



Habillage de modèles mécaniques

Facettisation temps réel de surfaces implicites

présentée et soutenue publiquement le 11 décembre 2001

pour l'obtention du

Doctorat de l'Université des Sciences et Technologies de Lille

(spécialité informatique)

par

Frédéric TRIQUET

Composition du jury

<i>Présidente :</i>	Sophie TISON	LIFL, Université de Lille 1
<i>Rapporteurs :</i>	Marc NEVEU	LE2I, Université de Bourgogne
	Christophe SCHLICK	LABRI, Université de Bordeaux
<i>Examineurs :</i>	Marie-Paule CANI	GRAVIR, Institut National Polytechnique de Grenoble
	Laurent LUCAS	LERI, Université de Reims
	Éric VARLET	Société SIMEDGE, Lille
<i>Directeurs :</i>	Christophe CHAILLOU	LIFL, Université de Lille 1
	Philippe MESEURE	LIFL, Université de Lille 1

UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE

Laboratoire d'Informatique Fondamentale de Lille — UPRESA 8022

U.F.R. d'I.E.E.A. - Bât. M3 - 59655 VILLENEUVE D'ASCQ CEDEX

Tél. : +33 (0)3 20 43 47 24 - Télécopie : +33 (0)3 20 43 65 66 - email : direction@lifl.fr

Remerciements

Je tiens à remercier Christophe Chaillou pour m'avoir donné la possibilité de réaliser cette thèse. Je le remercie pour le regard frais qu'il a toujours su m'apporter. Ses commentaires m'ont permis de prendre du recul sur mes travaux.

Je remercie du fond du cœur Philippe Meseure pour m'avoir co-encadré. Malgré un emploi du temps inimaginable, il a toujours été présent lorsque j'en ai eu besoin. Ces compétences techniques et ses qualités humaines m'ont souvent permis d'aborder mes problèmes sous des angles différents et m'ont toujours donné envie d'aller plus loin.

Un grand merci à Marc Neveu et à Christophe Schlick pour avoir accepté la tâche ingrate de rapporter ma thèse. Leurs commentaires pertinents m'ont permis d'améliorer ce document.

Merci à Sophie Tison qui m'a fait l'honneur de présider mon jury.

Merci à Marie-Paule Cani, à Laurent Lucas et à Éric Varlet pour avoir accepté d'être examinateurs de ma thèse.

Merci aux membres et ex-membres de l'équipe Graphix : Lol (pour nos conversations sans fin et son aide en toutes circonstances), Caliméro (pour ses conseils et son regard positif sur tout), Doug (pour ses conseils et nos fous-rires), Steph (pour ses cascades et ses gloussements), JP (pour son soutien, «photomaton»!!!), Grinch' (pour sa motivation et ses canards), Sam (pour ses réponses à tout ou presque et pour son coffre de toit!), Juju (pour sa sérénité), HollowMan (pour sa discrétion), Pérux (pour le Maxiclène), Patata (pour sa cafetière), Laure (pour ses réactions à mes blagues), Lux 11-38 (pour sa rigueur et ses conseils), Dual (pour Linux), Greg (pour ses URLs), Coyote et l'Infââme. Quelle équipe complémentaire!

Un grand merci à Annie Dancoisne, Gilles, Alain et les autres pour leur travail qui permet de faire vivre le labo. Merci également aux personnes de l'IUT ainsi qu'aux membres du club d'aïkido.

Merci à mes parents pour leur soutien et pour m'avoir soutenu et donné la possibilité de faire ces huit années d'études supérieures. Merci à tous mes proches pour leurs encouragements.

Enfin, un immense merci à Karine, à qui je dédie cette thèse et sans qui je ne serais sans doute pas parvenu au bout de mes peines. Elle a toujours été présente quand j'en ai eu besoin. Tout ce qu'elle m'a apporté dans mes périodes de doute m'ont tout simplement donné envie d'être meilleur.

50/50, ça me paraît raisonnable...

Table des matières

Introduction	1
1 Les surfaces implicites	5
1.1 Présentation	6
1.1.1 Définition	6
1.1.2 Exemples de surfaces implicites	7
1.1.3 Le contrôle des surfaces implicites	7
1.2 Cas particulier des surfaces implicites à squelette	9
1.2.1 Présentation	9
1.2.2 Le squelette	9
1.2.3 Le blending	9
1.2.4 Les fonctions de potentiel	11
1.2.5 Les fonctions de distance	13
1.3 L’affichage de surfaces implicites à squelette	19
1.3.1 Le lancer de rayons	19
1.3.2 Le rendu discret	21
1.3.3 La facettisation	21
1.3.4 Le placage de textures	34
2 L’habillage de modèles mécaniques	37
2.1 Les simulateurs médicaux pédagogiques	38
2.1.1 Présentation	38
2.1.2 La simulation médicale à but pédagogique	39
2.1.3 Intérêts et problèmes de la simulation pédagogique	42
2.1.4 Classification technique des simulateurs	43
2.2 Des modèles mécaniques	47
2.2.1 Modèles continus	47
2.2.2 Modèles discrets	49
2.2.3 L’habillage	51

2.3	Notre plateforme : ALSPeMe	53
2.3.1	Présentation	53
2.3.2	Le moteur SPORE	58
3	L’Affichage temps réel de surfaces implicites à squelette	65
3.1	Quelques faiblesses des techniques actuelles	66
3.2	Les choix concernant le potentiel	67
3.2.1	La fonction de potentiel	67
3.2.2	Accélération de la sommation	69
3.3	Les améliorations des Marching Cubes	71
3.3.1	Le suivi de surface	71
3.3.2	La table de facettisation	73
3.3.3	La détection des calculs redondants	75
3.4	Résolution des cas de facettisation incorrecte	76
3.4.1	Rappels	76
3.4.2	Une première solution	77
3.4.3	Notre solution	78
3.5	Bilan	81
3.5.1	Besoins en mémoire	81
3.5.2	Utilisation de la cohérence temporelle	85
3.5.3	Les Marching Cubes adaptatifs	86
4	Le contrôle du blending	89
4.1	Exposé du problème	90
4.2	Désactivation du blending	91
4.2.1	Solutions de la littérature	92
4.2.2	Solutions intuitives	96
4.2.3	Notre solution pour les Marching Cubes	99
4.2.4	Limites de la solution	101
4.3	La propagation des déformations	103
4.3.1	Solutions de la littérature	104
4.3.2	Notre solution pour les Marching Cubes	106
4.3.3	Limites de la solution	107
5	Résultats expérimentaux	111
5.1	Présentation et analyse des performances	112
5.1.1	Incidence du nombre de primitives	112
5.1.2	Incidence du nombre de cubes traités	114

5.2	Comparaisons avec d'autres implantations	117
5.2.1	L'implantation de Bloomenthal [Blo94]	118
5.2.2	VTK et GTS	119
5.3	Performances avec contrôle du blending	121
5.4	Quelques captures d'écrans	123
5.5	Utilisation dans SPIC	127
5.6	Librairie et collaborations	130
Conclusion et perspectives		133
Bilan des travaux		135
Perspectives		137
1	Améliorations du contrôle du blending	137
2	Les squelettes continus	137
3	Économies en mémoire	138
4	Évolution de la bibliothèque	138
Bibliographie		141
Annexes		149
A Tables de facettisation		151
A.1	Table "séparation"	152
A.2	Table "fusion"	153
B Vue d'ensemble du logiciel développé au CRAS de Valence (Espagne)		155

Introduction

Depuis une dizaine d'années, le monde de l'informatique a vu sans cesse ses limites repoussées. La puissance en MIPS des ordinateurs a été multipliée par 80, et on a vu apparaître pour le grand public des accessoires autrefois réservés au monde professionnel. Une machine actuelle, équipée d'une carte vidéo performante permet d'afficher des animations fluides composées de plusieurs centaines de milliers de polygones, chose encore inimaginable il y a quelques années.

De nombreux domaines tirent profit de ces progrès : les jeux vidéos en sont même devenus les moteurs et tendent à faire évoluer dans un sens ou un autre certains matériels comme les cartes graphiques ou le matériel sonore. À côté de cela d'autres disciplines, générant nettement moins de profits¹, bénéficient elles aussi des avancées technologiques. Je citerai entre autres la CAO, l'architecture (on peut maintenant bâtir virtuellement sa future maison et se promener virtuellement à l'intérieur, ou visiter des constructions détruites comme le temple de Karnak), et la simulation d'environnements virtuels.

En particulier, cela fait presque 10 ans que l'équipe GRAPHIX du LIFL de Lille travaille sur la simulation appliquée au domaine médical : notre but est la réalisation de simulateurs pédagogiques d'interventions chirurgicales au même titre qu'il existe des simulateurs de vol pour les pilotes d'avions. Grâce à ces simulateurs, il est offert aux chirurgiens d'apprendre des gestes opératoires sans risque.

Une grande part de ces simulateurs nécessite l'utilisation d'animations de corps basées sur la physique. Il est alors nécessaire de *modéliser* les propriétés mécaniques de ces corps. Ces modèles mécaniques doivent être complétés par des modèles géométriques, les premiers servant à mettre en place les lois de la dynamique afin de calculer les mouvements des différents corps mis en jeu, et les seconds permettant la visualisation des premiers. On doit donc bâtir la géométrie à partir des informations fournies par la mécanique. Ce procédé est appelé l'«habillage d'un modèle mécanique».

