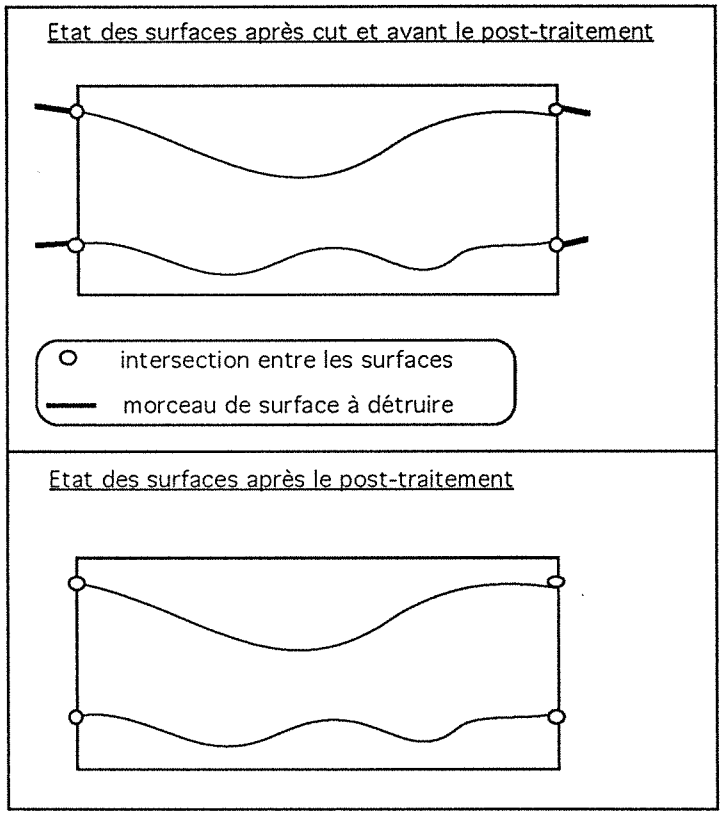


Figure 8.6 Post-traitement de l'opération Cut (Représentation 2D).

A ce stade de la modélisation, il faut détruire les triangles résultant du pré-traitement de l'opération Cut (voir Figure 8.6). Ce post-traitement peut être vu comme un "ébarbage" des surfaces, de façon à obtenir un résultat propre dans lequel les bords de surfaces en contact correspondent parfaitement (mêmes sommets de triangles).

8.4 Conclusion

L'objectif de ce traitement est de préparer automatiquement les données, c'est-à-dire les surfaces initiales, de façon à ce que la création automatique du modèle soit la plus simple possible. En cela, la contrainte OTS peut se révéler très utile pour créer des relations entre deux surfaces initiales qui n'en avaient pas. Avant d'entamer la construction du modèle, le système possède un ensemble de triangles pour lesquels les adjacences sont parfaitement déterminées. En d'autres termes, l'adjacence de deux triangles déconnectés peut être aisément définie en comparant les coordonnées des extrémités de leurs edges respectifs. La création des structures décrivant les adjacences topologiques entre ces triangles constitue l'étape suivante du processus de création automatique du modèle.

Chapitre 9

Construction du modèle

9.1 Introduction

L' objectif du pré-traitement présenté dans le chapitre précédent est la création de l'ensemble des triangles qui vont être utilisés pour la création du modèle solide. A partir de ce stade, où nous entrons à proprement parlé dans la phase de création du modèle volumique à base topologique, nous ne parlerons plus de surfaces, mais de modèle, *Face*, *Shell*, *Région*, *Layer* et *Tlink*. Les structures du modèle sont créées lors de plusieurs étapes:

1. Création des *Faces*.
2. Création des structures radiales.
3. Création des *Tlinks*.
4. Création des *Shells*, *Régions* et *Layers*.

9.2 Création des Faces

Une *Face* est un ensemble de triangles connectés. Elle correspond à un sous-ensemble connexe de la surface initiale. Ce découpage d'une surface en *Faces* peut provenir soit de l'état initial de la surface (plusieurs morceaux déconnectés), soit de l'opération précédente de Cut.

Pour un ensemble de surfaces donné, le procédé de création des *Faces* (voir Figure 9.1) consiste à définir de façon récursive l'ensemble des triangles adjacents à un triangle donné. Ces triangles constituent un sous-ensemble de l'ensemble global des triangles du modèle. Ce sous-ensemble est ensuite déduit de l'ensemble de départ. Puis un nouveau triangle est choisi,

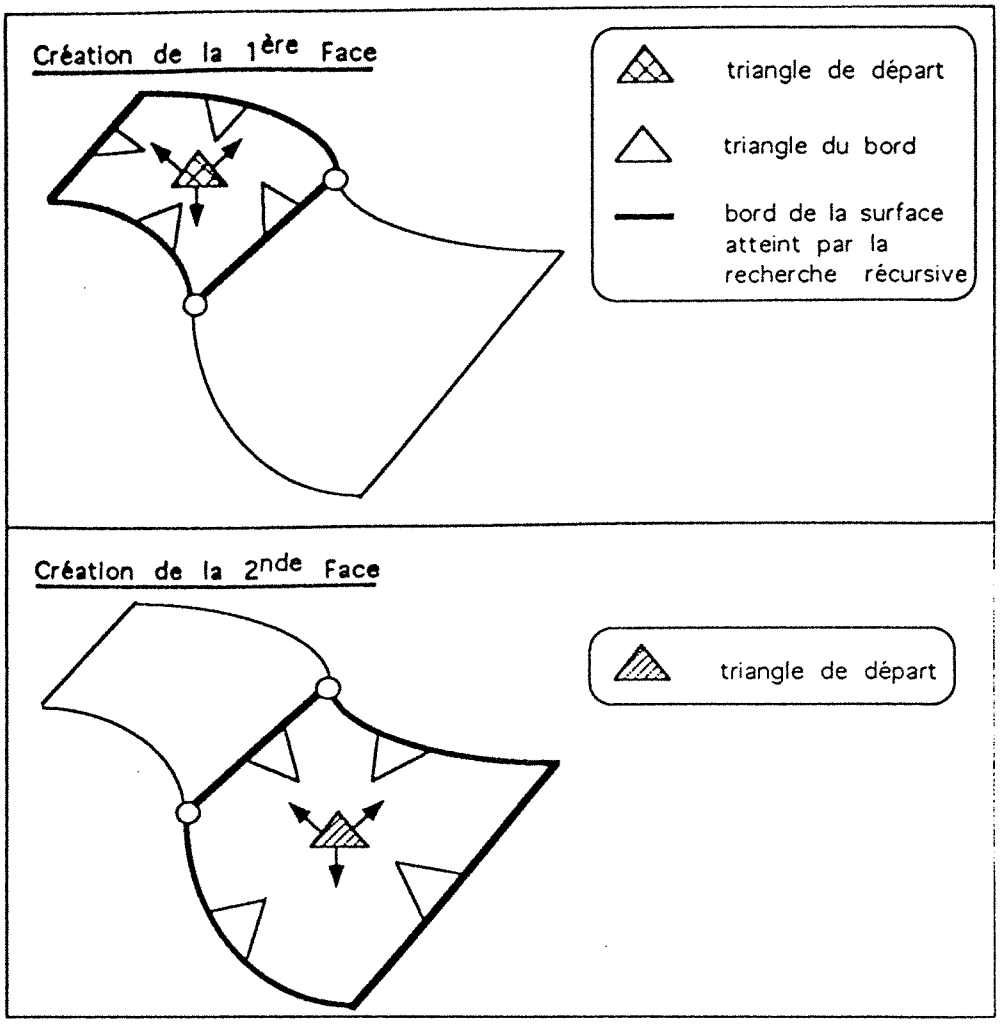


Figure 9.1 Création des *Faces*.

et l'opération de construction récursive est répétée. Ces opérations s'exécutent jusqu'à ce que l'ensemble des triangles de départ soit vide.

void Create_FACES (M , *trgl_Set*)

- **Input:**
  - *trgl_Set* = ensemble de tous les triangles du modèle.
  - Modèle M.
- **Output:** Modèle M.

```
{
  /* Initialisation */
  Créer une pile vide Stack ;

  tant que ( trgl_Set ≠ ∅ )
    {
      • trgl = un triangle quelconque de trgl_Set ;
      • Créer une Face vide new_Face ;
      • Empiler ( Stack , trgl ) ;

      /* Recherche récursive des triangles adjacents au triangle donné, pour définir une
         nouvelle Face. */
      tant que ( Stack ≠ ∅ )
        {
          • Dépiler ( Stack , trgl ) ;
          pour ( i=0; i < 3; i++ )
            {
              • adj_Trgl = i^ème triangle adjacent de trgl ;
              si ( adj_Trgl == NULL )
                continue ;

              • Ajouter ( new_Face , adj_Trgl ) ;
              • Supprimer ( trgl_Set , adj_Trgl ) ;
              • Empiler ( Stack , adj_Trgl ) ;
            }
        }

      /* Ajouter la nouvelle Face dans le modèle. */
      • Ajouter ( M , new_Face ) ;
    }
}
```

9.3 Création des structures radiales

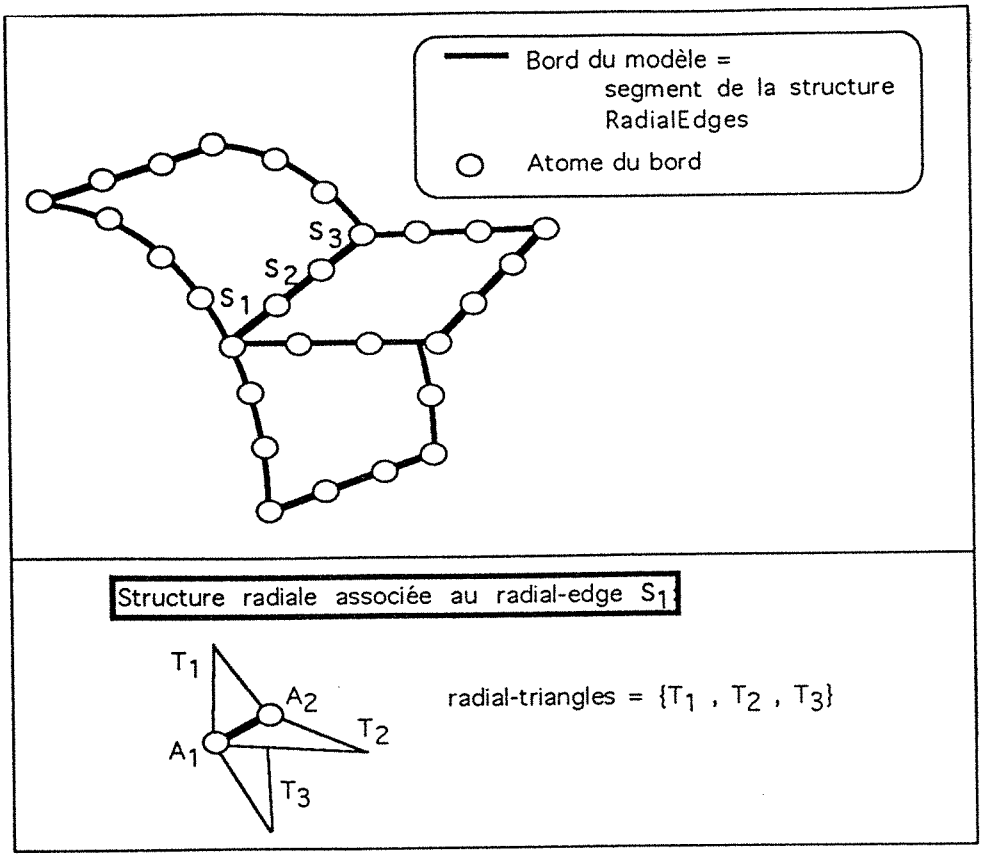


Figure 9.2 Création des structures radiales.