La mécanique et la géométrie étant découplées, mon travail a consisté à établir le lien permettant de fournir un affichage suffisamment réaliste tout en n'ayant pour seule information que l'état courant d'un modèle mécanique le plus simple possible. Ce modèle mécanique doit être simplifié pour diminuer et fiabiliser les calculs de la dynamique. A contrario, la représentation graphique des corps doit pouvoir être détaillée et de bonne qualité. À cet effet, nous avons utilisé des surfaces implicites à squelette. Ces surfaces présentent la particularité de satisfaire deux contraintes opposées. En effet, le *blending* donne la possibilité d'obtenir des formes complexes à partir d'une description initiale simple. L'obstacle principal à leur utilisation est la difficulté que l'on a à les afficher rapidement, que cet affichage utilise ou non les capacités des matériels

1. L'industrie du jeu vidéo est la deuxième mondiale en terme de bénéfices

graphiques actuels.

Les méthodes d’affichage sont trop lentes dans notre contexte de «temps réel» où au moins 25 images doivent être produites à chaque seconde. C’est pourquoi nous avons étudié et implanté une méthode d’affichage permettant de facettiser rapidement des surfaces implicites à squelette. Nous nous sommes basés sur l’algorithme des Marching Cubes, connu pour être très lent, mais notre implantation a été enrichie de plusieurs améliorations.

De plus, toujours dans le cadre des Marching Cubes, nous avons tenu à contrôler le *blending* en se donnant la possibilité de préciser les portions de surfaces où la fusion des surfaces implicites peut ou non avoir lieu.

Notre implantation en C++ a été réalisée sous la forme d’une librairie dont les performances et la simplicité lui ont valu d’être utilisée dans d’autres laboratoires de recherche français et étrangers.

Dans le premier chapitre, nous présentons les surfaces implicites et décrivons les différentes méthodes permettant d’en obtenir un affichage. Le second chapitre situe le contexte de notre étude en introduisant les simulateurs chirurgicaux. Nous y décrivons également les principaux modèles mécaniques et géométriques et la plateforme logicielle élaborée au LIFL. La notion d’habillage est alors amenée naturellement. Les améliorations que nous apportons à l’algorithme des Marching Cubes sont décrites au troisième chapitre et notre méthode permettant de contrôler le *blending* des surfaces est exposée au chapitre suivant. Enfin nous présentons des résultats en terme de rapidité principalement et certaines collaborations mises en place durant ma thèse.

Chapitre 1

Les surfaces implicites

Sommaire

1.1	Présentation	6
1.1.1	Définition	6
1.1.2	Exemples de surfaces implicites	7
1.1.3	Le contrôle des surfaces implicites	7
1.2	Cas particulier des surfaces implicites à squelette	9
1.2.1	Présentation	9
1.2.2	Le squelette	9
1.2.3	Le blending	9
1.2.4	Les fonctions de potentiel	11
1.2.5	Les fonctions de distance	13
1.2.5.1	Cas des primitives discrètes	13
1.2.5.2	Cas des primitives continues	14
1.3	L’affichage de surfaces implicites à squelette	19
1.3.1	Le lancer de rayons	19
1.3.2	Le rendu discret	21
1.3.3	La facettisation	21
1.3.3.1	Les Marching Cubes	21
1.3.3.2	Les méthodes à base de graines	28
1.3.3.3	Les Marching Triangles	32
1.3.4	Le placage de textures	34

Introduction

Dans cette partie nous décrivons les surfaces implicites et plus particulièrement nous nous attardons sur une sous-famille que sont les surfaces implicites à squelette. Le principal frein à leur utilisation dans le domaine de l'animation et plus particulièrement de l'animation temps-réel est la difficulté à les afficher. Nous dressons donc un état de l'art des différentes méthodes permettant d'obtenir un affichage de ces surfaces.

Remarquons dès maintenant que ces méthodes sont applicables dans le cas général de surfaces implicites «classiques» mais que notre étude se limite à l'utilisation de telles méthodes pour le cas particulier des surfaces à squelette.

1.1 Présentation

1.1.1 Définition

Une surface implicite est définie par deux entités :

- une fonction F qui à tout point de l'espace associe une valeur scalaire ($F : \mathbb{R}^3 \rightarrow \mathbb{R}$);
- une valeur scalaire arbitrairement fixée (appelée ici *seuil*, $seuil \in \mathbb{R}$).

On obtient alors une surface (ou plusieurs surfaces disjointes) en considérant l'ensemble des points S de l'espace pour lesquels la fonction est égale à la valeur scalaire fixée :

$$S = \{P = (x_p, y_p, z_p) \in \mathbb{R}^3 / F(x_p, y_p, z_p) = seuil\} \quad (1.1)$$

La valeur scalaire présentée ici va permettre de déterminer la surface de niveau que l'on veut considérer : un point de potentiel supérieur sera à l'intérieur de la surface tandis qu'un point de potentiel inférieur sera à l'extérieur. C'est pourquoi nous nommons ce scalaire le *seuil* de la surface, il est parfois appelé *isovaleur* et l'on parle alors d'*iso-surface* pour la surface obtenue.

Remarque : *les fonctions utilisées étant continues, les surfaces obtenues le sont aussi. Ces surfaces forment donc un partitionnement de l'espace en deux parties : la première où $F > seuil$ et la seconde où $F < seuil$, ces deux zones sont séparées par la surface.*

Par extension du cas particulier des surfaces fermées, nous appellerons respectivement intérieure et extérieure ces deux parties.

Une propriété intéressante qui nous sera utile par la suite, lorsque l'on fournira un éclaircissement aux surfaces, est qu'en tout point (x,y,z) de la surface, un vecteur normal \vec{N} est donné par le gradient de la fonction implicite en ce point :

$$\vec{N}(x,y,z) = \pm \overrightarrow{\text{grad}} F(x,y,z) \quad (1.2)$$

1.1.2 Exemples de surfaces implicites

On pourra citer l'expression sous forme implicite de surfaces courantes comme :

– le plan :

$$F(x,y,z) = ax + by + cz + d \quad (\text{avec } (a,b,c) \neq (0,0,0))$$

$$\text{Plan} = \{ P = (x_p, y_p, z_p) \in \mathbb{R}^3 / F(x_p, y_p, z_p) = 0 \}$$

dont un vecteur normal a pour coordonnées (a,b,c) , qui est le gradient de F .

– la sphère centrée en (x_c, y_c, z_c) et de rayon R :

$$F(x,y,z) = (x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2$$

$$\text{Sphère} = \{ P = (x_p, y_p, z_p) \in \mathbb{R}^3 / F(x_p, y_p, z_p) = R^2 \}$$

– et d'une manière plus générale, la quadrique :

$$F(x,y,z) = (x \ y \ z \ 1).Q.(x \ y \ z \ 1)^T$$

$$\text{Quadrique} = \{ P = (x_p, y_p, z_p) \in \mathbb{R}^3 / F(x_p, y_p, z_p) = 0 \}$$

où Q est la matrice $4 * 4$ qui caractérise la quadrique.

1.1.3 Le contrôle des surfaces implicites

Les formes obtenues au moyen des surfaces implicites sont nombreuses. Il suffit de modifier la fonction F pour déformer la surface. L'exemple simple du «Bump-mapping» proposé par Blinn dans [Bli78] qui consiste à perturber les normales à la surface permet de donner l'impression que la surface est déformée. Dans [Coo84], Coons a proposé le «Displacement mapping» pour perturber réellement la surface. Ce type de déformation utilisant une fonction $D : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, il est facile de perturber une surface implicite en composant D et $F : F' = F \circ D$.

D'autres déformations proposées par Alan Barr [Bar84], permettent d'effectuer des torsions, des fuselages, etc, et sont directement applicables sur les surfaces implicites.

Avec ce qu'il nomme les *F-Rep*², Alexander Pasko [PASS95, SP98, HPF] s'intéresse aux formes que l'on peut obtenir en agissant directement sur l'expression des fonctions F . Dans ses F-Rep, un objet complexe est défini par une fonction continue de plusieurs variables (les coordonnées des points dans l'espace) et à valeurs réelles. La représentation d'objets peut être construite en appliquant différentes opérations à des objets «primitifs». Une primitive est considérée comme «une boîte noire» avec sa fonction de définition donnée par un procédé d'évaluation. Pasko utilise un riche ensemble d'opérations permettant de combiner les fonctions dans le but d'obtenir comme résultat une fonction à valeurs réelles : opérateurs ensemblistes, produits cartésiens (pour augmenter la dimension de l'espace de travail), projections (pour réduire cette dimension), mélange, métamorphose (interpolation entre objets), etc. Les F-rep sont un modèle très général qui englobe et unifie les modèles géométriques suivants : les surfaces implicites, les surfaces de convolution, les surfaces de distance, les CSG, les champs scalaires, etc.

Les surfaces construites par Éric Ferley sont stockées sous la forme de champs scalaires discrétisés et éditées au moyen d'outils de sculpture virtuelle [FCG00]. Il n'utilise pas de fonction pour définir ses surfaces mais des «outils» de sculpture permettant d'ajouter ou de retirer de la matière en n'importe quel point de l'espace.

Benoît Crespin procède à des extrusions de champs implicites pour construire des surfaces complexes (des formes torsadées, taraudées, etc) [Cre98].

Bilan

Tous ces travaux sont très différents les uns des autres. Cependant, leur richesse et leur diversité montre l'étendue des possibilités qui nous sont offertes pour contrôler les formes des surfaces implicites.

Ceci laisse supposer que le contrôle et la manipulation de ces surfaces ne sont pas toujours intuitifs. C'est pourquoi, dans les paragraphes qui suivent, nous nous intéressons à un type particulier de surfaces implicites : les surfaces implicites à squelette.

2. pour *Function representation*

1.2 Cas particulier des surfaces implicites à squelette

1.2.1 Présentation

Ces surfaces ont été introduites par J.F. Blinn [Bli82] en 1982 : son idée est alors de modéliser la densité électronique dans une structure moléculaire grâce aux fonctions implicites. Il introduit la notion de primitive et approxime cette densité par une gaussienne :

$$f_i(x,y,z) = b_i \cdot e^{-a_i \cdot D_i^2(x,y,z)} \quad (1.3)$$

où D_i est la distance entre le point de coordonnées (x,y,z) et la $i^{\text{ème}}$ primitive.

1.2.2 Le squelette

Le *squelette* de la surface est composé d'un ensemble de primitives géométriques. Blinn a utilisé des primitives ponctuelles mais d'autres primitives ont également été proposées par la suite comme des portions de droites ou de plans voire des courbes et même des objets encore plus complexes comme des surfaces ou des volumes (voir la figure 1.1).

Dans la majeure partie des cas, et en particulier dans le contexte du rendu temps réel, on préfère se cantonner à l'utilisation de primitives géométriques simples (points, et éventuellement segments de droites) afin de simplifier les calculs de distances que nous allons présenter un peu plus loin.

1.2.3 Le blending

Lorsque le squelette de la surface est composé de plusieurs primitives, le potentiel d'un point est obtenu en effectuant la somme des contributions de chacune d'elles :

$$F(x,y,z) = \sum_1^N f_i(x,y,z) \quad (1.4)$$

Ceci a pour conséquence de mélanger les surfaces engendrées par chacune des primitives. Cette propriété est appelée le *blending*³ que l'on peut traduire par le *mélange* ou la *fusion*.

La forme de la fonction de potentiel influe sur la manière dont les surfaces vont se mélanger.

3. en anglais *to blend* signifie mélanger, fusionner

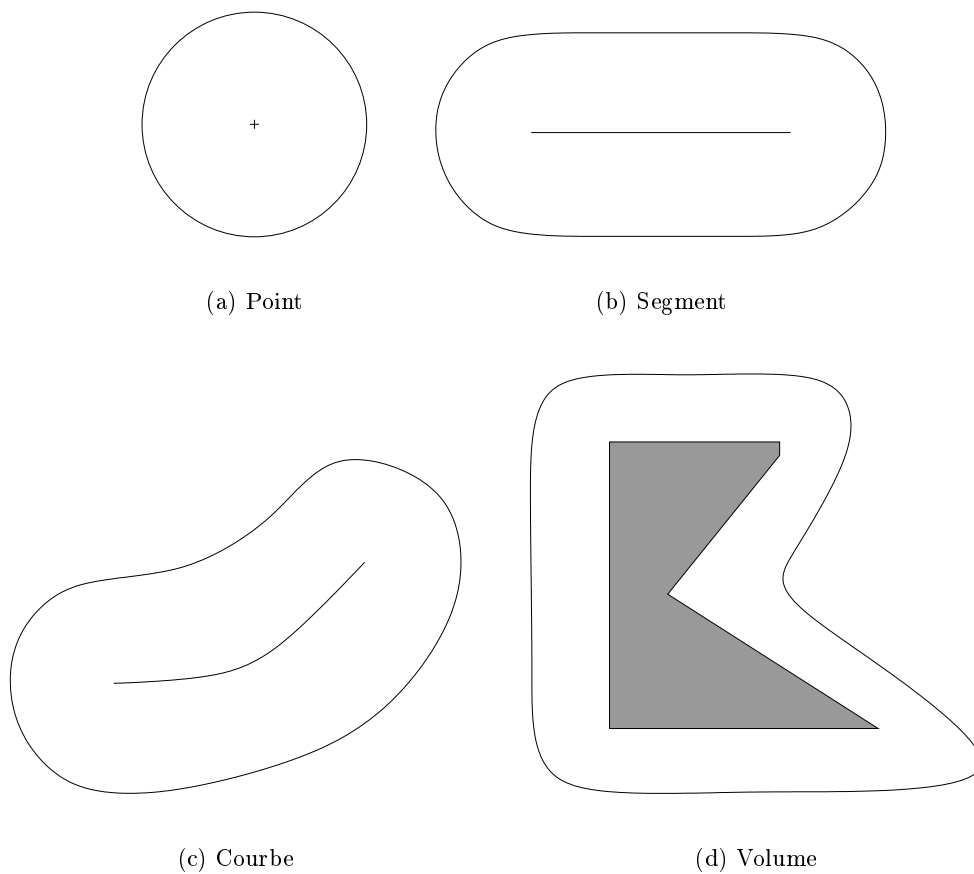


FIG. 1.1 – *Exemples de squelettes et des surfaces engendrées*

1.2.4 Les fonctions de potentiel

Blinn utilise une fonction gaussienne (équation 1.3) car c'est la fonction qui sert à mesurer la densité électronique dans une structure moléculaire. D'autres fonctions tendant à approximer plus ou moins cette gaussienne ont été proposées [WMW86b, BS95, Gas93, CS98, HDH98], mais les points communs que l'on peut leur trouver sont les suivants :

- la fonction atteint sa valeur maximale pour une distance nulle ;
- elle est monotone ;
- elle décroît en fonction de la distance ;
- elle ne présente pas de discontinuité.

Certaines fonctions présentent la particularité d'être identiquement nulles au-delà d'une certaine distance, on parle alors de primitive à rayon d'action fini, dans le sens où cette primitive n'influence le potentiel des points que dans une zone finie de l'espace. Alors que d'autres fonctions ne vont que tendre vers 0^+ (c'est le cas de l'équation 1.3), on parle ici de fonctions à rayon d'action infini.

L'allure générale de la fonction de potentiel influe sur la manière plus ou moins douce avec laquelle les surfaces fusionnent : plus la pente de la courbe sera douce et plus le blinding sera progressif. A contrario une fonction «tout ou rien» (créneau) donnera des surfaces qui s'interpénètrent sans qu'il y ait eu de blinding auparavant. Plusieurs fonctions de potentiel sont rassemblées dans le tableau 1.2. On y trouve aussi la représentation graphique de la fonction et une image du blinding obtenu avec deux primitives ponctuelles. Ce tableau présente deux fonctions à rayon d'action infini et trois fonctions à rayon d'action fini. La dernière fonction est celle que nous utilisons et que nous présentons au paragraphe 3.2.1.

Le point fort du blinding est qu'étant donnée la forme de la fonction de potentiel utilisée, la surface qui en résulte est lisse, et ne présente pas de discontinuité particulière (en particulier les surfaces obtenues ne présentent pas de contour anguleux). C'est cette particularité qui a fait le succès des surfaces implicites à squelette dans le monde de l'animation. En effet, la modélisation, tout en restant très intuitive, permet de construire des surfaces complexes même en n'utilisant qu'un faible nombre de primitives. De plus, le fait que les surfaces se mélangent en formant des surfaces «lisses» est un avantage considérable pour la modélisation d'organes tels que ceux que l'on veut représenter dans les simulateurs médicaux (voir chapitre suivant).

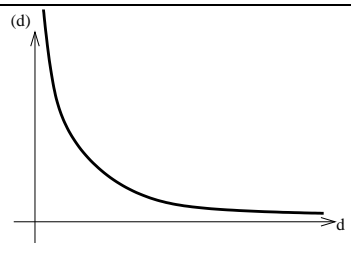

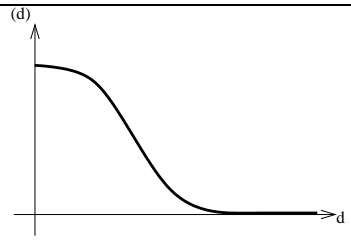

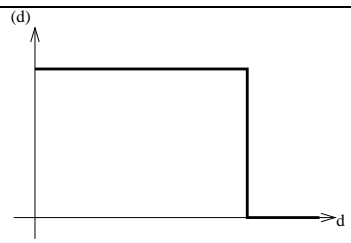

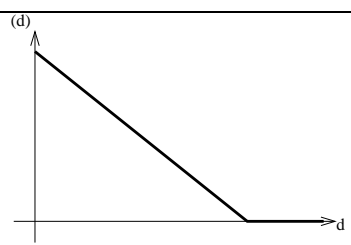

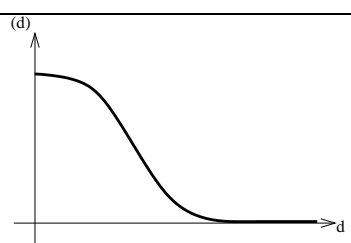

	Fonction	Courbe	Blending obtenu
Rayon d'action infini	$F(d) = 1/d$		
	$F(d) = b.e^{-a.d^2}$		
Rayon d'action fini	$F(d) = (d \leq 1)?1/2 : 0$		
	$F(d) = (d \leq 1)?1 - t : 0$		
	$F(d) = (d \leq 1)?a.(1 - d^2)^2 : 0$ ([MI87])		

FIG. 1.2 – Différentes fonction de potentiels et le blending obtenu

Durant une animation où deux objets s'approchent l'un de l'autre, fusionnent ensemble puis s'éloignent, on constate que le volume global des objets varie. En effet, lorsqu'ils sont suffisamment proches pour que leurs surfaces fusionnent, les déformations dues au *blending* les font grossir. Le volume de l'objet obtenu est supérieur à la somme des volumes des deux objets initiaux. Mathieu Desbrun s'est intéressé au contrôle du volume d'objets lors de telles animations et propose dans [DG95, Des97] de calculer le volume des objets puis d'augmenter ou au contraire de réduire la taille de certaines primitives afin de conserver un volume constant.

Parfois le *blending* n'est pas désiré, or il est automatique dans la formulation de base de la fonction de potentiel. C'est ce que l'on appelle le *blending involontaire* (*unwanted blending* en anglais) qui se produit lorsque deux objets fusionnent en altérant la structure des corps initialement modélisés. Ce problème est étudié en détail au chapitre 4.