Cette étape consiste à créer une **structure globale décrivant le bord du modèle.** En d'autres termes, on a choisi de grouper tous les segments définissant un bord de *Face* dans une seule et même structure (voir Figure 9.2). Cette étape a été introduite pour simplifier et accélérer la phase suivante, à savoir la création des *Tlink*s du modèle.

La structure décrivant le bord du modèle, appelée **Radial-Edges**, est définie par des segments et des points. Chaque segment (ou **radial-edge**) correspond à un edge de triangle sur le bord d'une *Face*. En conséquence, les points définissant cette structure sont tous sur le bord d'une *Face*. A chaque *radial-edge* sont associés les triangles qui lui sont adjacents, c'est-à-dire les triangles dont un edge correspond à ce *radial-edge*. Pour un *radial-edge* donné, chacun des triangles qui lui sont associés s'appelle **radial-triangle**. Pour deux points consécutifs sur le bord, il existe un nombre $n \geq 1$ de triangles dont un edge est défini par ces points. Chacun de ces edges est un edge du bord provenant soit du bord initial de la surface, soit de l'opération de Cut.

Par exemple, dans le cas présenté par la figure 9.2, les points A_1 et A_2 définissent un edge

partagé par trois triangles (T_1, T_2 et T_3). Notre objectif est de construire un unique edge défini par ces deux points consécutifs, et de lui associer les n triangles qui lui sont adjacents:

- Soit A_1 et A_2 deux points du bord.
- Soient T_1, T_2, ..., T_n les n triangles ayant un edge défini par A_1 et A_2.
- Traitement du triangle T_1:
  Un *radial-edge* est créé, défini par les deux points A_1 et A_2.
  T_1 est ajouté à l'ensemble de triangles attaché à ce nouveau *radial-edge*.
- Traitement des autres triangles:
  Chaque triangle est ajouté à l'ensemble de triangles attaché au *radial-edge* dont les deux points extrémité correspondent géométriquement aux deux points définissant l'un des edges du triangle.

Le procédé de création de la structure *Radial-Edges* est le suivant:

Notations

- *Radial_Edge* est la structure rattachée au modèle décrivant le bord de celui-ci.
- *radial_edge* est un segment du bord du modèle défini par deux atomes-extrémité et les triangles qui lui sont adjacents.
- *radial_triangle* est un triangle adjacent à un edge du bord du modèle (*radial-edge*).

void Create_Radial_Edges (M)

- **Input:** Modèle M.
- **Output:** Modèle M.

```
{
  /* Initialisation */
  Créer un ensemble vide Radial_Edges destiné à stocker tous les radial-edges du modèle ;

  pour ( chaque Face F_i ∈ M )
    {
      pour ( chaque triangle T_j ∈ F_i )
        {
          pour ( k=0; k < 3; k++ )
            {
              adj_Trgl = k^ème triangle adjacent de T_j ;

              si ( adj_Trgl ≠ NULL )
                continue ;

              /*
                 Recherche du radial-edge associé au i^ème côté du triangle:
                 Les atomes de ce côté correspondent aux atomes numérotés
                 (k + 1)%3 et (k + 2)%3, d'après [3].
              */

              • A_1 = adj_Trgl → atom[(k + 1)%3] ;
              • A_2 = adj_Trgl → atom[(k + 2)%3] ;
              • radial_edge=Create_or_Retrieve_Radedge (Radial_Edges, A_1, A_2) ;
              • radial_triangle = adj_Trgl ;
              • Ajouter ( radial_edge , radial_triangle ) ;
            }
        }
    }
}
```

RADIAL_EDGE_t **Create_or_Retrieve_Radial_Edge** (*Radial_Edges* , A_1 , A_2)

- **Input:**
  - *Radial_Edges* = ensemble de tous les *radial-edges* du modèle.
  - A_1 et A_2 sont les deux atomes-extrémité de l'edge du triangle.
- **Output:** *radedge* = *radial-edge* partageant les atomes A_1 et A_2.

```
{
  /*
     Recherche d'un radial-edge de mêmes coordonnées:
     Cette opération est réalisée à l'aide de la fonction
     Same_Geometry_as_Radial_Edge () qui, pour un radial-
     edge et deux atomes donnés retourne VRAI si les extrémités
     du radial-edge ont les mêmes positions (x,y,z) que les deux
     atomes donnés et FAUX sinon.
  */

  pour ( chaque radedge ∈ Radial_Edges )
     {
       si ( Same_Geometry_as_Radial_Edge ( radedge , A1 , A2 ) )

          break ;
     }

  si ( radedge = NULL )
     {
       /* Création d'un nouveau radial-edge. */

       • radedge.atome1 = A1 ;

       • radedge.atome2 = A2 ;

       • Ajouter ( Radial_Edges , radedge ) ;
     }
  return ( radedge ) ;
}
```

9.4 Création des Tlinks

A ce stade, le système connait:

- les *Faces* du modèle,
- pour chaque edge du bord, les triangles qui lui sont géométriquement associés.

Pour construire les adjacences topologiques entre les triangles partageant un même edge, c'est-à-dire les *Tlinks*, il est nécessaire de connaitre pour chaque triangle les deux triangles qui l'encadrent. C'est l'objectif de la première étape de la création des *Tlinks*: le tri des *radial-triangles*.

9.4.1 Tri des radial-triangles

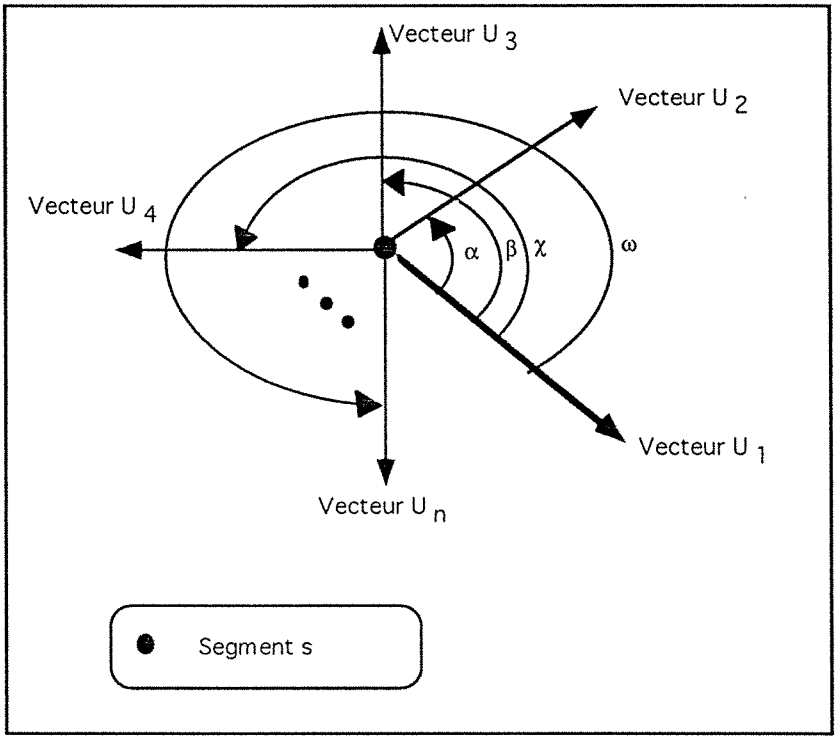


Figure 9.3 Tri des *radial-triangles* (Représentation 2D).

On dispose d'un *radial-edge* et des n *radial-triangles* qui lui sont associés. Notre objectif est de trier ces *radial-triangles* en fonction de l'angle algébrique que chacun d'eux forme avec un triangle de référence (voir Figure 9.3). Dans un premier temps, un vecteur unitaire U_i, $i \in [1..n]$ est calculé pour chaque *radial-triangle*. La méthode de calcul d'un vecteur unitaire d'un triangle

donné a été exposée dans le chapitre 7, paragraphe 7.3.2. Le vecteur U_1 est arbitrairement choisi comme vecteur de référence. Ensuite, l'angle algébrique entre le vecteur U_1 et chaque vecteur U_i, $i \in [2..n]$ est calculé.
Enfin, les vecteurs U_i sont triés par ordre croissant de leurs angles avec le vecteur de référence U_1. Les triangles associés à ces angles sont rangés dans une double liste circulaire.

9.4.2 Création des structures Tlink

La double liste circulaire offre la possibilité d'accéder, à partir d'un élément, à son précédent et à son suivant. De cette façon, à partir d'un triangle T de la liste, on peut obtenir directement les deux triangles qui l'entourent. Ces triangles T_1 et T_2 correspondent aux deux triangles adjacents dont nous avons besoin pour créer le *Tlink* du triangle T. De plus, pour la création du *Tlink* associé au triangle T, il est nécessaire de connaitre les orientations des deux triangles adjacents à T. Ceci est obtenu en testant le signe du produit scalaire de la normale au triangle adjacent et du vecteur unitaire du triangle T. On obtient donc les adjacences suivantes pour le triangle T:

- $[T,'+']$ est lié à $[T_1, side_1]$.
- $[T,'-']$ est lié à $[T_2, side_2]$.

D'où le *Tlink* associé au triangle T, à l'edge E correspondant au *radial-edge*:

$$Tlink(T,E) = [T_1, side_1], [T_2, side_2].$$

9.5 Création des Shells, Régions et Layers

A ce stade, toutes les structures utiles à la création des *Régions* sont disponibles. L'ultime opération pour achever la création du modèle solide consiste à créer la base de données des *Régions*. Pour cela, chaque *Face* du modèle est examinée à l'aide de l'algorithme de détection de *Régions* présenté dans la première partie de cet ouvrage (voir fonction **Create_Regions()** dans Chapitre 4, Paragraphe 4.2.4).

Comme nous l'avons précisé lors de la définition du *Layer*, celui-ci dépend exclusivement de l'interprétation de l'utilisateur. Nous avons choisi d'initialiser le processus de regroupement des *Régions* en *Layers* en créant un *Layer* pour chaque nouvelle *Région* détectée.

9.6 Optimisation du maillage avant calcul du modèle

Les traitements effectués lors de la préparation des données pour la construction du modèle (voir Chapitre 8) , et en particulier l'opération de "découpe" des surfaces entre-elles, produisent souvent des triangles ne respectant pas les "critères de beauté" [3]. On a donc imaginé d'introduire un embellisseur de triangles, juste avant l'étape de création des *Tlinks*. Cette opération consiste

à améliorer l'aspect des *radial-triangles* en respectant la structure *Radial-Edges*. La retriangulation s'effectue donc pour chaque *radial-triangle* d'un *radial-edge* donné et consiste à fusionner certains *radial-edges* si les *radial-triangles* qui lui sont attachés ne respectent pas les critères de beauté. Le problème de cette méthode réside dans le fait que l'embellissage d'un *radial-edge* peut conduire à un désembellissage d'un autre *radial-edge*. C'est pourquoi les critères de beauté ne sont respectés qu'à un epsilon près.

9.7 Exemple de modèle volumique

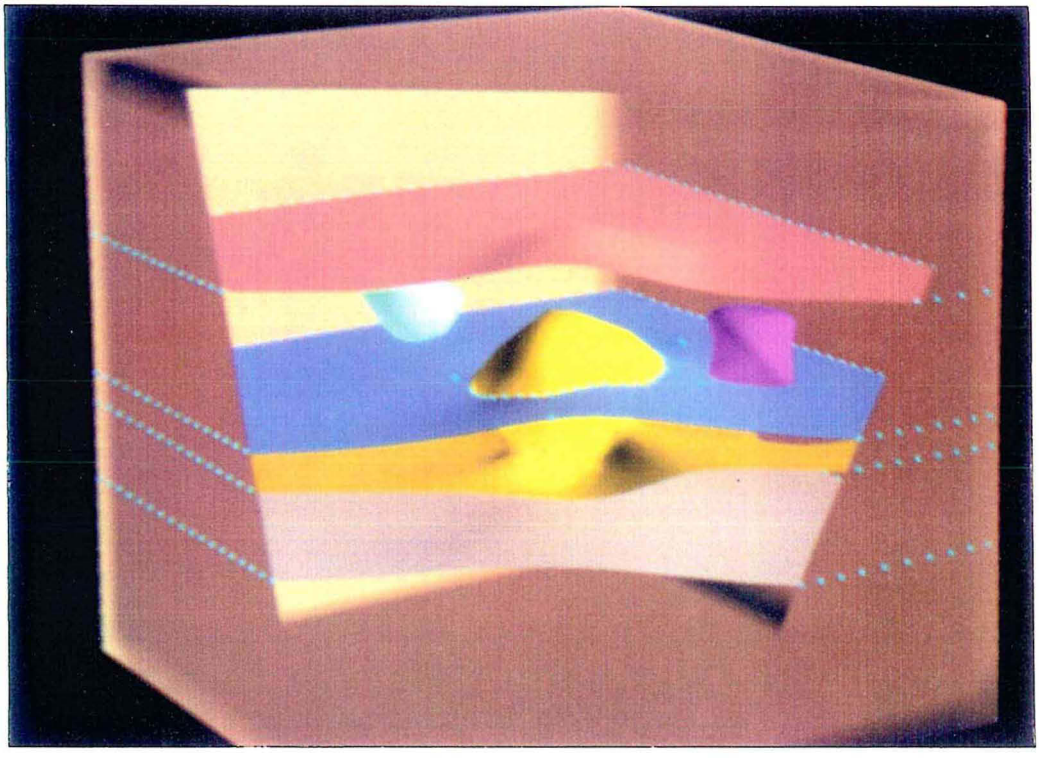


Figure 9.4 Modèle Surfacique utilisé pour la construction automatique du modèle volumique. On distingue quatre horizons et deux lentilles, le tout compris dans une boite limitant l'espace de modélisation (le plan frontal de la caméra a été avancé pour permettre de visualiser l'intérieur de la boite). Les horizons sont collés à la boite à l'aide de contraintes OTS.

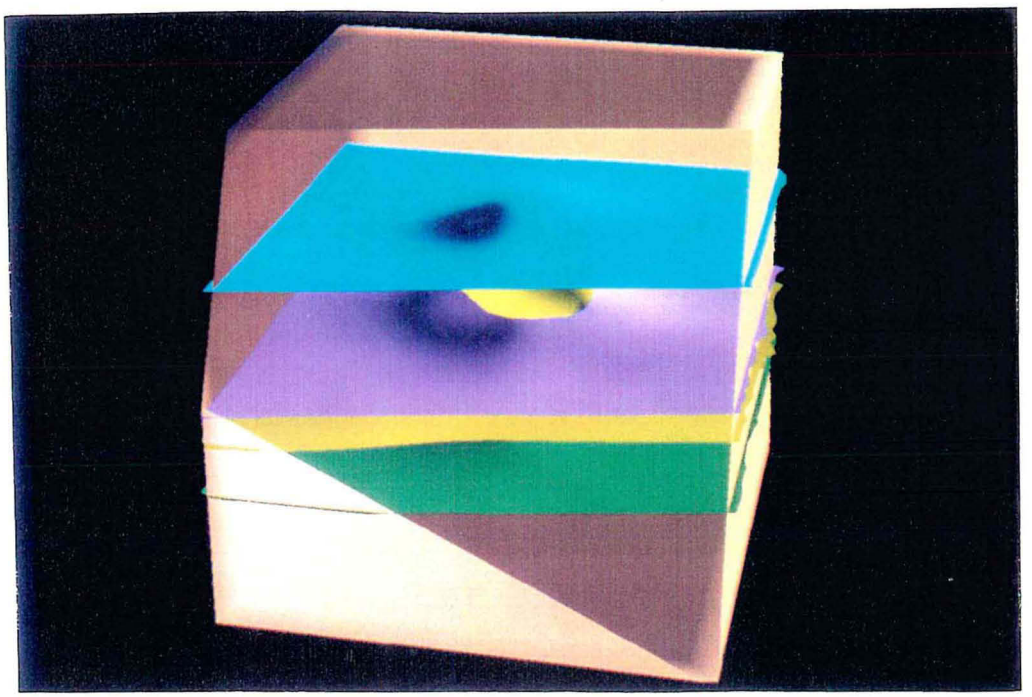


Figure 9.5 Etat des données après le pré-traitement de l'opération Cut. On distingue nettement que les horizons ont été étirés vers la boite.

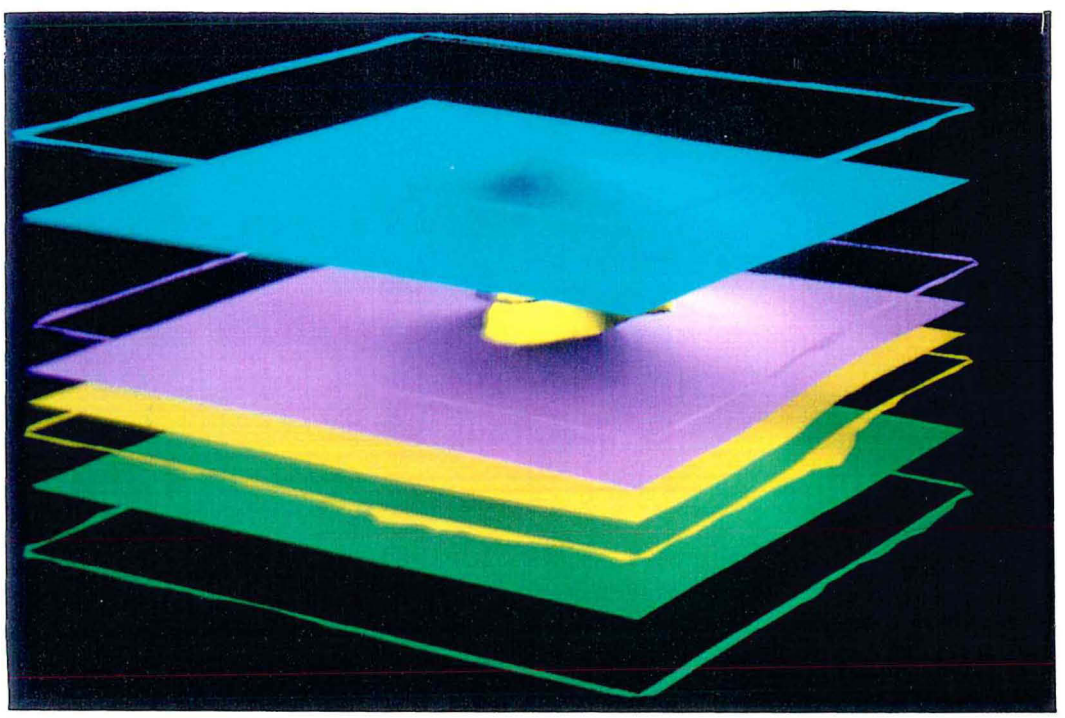


Figure 9.6 Etat des données aprés l'opération Cut. Les morceaux de surfaces extérieurs à la boite ont été déplacés pour permettre une meilleure visualisation du résultat.

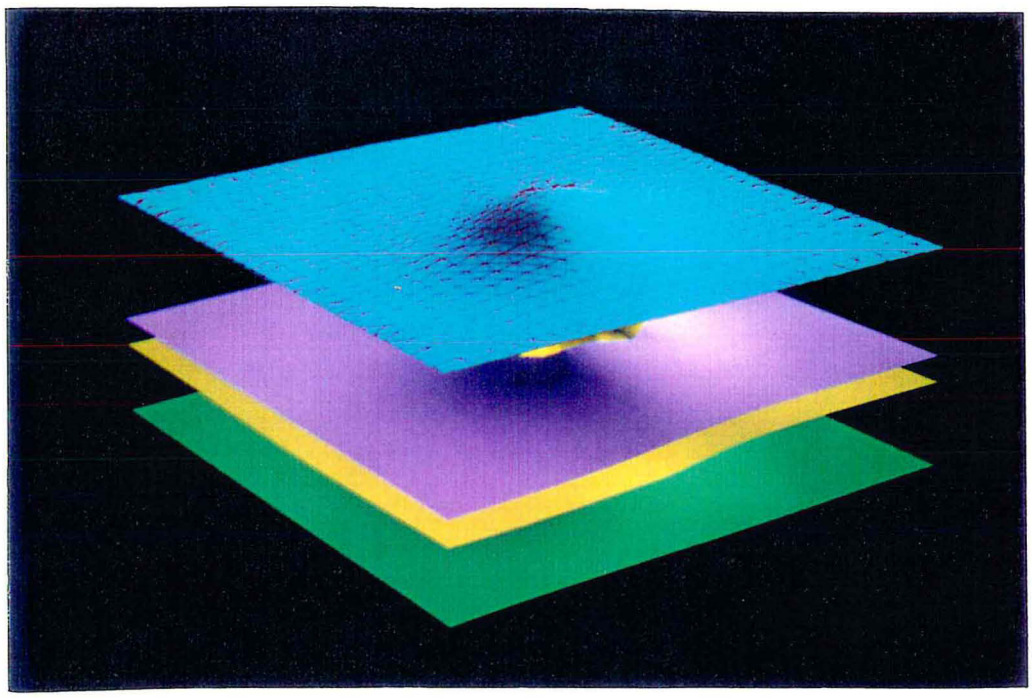


Figure 9.7 Etat des données après le post-traitement de l'opération Cut. Les morceaux de surfaces extérieurs à la boite ont été supprimés.

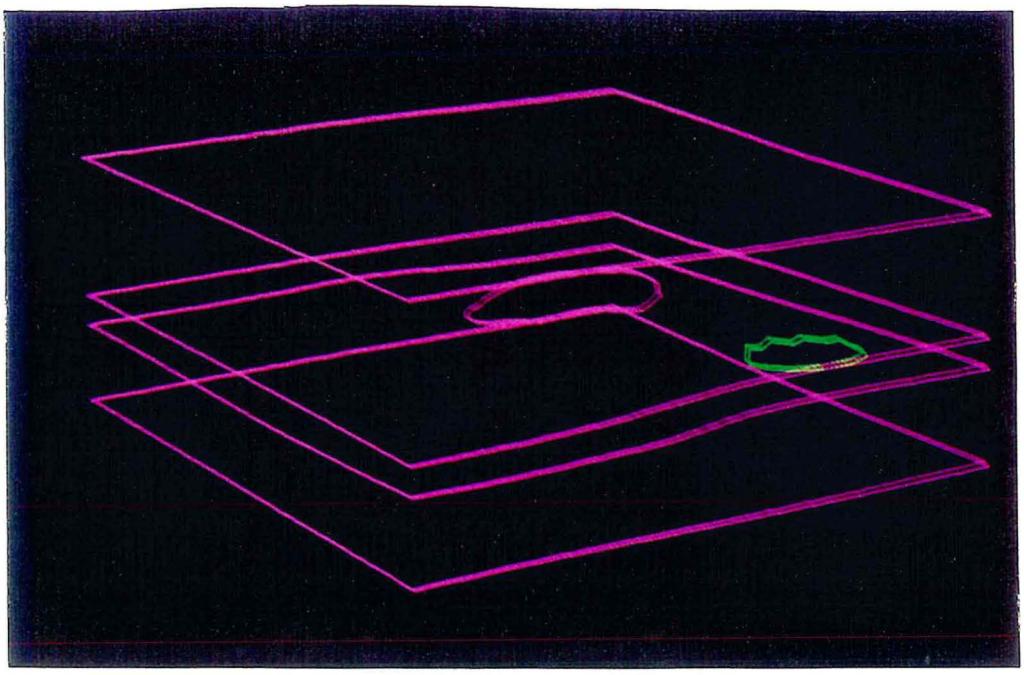


Figure 9.8 Représentation Fil de Fer du modèle volumique. Les lignes roses correspondent à des edges auxquels sont associés plus d'un triangle. Les lignes vertes représentent les edges libres du modèle.