1.2.5 Les fonctions de distance

Pour les surfaces à squelette, l'expression de la fonction F est un peu différente des surfaces implicites générales. En effet pour évaluer F en un point P de l'espace (on dira «évaluer le potentiel de P »⁴), il est nécessaire de calculer la distance entre ce point et la primitive géométrique. C'est cette distance qui sert ensuite à évaluer le potentiel du point par rapport à la primitive.

1.2.5.1 Cas des primitives discrètes

Pour ces primitives, les calculs de distance sont simples et la distance euclidienne est le plus souvent utilisée. Par ailleurs, C. Blanc et C. Schlick [BS95] proposent l'utilisation de fonctions de distances anisotropes :

$$D_i(x,y,z) = \frac{1}{r_i} \cdot \sqrt[n]{|x - x_i|^n + |y - y_i|^n + |z - z_i|^n} \quad (1.5)$$

On remarquera le cas particulier $n = 2$ qui est la distance euclidienne.

4. ce terme «*potentiel*» vient de Blinn [Bli82] qui modélisait des potentiels électriques au moyen de surfaces, ces surfaces ont alors été appelées «*surfaces implicites équipotentielles*»



FIG. 1.3 – Objets calculés avec une distance D^n (a) $n=1.5$, (b) $n=3$, (c) $n=\infty$, images issues de l'article [BS95]

1.2.5.2 Cas des primitives continues

Soit S le segment d'extrémités A et B , on a :

$$S(t) = (1 - t).A + t.B \quad \text{avec } t \in [0,1] \quad (1.6)$$

L'évaluation du potentiel d'un point P de l'espace par rapport à ce type de primitive pourra être vu sous deux angles différents.

1. Le calcul du potentiel par rapport au point le plus proche.

Ceci revient donc à calculer $D_{[AB]}(P)$ la distance minimale entre le point P et le segment $[AB]$. Il est alors nécessaire de calculer la coordonnée $t_{P'}$ dans le système paramétrique du segment (équation 1.6) pour P' le projeté orthogonal de P sur la droite (AB) . $t_{P'}$ est donné par :

$$t_{P'} = \frac{(P - A).(B - A)}{\|B - A\|^2} \quad (1.7)$$

La distance $D_{[AB]}(P)$ entre P et le segment $[AB]$ se résume alors à trois cas :

- Si $0 < t_{P'} < 1$ alors $D_{[AB]}(P) = \|P - P'\|$
- Si $t_{P'} \leq 0$ alors $D_{[AB]}(P) = \|P - A\|$
- Si $1 \leq t_{P'}$ alors $D_{[AB]}(P) = \|P - B\|$


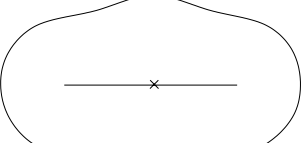
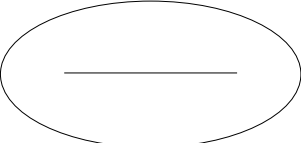
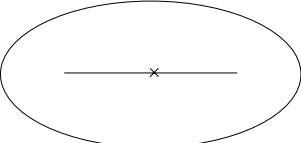
L'évaluation du potentiel de P utilise alors cette distance comme dans le cas des squelettes ponctuels. On appelle ces surfaces des «*surfaces de distance*».

2. La sommation des potentiels par rapport à chaque point du segment.

Le segment étant continu, cette sommation va donc être une intégration du potentiel le long du segment. C'est en fait une convolution du squelette par la fonction de potentiel.

Ce type de surfaces a été proposé par Bloomenthal et Shoemake dans [BS91] sous le nom de «*surfaces de convolution*». Ces surfaces présentent une propriété particulièrement intéressante : comme la convolution est un opérateur linéaire, la convolution de la somme et la somme des convolutions des squelettes sont équivalentes. Une conséquence très concrète à cette propriété est que le segment peut être scindé en deux sous-segments sans altérer la surface obtenue.

Le tableau 1.2.5.2 présente des exemples de surfaces obtenues en considérant les deux évaluations de potentiel pour les cas où le segment est composé d'un (première colonne) ou de deux morceaux (deuxième colonne). Dans [Blo95], Bloomenthal définit le phénomène de «*gonflement*» (qu'il nomme en anglais *bulge*). Cette déformation qui ne se produit que pour les surfaces de distance est due à la superposition ou à la proximité de plusieurs primitives lorsque l'on somme⁵ les potentiels obtenus (première ligne du tableau). Les surfaces de convolution, obtenues par intégration du potentiel sur l'ensemble du squelette ne sont donc pas sujettes à ce type de problème.

	1 segment	2 segments
distance minimale (surfaces de distance)		
intégration (surfaces de convolution)		

Elles ouvrent donc la porte à l'utilisation de courbes dont la construction présente un mécanisme de subdivision. C'est par exemple le cas des courbes de Bézier dont l'algorithme de De Casteljau (voir la figure 1.4 et [FDFH90]) construit récursivement les points à partir des points de contrôle. D'une manière plus générale, comme le proposent Marie-Paule Cani et Samuel Hornus dans [CH01], les courbes à subdivision présentées dans [SDS96] peuvent être utilisées comme primitives. Le schéma de subdivision donne la possibilité de considérer le squelette à différents niveaux de détails. La surface (de convolution) construite autour d'un tel squelette pourra donc être vue à différents niveaux de détails selon les besoins de l'utilisateur.

Le procédé de raffinement des courbes à subdivisions se décompose en deux étapes : le dédou-

5. voir le paragraphe 1.2.3 sur le blending

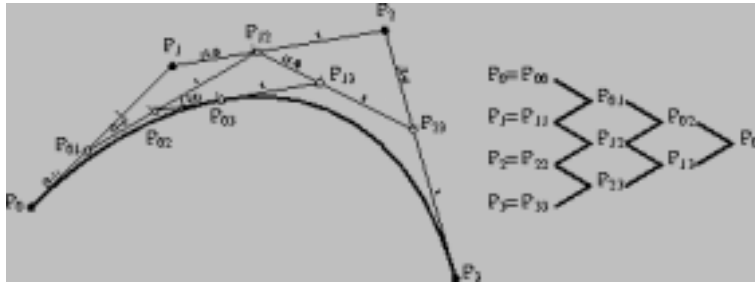


FIG. 1.4 – *Algorithme de De Casteljau pour une courbe de Bézier : les points intermédiaires $P_{i,j}$ sont obtenus à partir des points $P_{i,j-1}$ et $P_{i+1,j-1}$*

blement et le déplacement. De l'étape de dédoublement résulte une liste de points intermédiaires $\overset{\circ}{c}_i^j$. Les points d'indices (i) pairs sont obtenus directement à partir des points du rang (j) précédent alors que les points d'indices impairs sont les milieux des segments du contour polygonal (voir figure 1.5) :

$$\begin{aligned} \overset{\circ}{c}_{2i}^j &:= c_i^{j-1} \\ \overset{\circ}{c}_{2i+1}^j &:= \frac{1}{2}(c_i^{j-1} + c_{i+1}^{j-1}) \end{aligned}$$

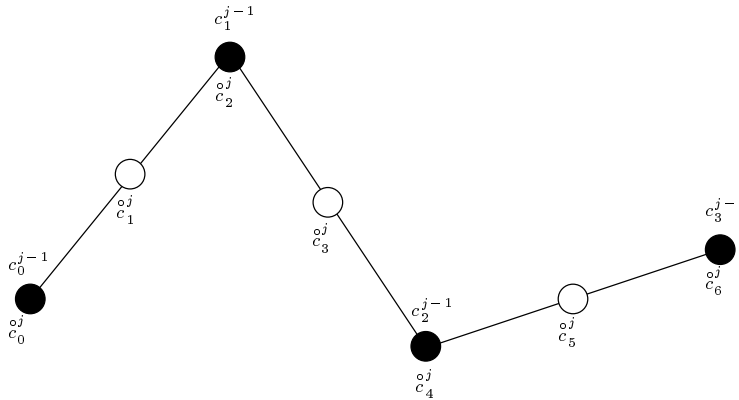


FIG. 1.5 – *Courbes à subdivision, première étape : le dédoublement (les points blancs sont insérés dans le contour polygonal)*

La seconde étape, le déplacement, construit les nouveaux points de contrôle pour le rang suivant :

$$c_i^j = \sum_k r_k \overset{\circ}{c}_{i+k}^j$$

où $r = (\dots, r_{-2}, r_{-1}, r_0, r_1, r_2, \dots)$ est appelé le masque de déplacement. C'est ce masque qui détermine le type de courbe que l'on construit : par exemple $r = (r_{-1}, r_0, r_1) = \frac{1}{4}(1, 2, 1)$ donne une B-Spline cubique (voir figure 1.6).