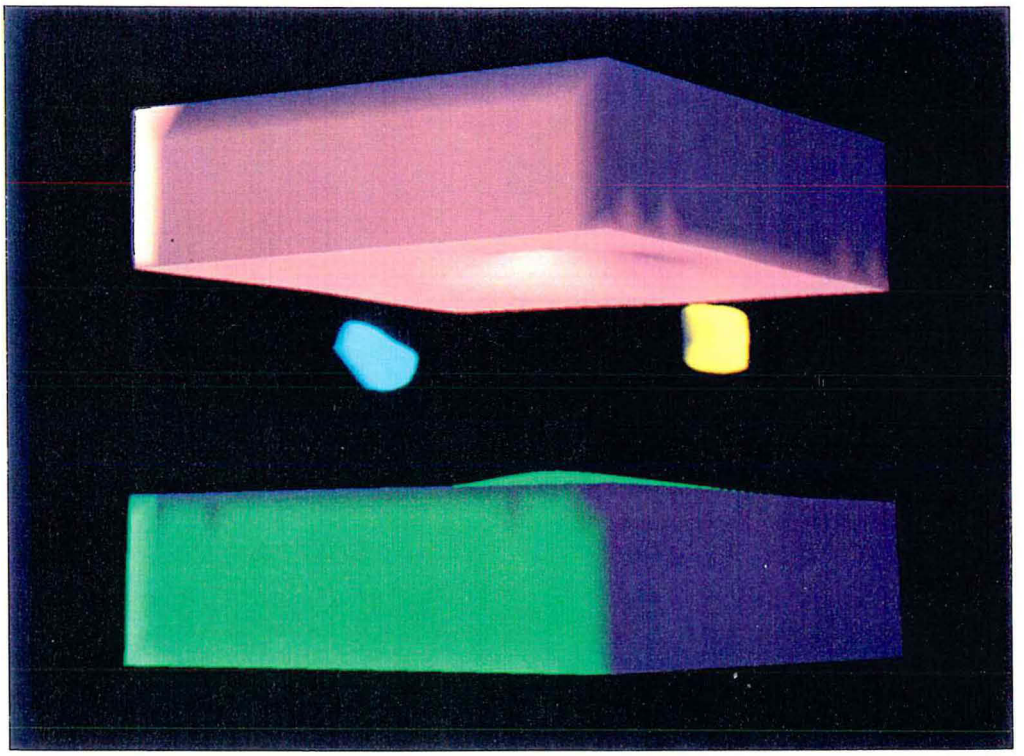


Figure 9.9 Représentation des *Régions* du modèle volumique. On distingue la couche inférieure et la couche supérieure du modèle, ainsi que la *Région* correspondant à l'intérieur de chaque lentille.

9.8 Conclusion

La construction automatique du modèle volumique se révèle très rapide, grâce au traitement de préparation des données présenté dans le chapitre précédent. A partir de ce moment, l'utilisateur dispose d'une série d'opérations lui permettant de manipuler le modèle obtenu (voir Annexes B et C). Manipuler, mais non pas modifier. En effet, le modèle construit par cette méthode automatique ne peut être modifié. La solution pour l'utilisateur est de revenir aux surfaces initiales (qui rappelons-le n'ont pas été modifiées grâce à l'opération de duplication), puis de reconstruire un nouveau modèle. Plusieurs modèles peuvent ainsi coexister lors d'une même session. Cela offre non-seulement l'avantage de pouvoir affiner une représentation pour un objet donné, mais on pourrait imaginer une application dans laquelle chaque modèle correspondrait à un instant t de l'existence de l'objet. La succession de modèles permettrait ainsi d'obtenir une idée de l'évolution temporelle d'un objet complexe.

CONCLUSION

Conclusion

Le système de modélisation volumique de surfaces *non-manifold* proposé dans ce manuscrit est basé sur l'utilisation intensive de la topologie. Cette dernière est composée des relations d'adjacence entre les *Faces* définissant les frontières des *Régions*. La gestion des adjacences est entièrement automatique. Dans un premier temps, des structures topologiques sont créées au niveau des triangles. Ensuite, ces structures sont utilisées pour la détection et la gestion des *Régions*.
Ce système de modélisation à base topologique comporte de nombreux avantages:

- Le domaine de modélisation est plus étendu que la majorité des systèmes Brep, car il prend en compte à la fois les objets *manifold* et *non-manifold*.
- La représentation topologique est suffisante et valide.
- Pour l'utilisateur, la création du modèle est aisée (surtout avec la méthode de construction automatique).

Cependant (nul n'est parfait), il existe également quelques inconvénients:

- Les informations contenues dans la base de données sont redondantes, comme dans tout système de représentation par surfaces-frontières (Brep).
- Il n'existe aucun moyen automatique de vérifier la cohérence des informations géométriques du modèle. En effet, la construction du modèle étant entièrement automatique, la validité de la géométrie dépend des données initiales fournies par l'utilisateur.
- La construction automatique du modèle est relativement longue. En fait le calcul des intersections entre les surfaces en vue de la création des structures d'adjacence s'avère l'opération la plus coûteuse en temps.

Les paragraphes qui suivent donnent une idée succinte des applications actuelles et futures de ce nouveau système de modélisation volumique *non-manifold*.

10.1 Remplissage de Régions

1. **Couleur:**
   Actuellement, chaque *Région* est représentée par ses frontières. Pour donner réellement

l'illusion de volume, on pourrait imaginer un moyen de colorer l'intérieur de chaque *Région*. L'idée est de définir un plan coupant le modèle, de façon à obtenir une cross-section. L'intersection entre ce plan et le modèle produit des courbes définissant des *Régions* en 2D. Chaque *Région* du plan de coupe est ensuite "peinte" en fonction de la couleur attribuée à ladite *Région* [29]. En remplaçant ce plan défini par l'utilisateur par le clipping plane (plan frontal de la caméra), une cross-section serait calculée et visualisée à chaque mouvement de la caméra.

2. **Texture:**
   L'un des domaines d'études actuels de l'équipe GOCAD est le Texture Mapping. Tout comme on tente d'affecter une texture précise à une surface, on pourrait imaginer d'affecter une texture à chaque *Région*, selon ses caractéristiques physiques.

3. **Tétraèdres:**
   Dans GOCAD, les surfaces sont modélisées sous forme de triangles. Logiquement, le maillage utilisé pour les *Régions* est composé de tétraèdres. Il existe deux façons de créer ces tétraèdres [27]:

   - A partir d'une surface fermée: l'intérieur de la surface est rempli de tétraèdres.
   - A partir d'un ensemble de surfaces: l'intérieur de l'enveloppe convexe des surfaces est rempli de tétraèdres.

   La tétraédrisation s'avère une méthode très pratique pour utiliser des algorithmes d'éléments finis dans des applications géostatistiques .

10.2 Interpolation de propriétés dans une Région

De nombreux travaux ont été effectués par l'équipe GOCAD afin d'appliquer des données relatives à des propriétés physiques sur une surface, puis sur un volume. Dans les deux cas, ces propriétés se présentent sous forme d'un ensemble de valeurs affectées à certains points de l'objet (surface ou volume). En utilisant le mesh de l'objet (triangles pour la surface, tétraèdres pour le volume), ces propriétés sont ensuite interpolées en utilisant l'algorithme D.S.I. [4].

A un autre niveau, on pourrait imaginer attacher des propriétés aux *Faces* du modèle. Par exemple, si l'on affecte une propriété de perméabilité aux points d'une *Face* partagée par deux *Régions*, ceci se révèlerait utile pour des applications comme la simulation de flux.

10.3 Vers une troisième méthode de construction du modèle

Une nouvelle méthode de modélisation de surfaces à partir d'un ensemble de lignes polygonales a été développée par l'équipe GOCAD. Ces lignes constituent le squelette d'une surface. Cette approche utilise la méthode Weiler 2D [7].

A partir d'un modèle Fil de Fer, et en définissant certaines contraintes pour assurer la validité du modèle, cette méthode de modélisation de surfaces permettrait de construire un

modèle surfacique. Ce modèle serait ensuite automatiquement transmis à notre système de modélisation pour transformer le modèle surfacique en modèle volumique.

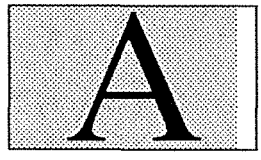
10.4 Orientation vers la quatrième dimension

Imaginons que l'utilisateur dispose d'un ensemble de données relatives à un même objet, à différents instants. A l'aide de notre méthode, un modèle volumique peut être construit pour chaque instant t de la vie de l'objet. La succession de ces modèles, suivant leur ordre chronologique, permet de créer un "film" de la vie de l'objet en question.

PARTIE IV

APPENDICES

Appendice



Base de données du modèle

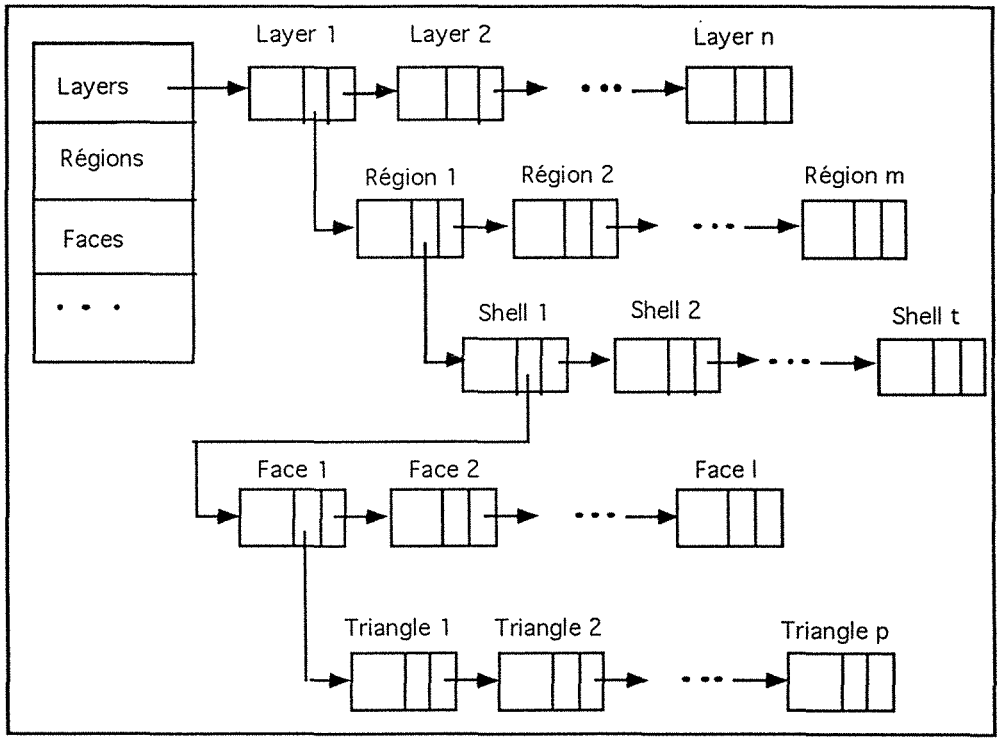


Figure A.1 Liaisons inter-structures.