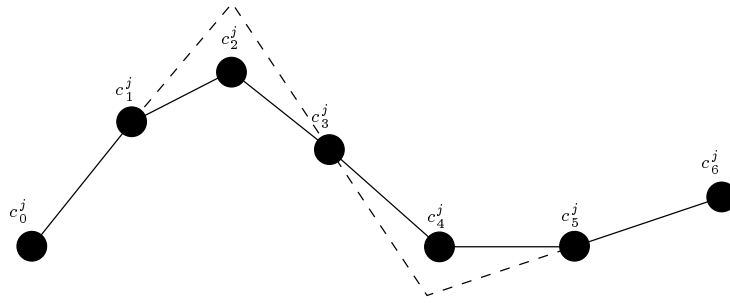


FIG. 1.6 – Courbes à subdivision, seconde étape : le déplacement

Remarque : Dans [Blo90], Bloomenthal propose une approche originale pour construire des embranchements continus entre deux courbes qui se rejoignent. Soit P un point entre les deux courbes (voir figure 1.7a), Bloomenthal propose de construire les projetés orthogonaux Q_1 et Q_2 de P sur chacune des courbes. Il calcule ensuite la distance entre P et le segment $[Q_1Q_2]$ (la distance PP' dans le cas de la figure). Malheureusement, cette méthode est coûteuse à cause des calculs de Q_1 et Q_2 . De plus, l'unicité de ces points n'est pas garantie (figure 1.7b) et la méthode ne fonctionne plus si les courbes sont parallèles. De plus, les embranchements multiples ne se prêtent pas à ces calculs. Par ailleurs, Ferley et al. précisent dans [FCA96] que les calculs ne donnent de bons résultats que lorsque l'angle entre les deux courbes est supérieur à $\pi/2$.

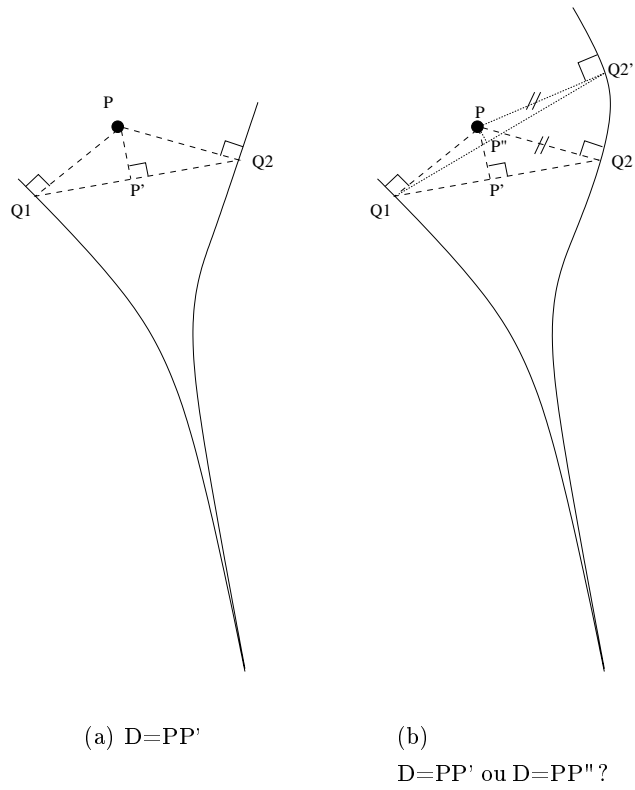


FIG. 1.7 – Distance procédurale de Bloomenthal

Bilan sur les fonctions de distances

Les fonctions de distances utilisées sont donc étroitement liées aux primitives auxquelles elles se rapportent. Dans la suite de ce document, nous ne nous intéressons qu’au cas des primitives discrètes. Néanmoins, généralement peu de modifications sont nécessaires pour prendre en compte des primitives continues.

1.3 L’affichage de surfaces implicites à squelette

Ayant la définition mathématique de la surface, à savoir :

$$S = \{P = (x_p, y_p, z_p) \in \mathbb{R}^3 / \sum_1^N f_i(D_i(x_p, y_p, z_p)) = \text{seuil}\} \quad (1.8)$$

la question de l’affichage se pose alors. En effet pour afficher une surface quelle qu’elle soit on a besoin d’y localiser un ensemble de points. Trouver des points sur de telles surfaces revient donc à trouver des points P de l’espace dont les coordonnées (x_p, y_p, z_p) vérifient l’équation 1.8.

Le problème pour trouver de tels points vient de la complexité de la fonction F : il n’est généralement pas possible de les obtenir de manière immédiate comme pour les modèles paramétriques de type splines. En effet, il s’agit ici de résoudre une équation alors que les modèles paramétriques ne requièrent qu’un simple calcul de fonctions.

Les trois familles classiques de méthodes de rendu se proposent d’effectuer ce travail. Il s’agit du lancer de rayons, du rendu discret (ou rendu *voxels*) et du rendu en Z -buffer utilisant une facettisation des surfaces.

Certains travaux visent à construire une représentation paramétrique à partir de l’expression de F ([GH95] par exemple). Cependant, les auteurs constatent que leur méthode fournit des surfaces ne correspondant pas exactement à la surface implicite initiale, un de leurs buts s’avère être de lisser le champ de potentiel et donc d’éliminer certaines irrégularités de la surface.

1.3.1 Le lancer de rayons

Le lancer de rayons, aussi appelé ray-tracing, permet d’obtenir des images de très haute qualité mais ceci au prix de nombreux calculs effectués pour chaque pixel de l’image à afficher.

Basé sur les lois de l'optique, il consiste à «lancer» un rayon passant par le centre de chaque pixel de l'écran et ayant pour origine l'œil de l'observateur. Des calculs d'intersection permettent de dire si ce rayon rencontre ou non des objets et donc s'il y a lieu d'allumer le pixel concerné. Ces calculs d'intersections consistent à trouver les points se trouvant à la fois sur le rayon (d'équation 1.9) et sur la surface d'équation 1.8 (voir figure 1.8).

$$\begin{cases} x(t) = x_o + x_v * t \\ y(t) = y_o + y_v * t \quad \forall t > 0 \\ z(t) = z_o + z_v * t \end{cases} \quad (1.9)$$

où (x_o, y_o, z_o) est la position de l'observateur et (x_v, y_v, z_v) le vecteur directeur du rayon émis.

$$F(x, y, z) = \text{seuil} \quad (1.10)$$

L'équation à résoudre est donc la suivante :

$$F(x_o + x_v * t, y_o + y_v * t, z_o + z_v * t) = \text{seuil} \quad (1.11)$$

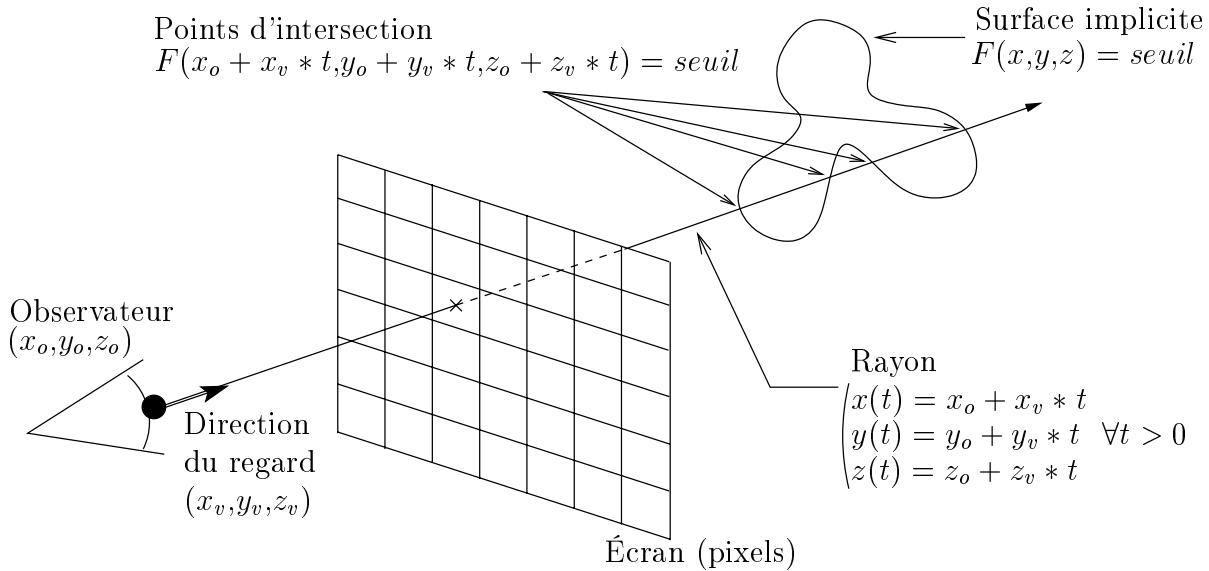


FIG. 1.8 – Lancer de rayons sur surface implicite

De nombreuses travaux s'intéressent à ce problème [Ito89, KB89, Ton89, WT90, Mit90, Har93, Gas95]. Cependant à cause de la complexité de l'équation à résoudre en chaque pixel, le rendu de surfaces implicites par lancer de rayons reste très coûteux en temps de calculs.

1.3.2 Le rendu discret

Il consiste à découper l'espace en petits cubes et à effectuer un lancer de rayons discret sur la grille [Pfi99] ou à projeter les cubes à l'écran [MJC00]. Le rendu discret peut également être mis en œuvre au niveau matériel de machines dédiées à la visualisation de volumes [Kau00, Pfi99].

Concernant les implantations logicielles de cette méthode de rendu, les temps de calculs tendent actuellement à approcher le temps-réel [MJC00]. En effet, certains choix de l'algorithme comme la projection orthogonale permettent d'uniformiser les traitements des cubes et donc de les précalculer.

Cette méthode de visualisation de volumes est généralement utilisée dans le cadre de l'affichage de données médicales (pour lesquelles la projection orthogonale est préférée) où le champ scalaire est *statique*.

1.3.3 La facettisation

Les méthodes de facettisation construisent une approximation d'une surface implicite (à squelette ou non) au moyen d'un maillage polygonal. Nous allons présenter ici les trois catégories d'algorithmes dont le but est de facettiser une surface implicite.

1.3.3.1 Les Marching Cubes

Cet algorithme a été proposé par William E. Lorensen et Harvey E. Cline [LC87] en 1987. Il consiste à découper l'espace selon une grille régulière et à approximer la surface au sein de chacun des cubes ainsi formés.

En chaque cube, on évalue le potentiel des 8 sommets (ce qui se résume dans [LC87] à lire une valeur dans une table). Lorsqu'un cube présente des sommets à l'intérieur de la surface et d'autres à l'extérieur, cela signifie que la surface coupe le cube. On génère alors des triangles en fonction de la position de ces 8 sommets par rapport à la surface.

Il existe 256 configurations possibles que Lorensen et Cline obtiennent en considérant 14 cas de base auxquels ils appliquent des symétries, des rotations et des inversions de polarités⁶. La

6. interchangeant les sommets intérieurs et les sommets extérieurs

figure 1.9 présente ces 14 cas de base.

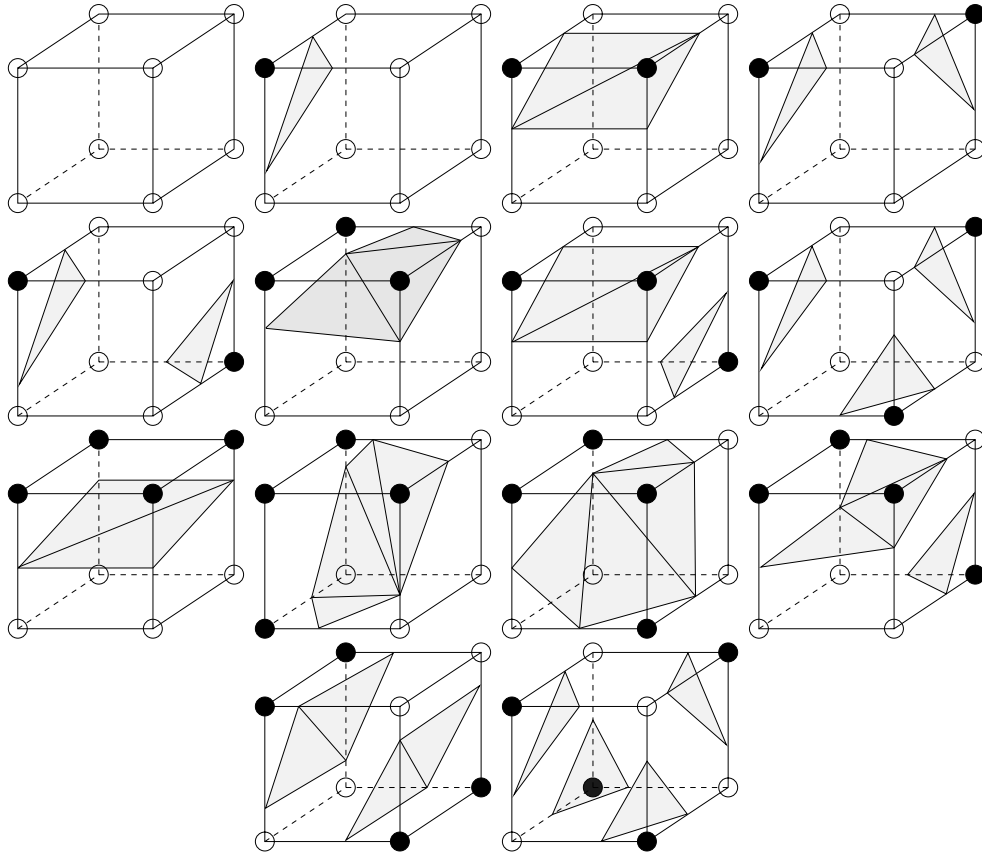
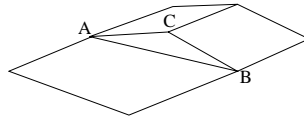


FIG. 1.9 – Les 14 cas d'intersections cube-surface

Les Marching Cubes adaptatifs L'algorithme initial se base sur le découpage de l'espace selon une grille régulière. Il a été proposé dans [Blo88, FS95] de conserver des cubes de taille importante là où la courbure⁷ est faible (en construisant alors des triangles de grande taille), et au contraire d'augmenter la résolution de la grille où les courbures sont importantes. Luiz Velho [Vel90, Vel95, Vel96, VdFG99] propose de facettiser une surface avec une grille régulière de Marching Cubes puis de raffiner le maillage. Certains triangles sont alors subdivisés en fonction de la courbure locale de la surface déterminée par l'angle entre les vecteurs normaux à la surface au niveau des sommets des triangles.

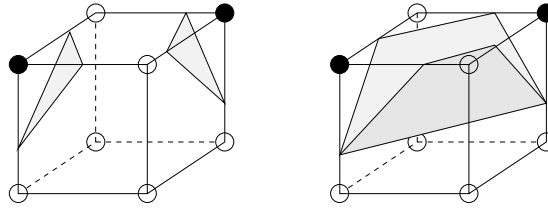
Un problème bien connu lié à cette méthode est présenté en figure 1.10. Il s'agit d'un trou qui se forme entre deux zones facettisées à des précisions différentes. Le cube du dessus ayant été raffiné, le segment $[AB]$ a été scindé en deux sous-segments $[AC]$ et $[CB]$. Le nouveau point C ne

⁷. ne pas confondre «courbure» et «rayon de courbure» qui sont des grandeurs inverses l'une de l'autre

FIG. 1.10 – *Problème trou dans une facettisation adaptative*

se trouve pas forcément sur le segment $[AB]$ et les facettes ne se rejoignent plus. Pour régler ce problème, Luiz Velho propose simplement de ne pas calculer de nouvelle position pour ce point C et de le laisser au milieu de $[AB]$.

Les facettisations ambiguës Un problème inhérent à l'algorithme des Marching Cubes est que la position des 8 sommets d'un cube par rapport à la surface ne suffit pas toujours à déterminer correctement la topologie locale de la surface. La figure 1.11 montre le cas d'un cube pour lequel on peut décider deux facettisations différentes qui conduisent à des topologies différentes : pour la première on obtient deux morceaux de surfaces alors que la seconde aboutit à une surface d'un seul tenant.

FIG. 1.11 – *Configuration ambiguë avec deux facettisations possibles*

La difficulté réside dans le choix de facettisations n'exhibant pas d'incompatibilité topologiques entre des cubes voisins. En effet, pour les cubes voisins (superposés) de la figure 1.12, les facettisations de la figure 1.12a ne coïncident pas et l'on obtient un trou dans la surface, alors que les cubes de la figure 1.12b sont facettisés de manière cohérente et le maillage présente une continuité C^0 . Ning et Bloomenthal [NB93] présentent deux aspects à ce problème :

- *la cohérence* : la facettisation de chaque cube ne produit pas de surface avec des trous ;
- *l'exactitude* : l'approximation construite est en accord avec la topologie de la surface théorique.

[CGMS00] souligne que l'exactitude implique la cohérence : si la géométrie du maillage coïncide avec celle de la surface théorique alors le maillage est nécessairement cohérent.

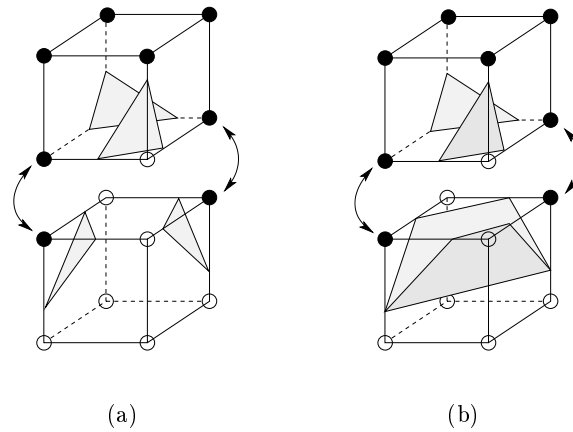
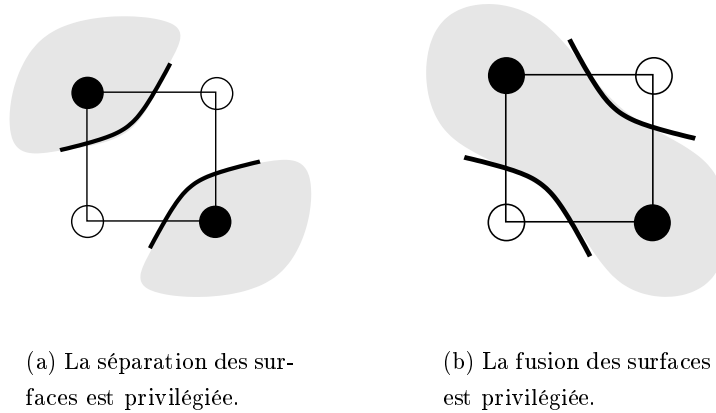


FIG. 1.12 – Deux facetisations différentes peuvent conduire à des problèmes de continuité C^0 dans les maillages obtenus.

Toutes les configurations de cubes ne mènent pas à des indéterminations, seuls les cubes présentant sur la diagonale d'une face deux sommets extérieurs à la surface posent des problèmes. La figure 1.13 montre ce cas et précise que l'un ou l'autre des deux choix a une influence sur la topologie locale de la surface.



(a) La séparation des surfaces est privilégiée.

(b) La fusion des surfaces est privilégiée.

FIG. 1.13 – Ambiguïté au niveau de la face d'un cube

Les trous surviennent donc lorsque l'on choisit de privilégier la fusion des surfaces d'un côté de la face et la séparation de l'autre côté. Pour éviter ce problème il est nécessaire de faire le même choix de part et d'autre de la face concernée pour que les cubes continuent bien de partager non seulement les points d'intersections grille/surface mais aussi les mêmes segments reliant ces

points. C'est ce critère que Ning et Bloomenthal nomment la cohérence.