Dans cette annexe, nous décrivons les structures de données utilisées pour la construction du modèle (voir Figure 3.5 dans Chapitre 3, Paragraphe 3.4.1). Pour la description de ces structures, nous utilisons des outils introduits dans GOCAD :

- le SET= ensemble d'objets ayant la même structure. L'accès à ces objets se fait par un parcours séquentiel.
- le SETK= ensemble d'objets ayant la même structure. L'accès à ces objets peut se faire soit par un parcours séquentiel, soit directement en utilisant une clé définie pour chaque objet.

La majorité des champs est commune aux deux méthodes exposées dans le manuscrit (méthode interactive et méthode automatique). Les champs spécifiques à l'une ou l'autre des méthodes sont exposés de façon explicite.

A.1 Modèle

Le modèle est la structure de plus haut niveau. Elle contient tous les éléments du modèle volumique. Les liaisons entre ces éléments sont présentées sur la figure A.1.

typedef struct MODEL_t

```
{
  SETK_t   *tlinks ;
  SET_t    *faces ;
  SET_t    *tsurfs ;
  SET_t    *layers ;
  SET_t    *regions ;
  int      last_region_number ;
  int      last_layer_number ;
}
```

Avec:

- **tlinks =**
  ensemble des *Tlink*s du modèle. L'accès à cette structure se fait à l'aide d'un triangle donné. Chaque élément de cette structure est un tableau de trois *Tlink*s, chacun associé à un edge du triangle ayant permis l'accès.
- **faces =**
  ensemble des *Face*s du modèle.
- **tsurfs =**
  ensemble des surfaces du modèle. Ces surfaces correspondent aux copies des données initiales.

- **layers =**
  ensemble des *Layers* du modèle.
- **regions =**
  ensemble des *Régions* du modèle.
- **last_region_number =**
  compteur de *Régions* pour la définition automatique du nom d'une nouvelle *Région*.
- **last_layer_number =**
  compteur de *Layers* pour la définition automatique du nom d'un nouveau *Layer*.

Des champs supplémentaires apparaissent selon la méthode choisie:

- Méthode interactive: SET_t **glines*:
  ensemble des *Glue-Lines* du modèle.
- Méthode automatique: SET_t **Radial_Edges*:
  ensemble des *radial_edges* du modèle.

A.2 Layer

La structure *Layer* correspond à la notion de couche dans une application géologique. Un *Layer* est composé d'un ensemble de *Régions* partageant certaines propriétés.

typedef struct LAYER_t

```
{
   char        *name ;
   SET_t       *regions ;
}
```

Avec:

- **name =**
  nom du *Layer*.
- **regions =**
  ensemble de *Régions* appartenant à ce *Layer*.

A.3 Région

La *Région* est la représentation, dans notre modèle, d'un volume clos de l'espace 3D de modélisation. Elle est définie de façon automatique et représentée par ses frontières. On peut lui

affecter diverses propriétés graphiques et physiques.

typedef struct REGION_t

```
{
   char        *name ;
   SET_t       *shells ;
   LAYER_t     *p_layer ;
}
```

Avec:

- **name =**
  nom de la *Région.*
- **shells =**
  ensemble des *Shells* constituant la frontière de la *Région.*
- **p_layer =**
  pointeur vers le *Layer* contenant la *Région.*

A.4 Shell

Un *Shell* correspond à une frontière d'une *Région.* Il est constitué de *Faces* adjacentes, chacune étant associée à une orientation (*Face-uses*).

typedef struct SHELL_t

```
{
   short         type ;
   int           shell_number ;
   REGION_t      *p_region ;
   SET_t         *faces ;
}
```

Avec:

- **type =**
  type du *Shell.* Ce flag prend la valeur 0 si le *Shell* est externe, 1 s'il est interne.
- **shell_number =**
  numéro du *Shell.*

- **p_region =**
  pointeur vers la *Région* dont ce *Shell* constitue une partie de la frontière.

- **faces =**
  ensemble des *Faces* constituant ce *Shell.*

A.5 Face

Une *Face* est un ensemble de triangles correspondant à un sous-ensemble d'une surface. Associée à une orientation ('+' ou '−' en fonction des vecteurs normaux de ses triangles), elle définit une partie d'une frontière d'une *Région*, encore appelée *Face-use*. La structure *Face-use* n'existe pas réellement. Mais elle peut être aisément retrouvée grâce à la *Face* et à une orientation 0 ou 1 (correspondant à '+' et '−').

typedef struct FACE_t

```
{
  SET_t      *trgls ;
  SHELL_t    *p_shell[2] ;
  short       side_traversed[2] ;
}
```

Avec:

- **trgls =**
  ensemble des triangles définissant la *Face*.

- **p_shell[2] =**
  pointeurs vers les *Shells* de chaque *Face-use* associée à la *Face*.
  p_shell[0] donne le *Shell* associé au côté positif de la *Face*.
  p_shell[1] donne le *Shell* associé au côté négatif de la *Face*.

- **side_traversed[2] =**
  flags utilisés lors de la détection des *Régions*.
  side_traversed[0] est utilisé pour marquer le côté positif de la *Face*.
  side_traversed[1] est utilisé pour marquer le côté négatif de la *Face*.

A.6 Tlink

Le *Tlink* est la structure permettant de connaitre parfaitement les adjacences d'un triangle donné du bord d'une *Face*.

typedef struct TLINK_t

```
{
   TRGL_t    *p_trgl ;
   short      edge_num ;
   TRGL_t    *p_adj_trgl[2] ;
   short      side_switch[2] ;
}
```

Avec:

- **p_trgl** =
  triangle associé au *Tlink*.
- **edge_num** =
  edge du triangle associé au *Tlink*.
- **p_adj_trgl[2]** =
  deux triangles adjacents à l'edge du triangle.
  p_adj_trgl[0] donne le triangle adjacent au côté positif du triangle.
  p_adj_trgl[1] donne le triangle adjacent au côté négatif du triangle.
- **side_switch[2]** =
  côtés des triangles adjacents en relation avec un côté du triangle.
  side_switch[0] donne le côté du triangle adjacent **p_adj_trgl[0]** en relation avec le côté positif du triangle.
  side_switch[1] donne le côté du triangle adjacent **p_adj_trgl[1]** en relation avec le côté négatif du triangle.

A.7 Glue-Line

La *Glue-Line* est l'outil mis en place dans la méthode de construction interactive du modèle. Elle permet de créer de façon "manuelle" des liens entre les différents objets en présence.

A.7.1 Structure d'un segment de la Glue-Line

typedef struct GLSEG_t

```
{
   ATOM_t    *p_atom[2] ;
   TRGL_t    *p_trgl[2] ;
   short      tedge_num[2] ;
   GLSEG_t   *next ;
}
```

Avec:

- **p_atom[2] =**
  information géométrique de la *Glue-Line*. Les deux atomes correspondent aux extrémités du segment.

- **p_trgl[2] =**
  information topologique de la *Glue-Line*. Les deux triangles correspondent aux triangles de la *Surface-Mère* adjacents au segment.

- **tedge_num[2] =**
  numéro des edges des triangles adjacents au segment.

- **next =**
  pointeur vers le segment suivant de la *Glue-Line*.

A.7.2 Structure Générale de la Glue-Line

typedef struct GLINE_t

```
{
   char      *name ;
   short      type ;
   TSURF_t   *p_owner ;
   char      *master_gline_name ;
   SET_t     *attached_tsurf_set ;
   GLSEG_t   *p_glseg ;
}
```

Avec:

- **name =**
  nom de la *Glue-Line*.

- **type =**
  statut de la *Glue-Line* (EXISTING, MASTER, SLAVE ou USED).

- **p_owner =**
  pointeur vers la *Surface-Mère* de la *Glue-Line*.

- **master_gline_name =**
  dans le cas d'une *Glue-Line* dans l'état USED, ce champ contient le nom de la *Glue-Line* MASTER à laquelle elle a été fusionnée. Dans tous les autres cas, ce champ est vide.

- **attached_tsurf_set =**
  dans le cas d'une *Glue-Line* dans l'état MASTER, ce champ contient l'ensemble des surfaces qui ont été collées à cette *Glue-Line*. Dans tous les autres cas, ce champ est vide.

- **p_glseg =**
  liste des segments définissant la *Glue-Line*.

Appendice

B

Opérations de manipulation du modèle

B.1 Introduction

Dans cette annexe, nous présentons les divers moyens mis à la disposition de l'utilisateur pour utiliser un modèle créé à l'aide de la méthode interactive. Ces opérations comportent les outils de création, modification et destruction du modèle.

B.2 Création et modification du modèle

Les opérations de création et de gestion du modèle sont entièrement réalisées par l'intermédiaire des *Glue-Lines* (voir Partie II). Elle permettent de créer et de gérer les diverses entités rattachées directement à la structure globale du modèle (voir Figure B.1). Ces opérations sont:

- Création d'une *Glue-Line* sur une surface.
- Ajout d'une surface dans le modèle.
- Fusion de deux *Glue-Lines*.

B.2.1 Création du modèle

Il n'existe pas de commande spécifique précisant qu'un nouveau modèle va être créé. En effet, cette création s'effectue lors de la première utilisation, au cours d'une session, d'une commande en relation avec notre modèle. Cette commande peut être:

- Création d'une *Glue-Line* sur une surface.

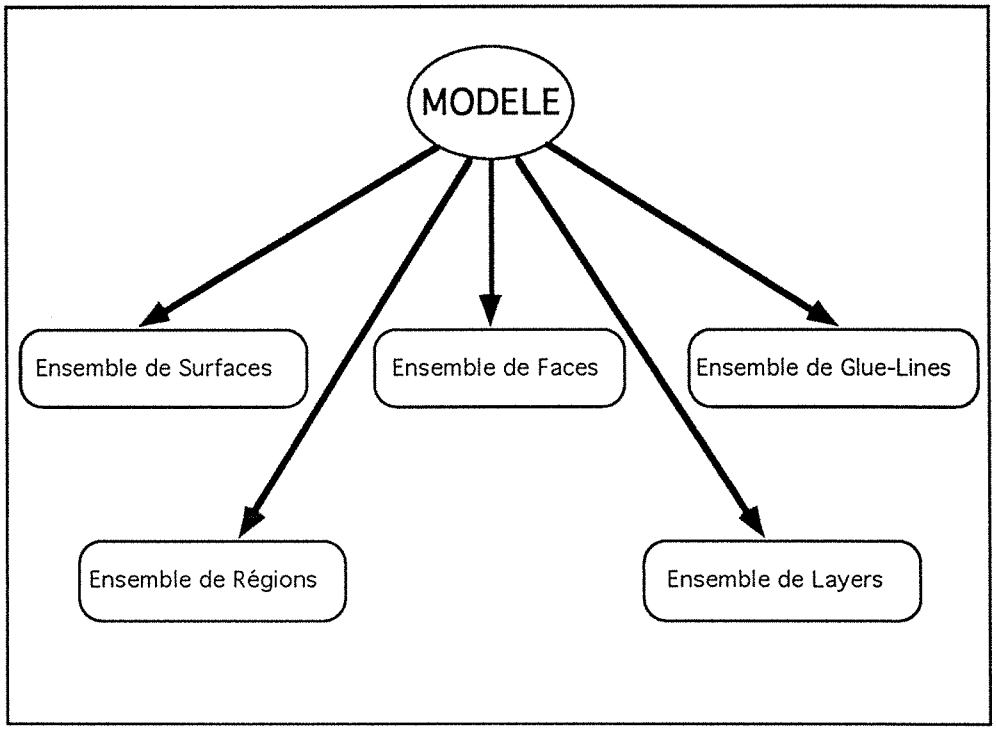


Figure B.1 La structure globale du modèle.