État de l'art des solutions

Subdivision La première idée naturelle pour tenter de résoudre le problème consiste simplement à subdiviser le cube en des cubes plus petits. Ceci permet d'obtenir un maillage réalisant une meilleure approximation de la surface théorique.

Néanmoins, cette ambiguïté topologique peut alors se poser également dans les «sous-cubes» que l'on considère. On n'a alors pas *résolu* le problème mais on l'a juste *réduit* à une taille inférieure.

Évaluation du potentiel au centre de la face Une autre solution permettant de faire correctement ce choix consiste à choisir l'une ou l'autre des facetisations possibles en fonction du potentiel au centre des faces posant problème. La figure 1.14 montre un cas simple mais dans la pratique cette méthode devient vite complexe à mettre en œuvre lorsque plusieurs faces posent problème en même temps : les 6 faces d'un cube peuvent être ambiguës et l'on obtient alors 64 facetisations possibles pour le cube. Dans [CGMS00], Cignoni et al. proposent une méthode pour réduire cette table exhaustive en traitant de manière spécifique les configurations ambiguës.

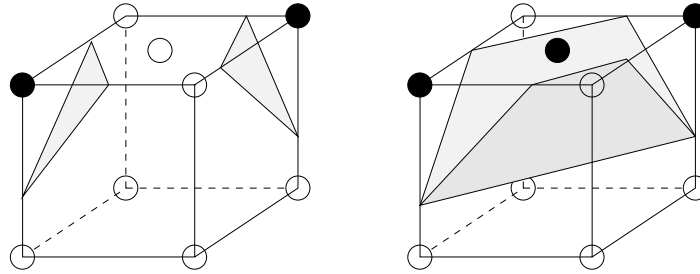


FIG. 1.14 – Facettisations différentes en fonction du centre de la face

Dans certains cas, l'évaluation du potentiel au centre la face n'est pas suffisante pour déterminer correctement la géométrie locale de la surface. La figure 1.15 montre un cas où il serait nécessaire d'affiner plus encore le traitement.

Approximation par hyperbole Une méthode voisine de la précédente consiste à approximer la topologie réelle de la surface au niveau de la face du cube. A. Pasko et al. proposent dans

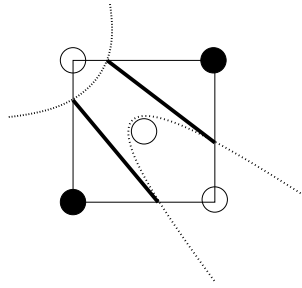


FIG. 1.15 – *Erreur de discrimination avec en pointillés la vraie surface et en traits pleins les segments générés*

[PPP88] d'approximer par une hyperbole la courbe formée par l'intersection entre la surface et la face du cube. C'est la forme de cette hyperbole qui détermine le choix de la facettisation.

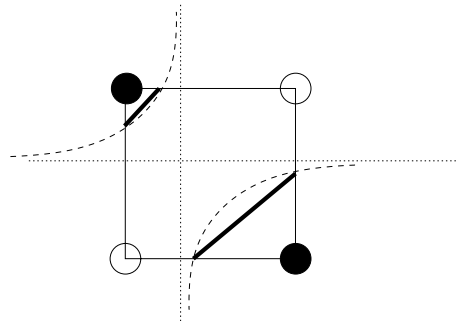
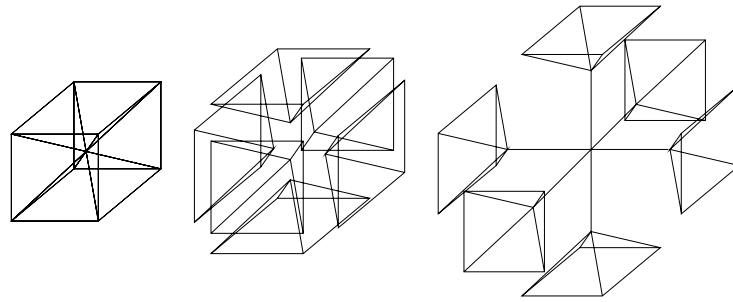


FIG. 1.16 – *Approximation locale de la surface par une hyperbole*

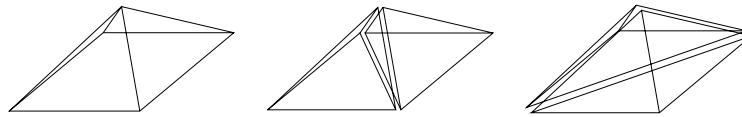
Cette méthode, bien que plus précise que l'évaluation du centre de la face, souffre des mêmes problèmes : elle nécessite certains calculs supplémentaires afin de discriminer les différentes solutions et n'est pas non plus fiable à 100%.

Simplification : décompositions en tétraèdres Bloomenthal décompose ses cubes en tétraèdres [Blo88] afin de lever les ambiguïtés. Le cube est d'abord découpé en 6 pyramides à base carrée et chaque pyramide en deux tétraèdres. Quelle que soit la position des 4 sommets du tétraèdre par rapport à la surface, il n'y a toujours qu'une seule facettisation possible. Cependant on voit sur la figure 1.17b qu'une pyramide peut se décomposer de deux façons différentes en deux tétraèdres (selon la diagonale choisie). Or selon la décomposition choisie on pourra aboutir, pour une configuration donnée, à deux facettisations différentes (voir la figure 1.18), il est donc primordial de décomposer tous les cubes de la même manière. Remarquons qu'il est de plus nécessaire de connaître le potentiel au centre du cube.

Une autre décomposition en tétraèdres est donnée en figure 1.19, elle n'utilise que 5 tétraèdres



(a) Décomposition du cube en pyramides



(b) Décomposition des pyramides en tétraèdres (2 possibilités).

FIG. 1.17 – *Décomposition d'un cube en 12 tétraèdres*

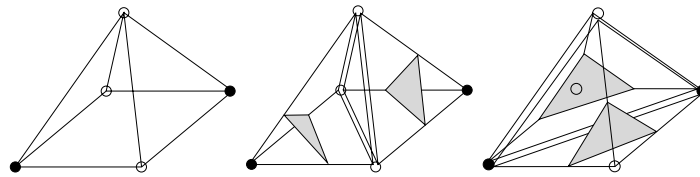


FIG. 1.18 – *Conséquence de la décomposition en tétraèdres sur la topologie de la surface : 2 facettisations différentes pour la même configuration*

et fournit donc moins de triangles. De plus, elle ne nécessite pas l'évaluation du potentiel au centre du cube. Moins coûteuse en calculs et en triangles, elle est cependant moins précise que la précédente et demande aussi de choisir une diagonale de prédilection dans la décomposition.

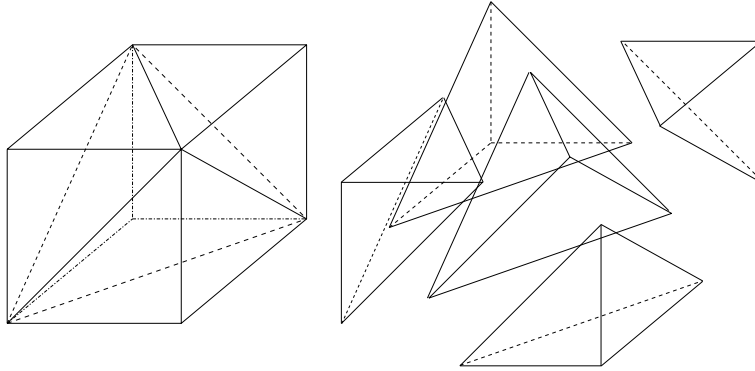


FIG. 1.19 – Décomposition d'un cube en 5 tétraèdres

Bilan des méthodes Toutes les solutions présentées ci-dessus présentent la particularité de nécessiter des calculs supplémentaires et de complexifier de manière importante le procédé de facettisation. Les méthodes de simplicisation présentent en outre le défaut de démultiplier le nombre de triangles construits. Enfin, les méthodes qui tendent à garantir l'*exactitude* du maillage vis à vis de la topologie de la surface théorique peuvent être mises en défaut.

Toutes ces méthodes ont cependant le point commun d'aider à construire des maillages *cohérents* topologiquement, c'est-à-dire présentant une continuité C^0 .

1.3.3.2 Les méthodes à base de graines

Ces méthodes mettent en scène un ensemble de points, appelés *graines*, que l'on déplace pour obtenir un échantillonnage de la surface.

De Figueiredo et al. [dFdMGTV92] proposent d'utiliser un réseau de masses/ressorts afin de construire un ensemble de points reposant sur la surface : ils leur appliquent une force dépendant du gradient de la fonction de potentiel pour que les points du maillage se rapprochent progressivement de leur position d'équilibre qui se trouve (par construction du champ de force) sur la surface. Les forces créées par le réseau de ressorts vont ensuite répartir les masses uniformément sur la surface (une autre force les maintenant sur la surface). Le maillage ainsi positionné fournit directement une triangulation quasi-régulière de la surface. Néanmoins, cette méthode ne peut pas s'appliquer au cas de corps dont la topologie varie au cours de l'animation.

Dans [WH94], Andrew Witkin et Paul Heckbert reprennent le même schéma tout en proposant certaines améliorations en particulier en remplaçant le réseau de masses/ressorts par un système particulaire. Les particules ne subissent alors que des forces de répulsions. Witkin et Heckbert introduisent également la notion de particules fissibles et de taille variable. Lors de déformations, de grosses particules sont utilisées afin de couvrir rapidement les zones sous-échantillonnées. Une fois l’équilibre atteint, ces particules sont subdivisées jusqu’à obtenir la précision d’échantillonnage voulue (voir figure 1.20). Les particules présentes aux endroits où la surface est comprimée se trouvent elles aussi comprimées.