- Ajout d'une surface dans le modèle.

Imaginons que l'utilisateur dispose d'un ensemble de surfaces pour lequel il désire construire un modèle volumique. La première opération qu'il effectue dans cette direction provoque la création d'un nouveau modèle. Les opérations suivantes, quand à elles, ne font que modifier le modèle existant.

Un seul modèle peut exister lors d'une session et son nom est fixé automatiquement. Ceci s'explique par le fait que la création d'un modèle provoque la modification de la géométrie et de la topologie des surfaces initiales. En conséquence, les surfaces initiales, adaptées à un modèle donné, ne peuvent être à nouveau utilisées pour la construction d'un autre modèle. On comprend aisément que l'opération de *fusion de deux Glue-Lines* ne peut, comme les deux autres opérations disponibles, provoquer la création d'un modèle. En effet, cette opération nécessite l'existence de deux *Glue-Lines*, et on a dit précédemment que la création d'une *Glue-Line* peut provoquer la création d'un modèle, si nécessaire. La création du modèle comporte les actions suivantes:

1. **Création de la *Région* initiale:**
   Une entité *Région*, sans *Shell*, est créée et stockée dans la structure globale du modèle. Cette *Région* initiale correspond à l'Univers (voir Chapitre 2, Paragraphe 2.2).

2. **Création du *Layer* initial:**
   Un *Layer* associé à la *Région* définissant l'Univers est créé.

3. **Définition du nom du modèle:**
   Le nom du modèle (HMODEL) est affecté à la nouvelle entité Modèle.

B.2.2 Modification du modèle

Une fois le modèle créé, toutes les opérations suivantes sont considérées comme des modifications du modèle. Ces opérations peuvent être:

- Création d'une *Glue-Line* sur une surface.
- Ajout d'une surface dans le modèle.
- Fusion de deux *Glue-Lines*.

Création d'une Glue-Line sur une surface

Cette opération, réalisée à l'aide de l'une des trois méthodes présentées dans le chapitre 6, paragraphe 6.2, provoque les actions suivantes au niveau de la structure globale du modèle:

1. **Stockage de la *Glue-Line*:**
   La *Glue-Line* est stockée dans un ensemble rattaché au modèle.

2. **Stockage de la *Surface-Mère* de la *Glue-Line*:**
   La surface à laquelle la *Glue-Line* est associée est elle-aussi stockée dans un ensemble défini au niveau du modèle. Cette opération est réalisée uniquement pour garder un souvenir des surfaces initiales ayant servi de support au modèle.

3. **Stockage des *Faces* originaires de la surface:**
   La (ou les) *Face*(s) créée(s) à partir de la surface est(sont) stockée(s) dans un ensemble défini au niveau du modèle.

4. **Stockage des *Tlinks*:**
   Les *Tlinks* créés suite à la création d'une *Glue-Line* sur une surface sont stockés dans un ensemble défini dans la structure globale du modèle (voir Chapitre 6, Paragraphe 6.4). Ces *Tlinks* décrivent les relations d'adjacence des triangles du bord de chaque *Face* dont le bord est défini en partie par la nouvelle *Glue-Line*.

Ajout d'une surface dans le modèle

Cette opération a été mise en place pour permettre la prise en compte, dans la définition du modèle, de certaines surfaces sur lesquelles aucune *Glue-Line* n'est créée. En particulier, les surfaces fermées, c'est-à-dire les surfaces sans bord, sont incluses dans le modèle de cette façon. Bien évidemment, toute autre surface peut être traitée en utilisant cette méthode. Cette opération provoque les actions suivantes au niveau du modèle:

1. **Stockage de la surface:**
   La surface qui doit être ajoutée au modèle est stockée dans un ensemble défini au niveau du modèle.

2. **Stockage des *Faces* originaires de la surface:**
   La (ou les) *Face*(s) créée(s) à partir de la surface est(sont) stockée(s) dans un ensemble défini au niveau du modèle.

3. **Stockage des *Tlinks*:**
   Les *Tlinks* créés suite à l'ajout d'une surface dans le modèle sont stockés dans un ensemble défini dans la structure globale du modèle. Ces *Tlinks* décrivent les relations d'adjacence des triangles du bord de chaque *Face* composant la nouvelle surface[1].

4. **Dans le cas d'une *Face* fermée:**

   - Stockage de la *Région* décrivant l'intérieur de cette *Face*.
   - Stockage du *Layer* associé à cette *Région*.

Fusion de deux Glue-Lines

La *fusion de deux Glue-Lines*, pour permettre un *collage entre deux surfaces* (voir Chapitre 7), comporte les action suivantes, au niveau du modèle:

1. **Stockage des nouvelles *Régions*:**
   Si l'algorithme de détection des *Régions* qui a suivi la *fusion des deux Glue-Lines* a résulté en la création de deux nouvelles *Régions*, chacune d'elles est stockée dans un ensemble défini au niveau de la structure globale du modèle.

2. **Stockage des nouveaux *Layers*:**
   Les *Layers* associés aux nouvelles *Régions* sont eux-aussi stockés dans un ensemble défini dans la structure Modèle.

3. **Stockage des *Tlinks*:**
   Les *Tlinks* créés ou modifiés suite à la *fusion de deux Glue-Lines* sont stockés dans un ensemble défini dans la structure globale du modèle.

4. **Dans le cas d'un *collage* indirect:**
   La surface intermédiaire créée pour lier les deux *Glue-Lines* est stockée dans l'ensemble de surfaces défini au niveau de la structure globale du modèle.

[1] Dans le cas, d'une *Face* fermée, aucun *Tlink* n'est stocké, car il n'existe pas de bord à la *Face*.

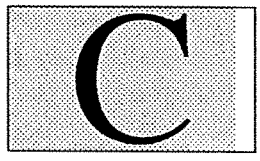
B.2.3 Destruction du modèle

La destruction d'un modèle s'effectue tout d'abord en détruisant toutes les entités qui le définissent, à savoir:

- Les *Faces*.
- Les *Shells*.
- Les *Régions*.
- Les *Layers*.
- Les *Glue-Lines*.
- Les *Tlinks*.

Les surfaces sont ensuite restaurées grâce aux entités stockées dans le modèle. Ces surfaces ne correspondent pas aux données initiales car, rappelons-le, les modifications topologiques nécessaires à la création du modèle s'effectuent directement sur les données de départ. Cette opération est suivie de la destruction de l'entité Modèle.

Appendice



Opérations Graphiques

C.1 Introduction

Dans cette annexe, nous proposons les divers outils mis à la disposition de l'utilisateur pour faciliter son interprétation du modèle. Ces outils comportent des opérateurs graphiques de visualisation et de localisation, ainsi que des opérateurs de construction. Certaines de ces opérations sont communes aux deux méthodes de construction du modèle (méthode interactive et méthode automatique).

C.2 Visualisation du modèle

Les opérations présentées dans ce paragraphe sont valables uniquement pour un modèle construit à l'aide de la méthode interactive. Deux modes de visualisation du modèle sont disponibles:

- mode WireFrame
- mode Région

C.2.1 Visualisation en mode WireFrame

Dans ce type de visualisation, seules les liaisons entre les *Faces* du modèle sont dessinées. L'utilisateur choisissant de représenter un modèle WireFrame peut donc visualiser la représentation des *Tlinks* du modèle. Cette représentation est très pratique lors de la construction du modèle, car elle permet de visualiser de deux couleurs différentes les *Tlinks* associés à une *Glue-Line* SLAVE et ceux d'une *Glue-Line* MASTER.

C.2.2 Visualisation en mode Région

Ce mode de visualisation consiste à représenter les *Régions* du modèle par leurs frontières (*Shells*). La couleur de chaque *Région* peut être modifiée à tout moment sans affecter de quelque façon que se soit celle des autre *Régions* du modèle. De plus l'utilisateur a la possibilité de représenter uniquement une partie de l'ensemble des *Régions*.

Dans ce type de visualisation, le problème majeur consiste à représenter simultanément les deux *Régions* partageant une même *Face*. En effet, sachant que les couleurs de ces deux *Régions* sont différentes, quelle doit être la couleur affectée à la *Face* commune? A ce stade de la recherche, nous avions choisi de représenter cette *Face* en utilisant la couleur de la dernière des deux *Régions* qui ait été affichée. Une amélioration importante à cette visualisation sera présentée dans le dernier paragraphe de cette annexe (voir Paragraphe C.5).

C.3 Localisation d'un point dans le modèle

Cet opérateur est commun aux deux types de construction du modèle (méthode interactive et méthode automatique).

Le "localisateur" permet de situer très rapidement un point dans le modèle. En d'autres termes, il permet de savoir dans quelle *Région* se situe un point donné. Cette opération est très rapide et se réalise de la façon suivante:

1. Choix d'un point par l'utilisateur.
2. Création d'une ligne de tir horizontale à partir du point donné.
3. Calcul du point d'intersection entre la ligne de tir et la première surface rencontrée.
4. Recherche du triangle contenant le point d'intersection.
5. Recherche de la *Face* contenant le triangle intersecté.
6. Calcul de la position du point par rapport à la *Face* (côté '+' ou côté '−').
7. La *Face-use* obtenue permet d'accéder directement à un *Shell* unique, puis à une *Région* unique contenant le point donné.

REGION_t * POINT_Get_REGION (*M*, *point*)

- **Input:**
  - *point* = point à localiser.
  - Modèle *M*.
- **Output:** *Région* contenant le point donné.

{
/ Création de la ligne de tir */*
line→ origine = *point* ;
line→ direction = [1. 0, 0] ;
/ Calcul de l'intersection entre la ligne et la première surface rencontrée. */*
min_dist = 1.*e*30 ;
pour (chaque surface $S_i \in M$)
{
• [*impact_trgl* , *impact_point*] = TSURF_**Shoot** (S_i , *line*) ;
/ Calcul de la distance entre le point donné et le point d'impact,s'il existe. */*
si ($\|(impact\_point, point)\| < min\_dist$)
{
• *closest_trgl* = *impact_trgl* ;
• *min_dist* = $\|(impact\_point, point)\|$;
}
}
si (*closest_trgl* == NULL)
{
pour (chaque région $R_i \in M$)
{
si (R_i == "Universe") **return** (R_i) ;
}
}
/ Retrouver la Face contenant le triangle d'impact (fonction* **Triangle2Face()***). */*
closest_Face = **Triangle2Face** (*closest_trgl*) ;
/ Recherche de la position du point donné par rapport à la Face.*
On considère que le triangle est doté d'un vecteur normal $\overrightarrow{N}$.
Soit $\overrightarrow{V}$ *le vecteur directeur de la ligne de tir line. */*
si ($\overrightarrow{V} \wedge \overrightarrow{N} < 0$)
closest_Side = 0 ;
sinon *closest_Side* = 1 ;

return (*closest_Face*— shell[*closest_Side*]— région) ;
}

C.4 Opérations de construction

Les opérateurs présentés dans ce paragraphe sont communs aux deux types de construction du modèle.
De nouveaux objets peuvent être créés à partir du modèle solide:

- **Création d'une surface à partir de la frontière d'une *Région*:**
  Pour une *Région* donnée, il est possible de créer une nouvelle surface triangulée correspondant à sa frontière. Cette surface est obtenue en regroupant dans un même objet tous les *Shells* définissant la *Région* choisie. La surface obtenue est fermée (les *Régions* étant des volumes clos de l'espace 3D).

- **Création d'une surface à partir d'une *Face*:**
  Il est également possible de créer une nouvelle surface triangulée à partir d'une *Face* du modèle.

La création d'une surface fermée à partir de la frontière d'une *Région* trouve une application précise dans GOCAD. En effet, il est possible de tétrahédriser l'intérieur de cette surface. On peut alors appliquer des propriétés physiques sur le nouvel objet obtenu, puis interpoler ces mêmes propriétés.

C.5 Optimisations des opérations graphiques

Cette partie regroupe toutes les opérations de visualisation et de localisation relatives à un modèle construit de façon automatique.

C.5.1 Visualisation en mode WireFrame

Ce mode de visualisation consiste à représenter la structure *Radial-Edges* du modèle, c'est-à-dire le bord global du modèle. On offre également la possibilité de différencier les *radial-edges* avec plusieurs *radial-triangles* (segments résultant d'une opération Cut, donc d'une *association de Faces*) et les *radial-edges* avec un seul *radial-triangle* (segments du bord initial d'une surface). Ceci permet de découvrir d'éventuelles erreurs (par exemple un "trou" entre deux surfaces qui devraient être parfaitement collées), erreurs dues à une mauvaise préparation des données initiales.

C.5.2 Visualisation en mode Région

Ce mode de visualisation consiste à représenter les *Régions* du modèle par leurs frontières (*Shells*). La couleur de chaque *Région* peut être modifiée à tout moment sans affecter de quelque façon que se soit celle des autre *Régions* du modèle.

L'un des problèmes majeur auquel nous avons été confrontés lors de l'implantation de ce mode de visualisation a été la représentation des *Faces* partagées par deux *Régions*. En effet,

chaque *Région* ayant sa propre couleur, comment choisir la couleur de la *Face* en question? Deux solutions sont envisageables:

1. par translation de la *Face* en fonction de son utilisation par chaque *Région.*

2. en utilisant le Center Of Projection de la camera.

Translation d'une Face partagée

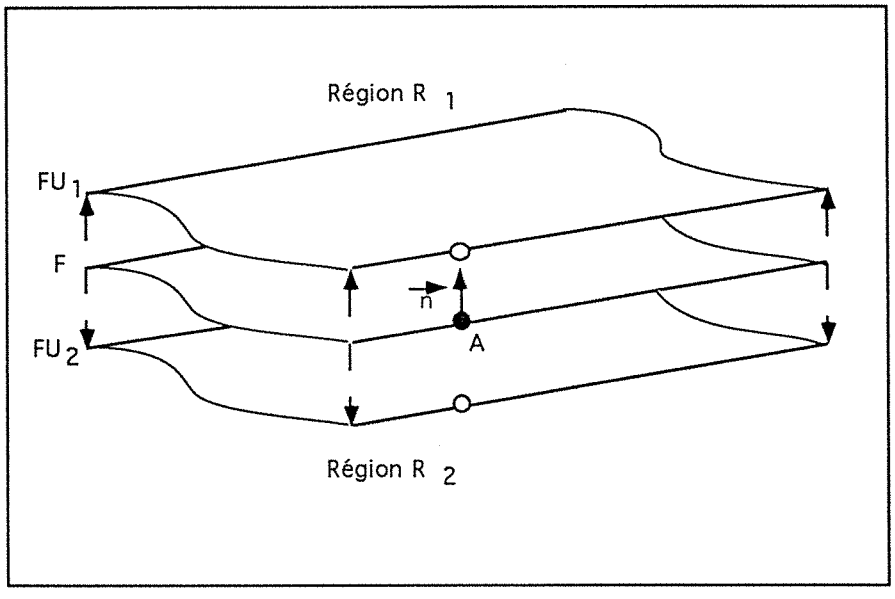


Figure C.1 *Face* partagée - Représentation par translation.

Soit F la *Face* à représenter.
Soit FU_1 la *Face-use* associée au côté positif de F. FU_1 délimite la *Région* R_1.
Soit FU_2 la *Face-use* mate de FU_1, donc $FU_2 = [F\ , '-']$. FU_2 appartient à la frontière de la *Région* R_2.
L'idée est de représenter physiquement chaque *Face-use* de F à l'aide de la couleur de la *Région* associée à chacune d'elles. La *Face* sert de support à la représentation de ses *Face-uses*. En effet, la position spatiale de chaque point d'une *Face-use* est calculée en effectuant une translation des points de la *Face*. Cette translation est réalisée dans la direction du vecteur normal associé à chaque point. On sait, par définition, que l'orientation d'une *Face* est donnée par les vecteurs normaux en chacun de ses points. Donc dire que la *Face-use* FU_1 de F définit la frontière de la *Région* R_1, c'est dire que les vecteurs normaux à chaque point de F pointent vers l'intérieur de R_1. En conséquence, le sens du vecteur de translation est défini par le côté de la *Face* auquel la *Face-use* est associée. Donc:

1. Représentation de F pour la *Région* R_1:
   Pour un point A de F:
   - **direction du vecteur de translation** = vecteur normal associé à A.
   - **sens du vecteur de translation** = sens du vecteur normal associé à A.
2. Représentation de F pour la *Région* R_2:
   Pour un point A de F:
   - **direction du vecteur de translation** = vecteur normal associé à A.
   - **sens du vecteur de translation** = sens inverse du vecteur normal associé à A.

Les translations sont de l'ordre d'un espilon donné. Le résultat obtenu à l'aide de ce type de repésentation n'est pas parfait en certains points critiques, en raison de problèmes d'ordre numérique.

Utilisation du Center Of Projection de la camera

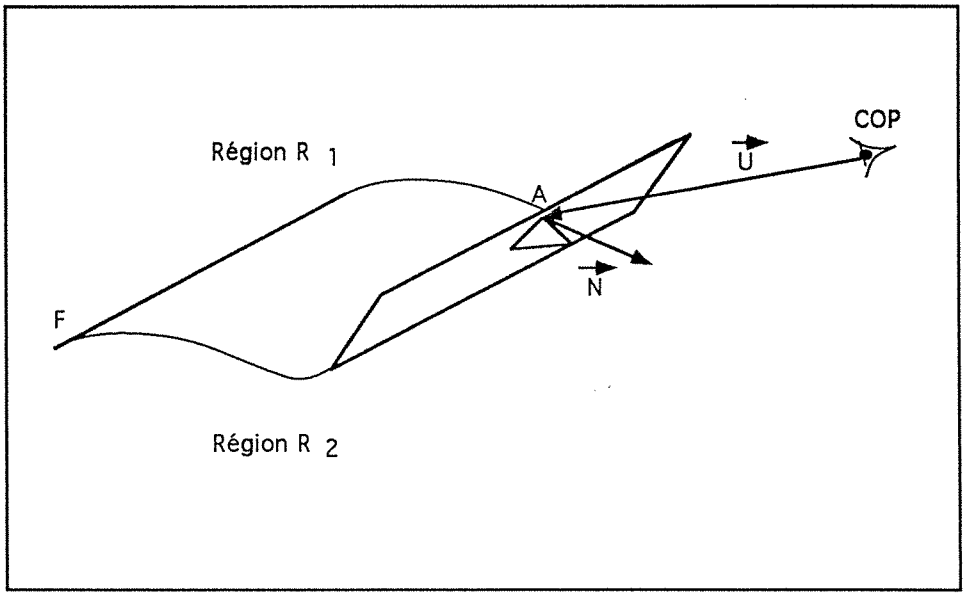


Figure C.2 *Face* partagée - Représentation utilisant le COP.

Le **Center Of Projection** (abrégé **COP**) d'une caméra est le centre optique de la caméra. Cette seconde méthode de représentation des *Faces* partagées consiste à déterminer la couleur avec laquelle chaque triangle de la *Face* doit être dessiné, en fonction de sa position dans la caméra, par rapport au COP.

/ Le COP est supposé connu grâce à la variable C. */*

pour (chaque triangle T_i de la Face F)

{

- Soit A l'un des sommets de T_i ;
- Soit $\overrightarrow{N}$ la normale au triangle T_i ;
- $\overrightarrow{U} = \overrightarrow{CA}$;

/ Calcul du produit scalaire des deux vecteurs $\overrightarrow{U}$ et $\overrightarrow{N}$. */*

- $PS = \overrightarrow{N} \cdot \overrightarrow{U}$;

/ Test du signe du produit scalaire. */*

si ($PS > 0$)

alors

Représentation du côté positif de la Face.

sinon

Représentation du côté négatif de la Face.

}

Cette méthode offrant des résultats parfaits et très précis, elle a été conservée dans la version finale du système de modélisation volumique *non-manifold*.

LEXIQUE

Lexique

Notes:

- Ce lexique concerne les mots spécifiques à notre modèle.
- Les mots imprimés en italique dans le rapport sont redéfinis ici.
- Les mots apparaissant en gras dans une définition font référence à d'autres éléments définis dans ce lexique.

****

Association de Faces — C'est le résultat, au niveau interne, de la **fusion de deux Glue-Lines**, c'est-à-dire du **collage de deux surfaces**.

Collage de deux surfaces — C'est l'action résultant, au niveau de l'utilisateur, de la **fusion de deux Glue-Lines**.

Décollage de surfaces — C'est la rupture d'un lien existant entre deux surfaces. Cette opération équivaut au "undo" de l'opération de **collage de deux surfaces**.

Face — Une **Face** est un ensemble de triangles connectés d'une surface. Cette notion est toujours utilisée en association avec une orientation ('+' ou '−').

Face-use — On nomme **Face-use** d'une **Face** F, et on note

$$[F, side]$$

le couple composé de la **Face** F et de l'un de ses côtés ('+' ou '−').
Toute **Face** F a deux **Face-uses**.

Face-use mate — Les **Face-uses** fonctionnent par paire. Chaque élément de ce couple est appelé **Face-use mate** de l'autre élément.

Fusion de deux Glue-Lines C'est l'opération permettant à l'utilisateur de créer un lien entre deux surfaces données.

Glue-Line Une **Glue-Line** est une ligne définie par l'utilisateur sur une surface donnée S_1 dans l'objectif de **coller la surface** S_1 à une autre surface S_2. Ceci est réalisé en fusionnant la **Glue-Line** définie sur S_1 à celle définie sur S_2.

Layer Un **Layer** est un ensemble de **Régions** partageant certaines propriétés (propriétés physiques, graphiques...).

Manifold Dans une représentation 2-**manifold**, chaque point d'une surface à un voisinage homéomorphe à un disque 2D.

Modèle Dans notre contexte de modélisation volumique **non-manifold**, le **modèle** est la décomposition de l'espace 3D de modélisation sous forme de volumes fermés distincts appelés **Régions**.

Non-manifold Les conditions **non-manifold** sont:

- Deux surfaces se touchant en un point unique.
- Deux surfaces se touchant le long d'une courbe ouverte ou fermée.
- Deux volumes distincts fermés partageant une face.
- Une droite émanant d'un point d'une surface.

Le domaine de définition de la modélisation géométrique **non-manifold** comprend les conditions **manifold** et **non-manifold**.

Radial-Edges La structure **Radial-Edges** est créée lors de la construction automatique du modèle. Elle décrit tous les segments du bord de chaque **Face** composant le modèle. Cette structure est utilisée pour la création des **Tlinks** du modèle.
Elle est composée d'une liste de **radial-edges**.

radial-edge Le **radial-edge** est la composante principale de la structure **Radial-Edges**. Un **radial-edge** correspond à un segment du bord d'une **Face**.
En plus de sa géométrie (atomes-extrémité), un **radial-edge** contient des informations sur les triangles qui lui sont adjacents.

radial-triangle Un **radial-triangle** est un triangle associé à un **radial-edge**. Il a un edge commun avec le **radial-edge**, ce qui nous fait dire qu'il est adjacent au **radial-edge**.

Région Une **Région** est un volume fermé de l'espace 3D, défini par ses frontières, ou **Shells**.

Shell Le **Shell** est une frontière d'une **Région** donnée. C'est en fait un ensemble de **Face-uses** adjacentes délimitant une **Région**. Un **Shell** peut être interne ou externe.

Surface-Mère Une surface S ayant une **Glue-Line** GL associée est appelée **Surface-Mère** de cette **Glue-Line**.

Tlink Un **Tlink** est une structure de données décrivant les relations d'adjacence d'un triangle d'une **Face**. Pour chaque edge E d'un triangle T, on appelle **Tlink**, et on note

$$Tlink(T, E) = \{[T_1, side_1], [T_2, side_2]\}$$

le couple composé de deux **Triangle-uses** adjacents.

Triangle-Link (voir **Tlink**).

Triangle-use On appelle **Triangle-use** d'un triangle T, et on note

$$[T, side]$$

le couple composé du triangle T et de l'un de ses côtés ('+' ou '−').
Tout triangle T a deux **Triangle-uses**.

Triangle-use mate Les **Triangle-uses** fonctionnent par paire. Chaque élément de ce couple est appelé **Triangle-use mate** de l'autre élément.

Bibliographie

[1] Borianne, Ph. et Jaeger, M., *Représentation à base topologique sur un espace discret.* Colloque Géométrie Discrète en Imagerie, Strasbourg, 20-21 Septembre 1993.

[2] Bouraz, A., *Modèles géométriques des surfaces, création et manipulation dans une démarche algébrique.* Thèse de Doctorat, Université Claude Bernard, Lyon I, Juin 1992.

[3] Chipot, Y., *Génération et Modification de surfaces triangulées.* Thèse INPL, Nancy, Juillet 1991.

[4] Cognot, R., *Définition, Manipulation et Interpolation de Propriétés Physiques en 3D.* Thèse INPL, Nancy (en préparation).

[5] Colnet, B., *Modélisation Volumique à l'aide de faces frontières.* Rapport de DEA, Université de Nancy I, Septembre 1991.

[6] Conraud, J., *Modélisation volumique d'objets naturels.* Rapport de DEA, Université de Nancy I, Septembre 1992.

[7] Della Malva, R., Mallet, J.L. et Mariez, O., *Modeling surfaces by interpolating polygonal lines.* Proceedings GOCAD Meeting Novembre 1992.

[8] Desaulniers, H. et Stewart, N.F., *An Extension of Manifold Boundary Representations to the r-Sets.* ACM Transactions on Graphics, Volume 11, Nb 1, Janvier 1992, Pages 40 à 60.

[9] Foley, J.J., Van Dam, A., Feiner, S.K. et Hughes, J.F., *Computer Graphics - Principles and Practice.* Second Edition, Addison-Wesley Publishing Company, 1990.

[10] Françon, J. *Surfaces Discrètes.* Colloque Géométrie Discrète en Imagerie, Strasbourg, 20-21 Septembre 1993.

[11] Gardan, Y., *Eléments méthodologiques pour la réalisation de systèmes de CFAO et leur introduction dans les entreprises.* Thèse INPL, Grenoble, Décembre 1982.

[12] Goldman, R.N. *The Role of Surfaces in Solid Modeling.* Geometric Modeling, G.E. Farin, ed. SIAM, 1987.

[13] Haldorsen, H.H. et Damsleth, E., *Stochastic Modeling.* SPE, Avril 1990.

[14] Hearn, D. et Baker, M.P., *Computer Graphics.* Prentice Hall, 1986.

[15] Huang, Y., *Modélisation et Manipulation de surfaces triangulées.* Thèse INPL, Nancy, Novembre 1990.

[16] Irwin, P.A. et Paradis, A., *Interactive Volume Modeling.* Proceedings, EAEG Florence, Mai 1991.

[17] Kondo, K., *PIGMOD: Parametric and Interactive Geometric Modeller for mechanical design.* Computer Aided Design, Vol. 22, Nb 10, Décembre 1990.

[18] Lamboglia, K., *Contraintes Cinématiques sur des surfaces naturelles.* Rapport de DEA, Université de Nancy I, Septembre 1990.

[19] Lamboglia, K., *Solid Modeling of Non-Manifold Triangulated Objects.* ACM Transactions on Graphics, Décembre 1993 (en attente).

[20] Lamboglia, K., *Interactive Building of a Non-Manifold Solid Object.* IEEE TENCON '94, special session on Computer Graphics and Applications, Singapour, Août 1994 (en attente).

[21] Lamboglia, K., *Solid Modeling of Non-Manifold Geometry.* International Conference on CAGD, Penang, Juillet 1994 (accepté).

[22] Le Melinaire, P., *Modélisation de Relations Géométriques par la méthode DSI- Application à la Géologie.* Thèse INPL, Nancy, Janvier 1992.

[23] Liebling, T. et Röthlisberger, H., *Infographie et Applications.* ed. Masson, 1988.

[24] Mallet, J.L., *Discrete Smooth Interpolation in Geometric Modelling.* Computer Aided Design, Volume 24, Nb 4, Avril 1992, Pages 178 à 191.

[25] Mallet, J.L., *Discrete Smooth Interpolation.* ACM Transactions on Graphics, Volume 8, Nb 2, Avril 1989, Pages 121 à 144.

[26] Mallet, J.L. et Nobili, P., *Domains in the 3D Modelling tool* GOCAD. Proceedings, EAEG Florence, Mai 1991.

[27] Mallet, J.L., GOCAD *Report.* Version 8.1, Volumes 1 et 2, Mars 1993.

[28] Mantyla, M., *An Introduction to Solid Modeling.* Computer Science Press, 1988.

[29] Mariez, O., *Génération de courbes et Modélisation de Surfaces.* Rapport de DEA, Université de Nancy I, Septembre 1993.

[30] Mortenson, M.E., *Geometric Modeling.* Wiley, 1985.

[31] Mortenson, M.E., *Computer Graphics - An Introduction to the Mathematics and Geometry.* Industrial Press Inc., 1989.

[32] Pratt, M.J., *Solid Modelling- Survey and Current Research Issues.* Computer Graphics Techniques- Theory and Practice, D.F Rogers and R.A. EarnShaw Editors, ed. Springer-Verlag, 1990.

[33] Requicha, A., *Representations for Rigid Solids: Theory, Methods and Systems.* ACM Computing Surveys, Décembre 1980

[34] Requicha, A. et Voelcker, H., *Constructive Solid Geometry.* Production Automation Project Tech. Memo 25, Univ. Rochester, Novembre 1977.

[35] Shephard, M., *Geometric Modeling needs of Finite Element Modeling.* IFIP WG5.2 Working Conference on Geometric Modeling for CAD Applications, Rensselaerville, NY, Mai 1986.

[36] Sword, C.H., *Building Flexible Interactive Geologic Models.* Proceedings, SEG Houston, Novembre 1991.

[37] Van, B., Pajon, J.L., Joseph, P. et Chautru, J.M., *3D Reservoir Visualization.* SPE, Novembre 1991.

[38] Weiler, K., *The Radial-Edge Structure:A Topological Representation for Non-Manifold Geometric Boundary Modeling.* IFIP WG5.2 Working Conference on Geometric Modeling for CAD Applications, Rensselaerville, N.Y., Mai 1986.

[39] Weiler, K., *Topological Structure for Geometric Modeling.* PhD Thesis, Rensselaer Polytechnic Institute, Août 1986.

[40] Weiler, K., *Boundary Graph Operators for Non-Manifold Geometric Modeling.* IFIP WG5.2 Working Conference on Geometric Modeling for CAD Applications, Rensselaerville, NY, Mai 1986.

[41] Weiler, K., *Topology as a Framework for Solid Modeling.* Proceedings, Graphics Interface, Ottawa, Mai 1984.

[42] Weiler, K., *Edge Based Data Structures for Solid Modeling in Curved-Surface Environments.* IEEE Computer Graphics and Applications., Janvier 1985.

[43] Weiler, K., *Adjacency Relationships in Boundary-Graph-Based Solid Models.* Juin 1983.

INSTITUT NATIONAL
POLYTECHNIQUE
DE LORRAINE

AUTORISATION DE SOUTENANCE DE THESE DU DOCTORAT DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

o0o

VU LES RAPPORTS ETABLIS PAR :

Monsieur CINQUIN Philippe,, Professeur, Univeristé Joseph Fourier, La Tronche,

Monsieur LIENHARDT Pascal, Chargé de Recherche CNRS, Université Louis Pasteur, Strasbourg,

Monsieur PAUL Jean-Claude, Responsable Graphis, CRIN Vandoeuvre.

NANCY BRABOIS
2, AVENUE DE LA
FORET-DE-HAYE
BOITE POSTALE 3
F - 5 4 5 0 1
VANDŒUVRE CEDEX

Le Président de l'Institut National Polytechnique de Lorraine, autorise :

Mademoiselle LAMBOGLIA Karine

à soutenir devant l'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE,
une thèse intitulée :
"Modélisation volumique des surfaces non-manifold"

en vue de l'obtention du titre de :

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

Spécialité : **"INFORMATIQUE"**

Fait à Vandoeuvre le, **28 Janvier 1994**
Le Président de l'I.N.P.L.,
M. LUCIUS

INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE NANCY

TEL. 33/83.59.59.59
FAX. 33/83.59.59.55

Résumé

La plupart des systèmes de modélisation volumique utilisant la représentation par surfaces-frontières ne considèrent que la géométrie manifold. Pour un objet manifold, et plus précisément 2-manifold, chaque point a un voisinage homéomorphe à un disque 2D. Cette restriction du domaine de représentation constitue un inconvénient majeur pour des applications manipulant des surfaces naturelles, comme c'est le cas par exemple en Géologie ou en Médecine. Le système de modélisation proposé permet d'étendre le domaine de représentation en prenant en compte à la fois les conditions manifold et non-manifold.

Les objets utilisés sont représentés par des facettes triangulaires. Une surface est divisée en plusieurs morceaux de triangles connectés appelés faces. Alors qu'une surface peut être non-manifold, une face est toujours manifold. En fait, par définition, une condition non-manifold apparait seulement aux frontières d'une face.

Ce système de modélisation consiste à découper l'espace 3D en plusieurs volumes distincts fermés (régions), définis par leurs frontières (shells). Afin de déterminer les faces adjacentes composant la frontière d'une région, il faut introduire des structures spécifiques décrivant les relations d'adjacence entre les faces. Plus précisément, la topologie du modèle apparait à deux niveaux: la Macro-Topologie décrit les adjacences entre faces, la Micro-Topologie décrit les adjacences entre triangles. Les structures d'adjacence permettent de détecter et définir automatiquement toute fermeture d'un volume de l'espace 3D.

Deux méthodes de construction du modèle volumique ont été développées. La première est une méthode interactive utilisant un outil de collage de surfaces, modifiant ainsi la topologie du modèle en créant des relations d'adjacence entre les faces. La seconde méthode est entièrement automatique et consiste à créer les structures d'adjacence du modèle en s'appuyant sur sa topologie.