Des problèmes apparaissent lorsque la densité de particules est élevée : les forces de répulsion mises alors en jeu sont très intenses et l’on atteint l’état d’équilibre plus difficilement (puisque la recherche de l’équilibre revient à amenuiser les forces et les vitesses du système). Aussi les auteurs donnent-ils la possibilité à ces graines de disparaître lorsqu’elles sont trop denses, et permettent ainsi de ne jamais avoir des particules trop concentrées. Paul Heckbert propose dans [Hec98] une méthode à base de voxels afin de limiter les calculs d’interactions entre les particules.

L’utilisation de particules indépendantes les unes des autres (contrairement au cas d’un maillage masses/ressorts où elles sont liées), permet de traiter sans difficulté les changements de topologie. En échange, on perd la structure de maillage qui fournissait une triangulation de manière immédiate. L’affichage proposé par Witkin et Heckbert est effectué en traçant des disques tangents à la surface, centrés sur les particules et dont le rayon est la taille de la particule (voir figure 1.20).

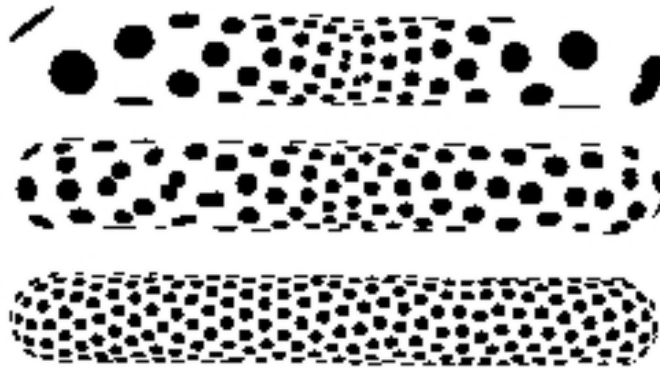


FIG. 1.20 – Représentation des particules au moyen de disques, images issues de l’article [WH94]

Une autre solution consisterait à construire un maillage de triangles à partir du nuage de points obtenus. On a alors recours à une triangulation de Delaunay [For92] qui produit un maillage optimal. La contrainte de Delaunay impose que les sphères circonscrites aux triangles

formés ne peuvent pas englober d'autres points du maillage. Les triangles tendent à être «bien formés» (leurs angles sont maximaux). Malheureusement, cet algorithme a un coût énorme en temps de calcul et ne peut pas être utilisé en temps réel sur des surfaces un peu trop complexes.

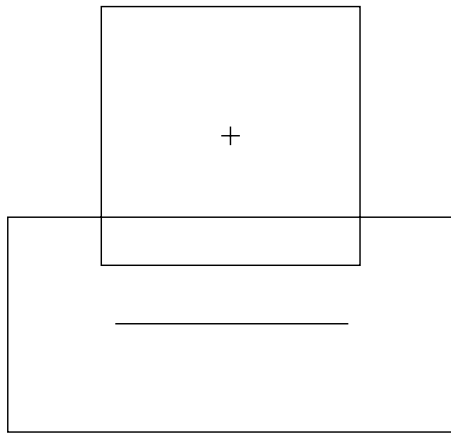
Dans [DTG95], Mathieu Desbrun et al. s'affranchissent de cette triangulation de Delaunay grâce à la méthode qui suit. Ils proposent d'attribuer un ensemble de graines à chaque primitive. Dans un premier temps, ils considèrent chaque primitive isolée des autres. Ils en facettisent la boîte englobante et déplacent les points de ce maillage le long d'une direction particulière (vers le squelette) pour obtenir une triangulation de la surface (figures 1.21a,b et c). La boîte englobante étant ainsi déformée pour épouser la forme de la surface, ceci procure une facettisation propre à chaque primitive.

Un mécanisme d'invalidation des graines est utilisé lors du blending entre les surfaces : les graines présentes dans la zone d'interpénétration des deux surfaces sont ignorées (partie grisée de la figure 1.21e). Deux modes de rendu sont proposés. Le premier ressemble à celui de Witkin : on trace des triangles (nommés «écailles» dans l'article, par analogie aux écailles de poissons) qui permettent d'avoir une vue approximative mais rapide de la surface (un peu à la manière de Witkin et Heckbert qui tracent des disques au niveau des particules). Cette méthode jugée peu satisfaisante pour le rendu de scènes complexes présentant par exemple des surfaces en contact, les auteurs proposent par ailleurs un autre mode de visualisation : une polygonisation par morceau est obtenue en se basant sur la facettisation des boîtes englobant les primitives. Un problème subsiste au niveau des zones d'interpénétrations : des discontinuités sont visibles à cause des graines invalidées car les graines appartenant à des primitives différentes ne sont pas reliées et l'on obtient des trous dans la surface (voir figure 1.21f).

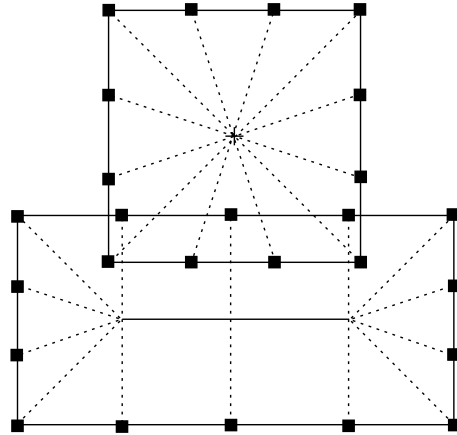
Dans [CGS98] B. Crespin et al. proposent de traiter à part la facettisation de ces zones en construisant des liens entre les bords des facettisations. Ces liens servent ensuite de supports pour une triangulation. Les illustrations et les essais font état d'un petit nombre de primitives (3 ou 4) dont la résolution de la facettisation est élevée (les triangles obtenus sont petits et nombreux). Dans de telles configurations les zones de blending sont très réduites et ce traitement, composé entre autres de recherches du plus proche voisin, peut être effectué sans vraiment être pénalisant.

Dans notre contexte, nous avons pour but de facettiser de nombreuses primitives (100 paraît un minimum afin d'obtenir pour nos organes ou nos flux sanguins une modélisation géométrique digne de ce nom). Dans une telle modélisation, les zones de blending deviennent majoritaires et leur traitement spécifique peut devenir handicapant.

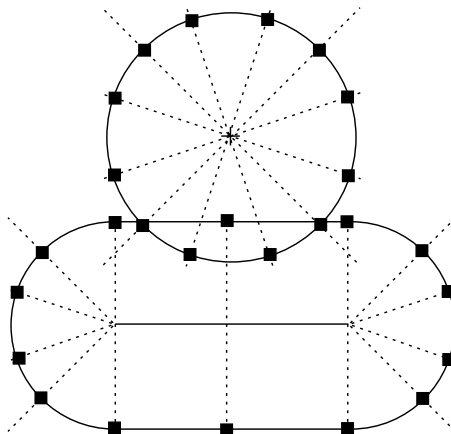
Dans le cas particulier d'objets structurés (dont la topologie ne varie pas au cours du temps),



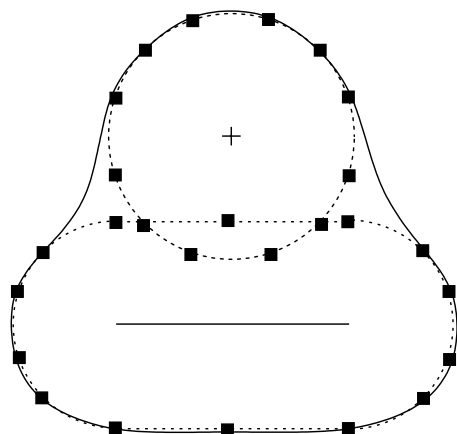
(a) Les primitives et leurs boîtes englobantes



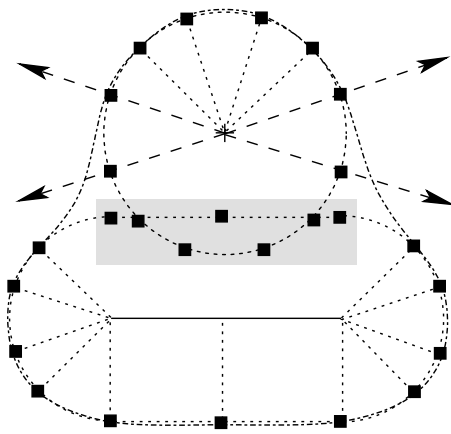
(b) Les graines et leurs axes



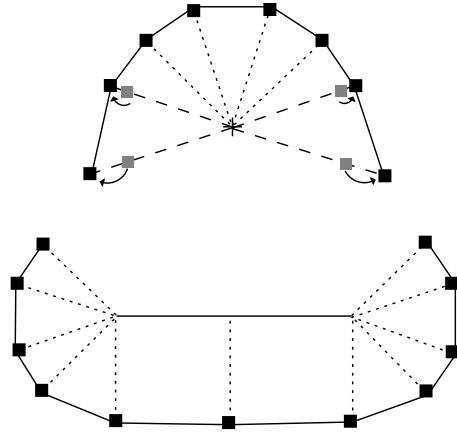
(c) Les graines initialisées



(d) La surface obtenue par blending



(e) Migration et invalidation de graines



(f) Facettisation finale

FIG. 1.21 – Les différentes étapes

il est possible d'adapter la méthode des graines de Desbrun : Marie-Paule Cani et Samuel Hornus proposent dans [CH01] de construire la facettisation d'une boîte englobant les différentes primitives à subdivision composant le squelette. Dans [Ang01], Alexis Angelidis a étendu cette idée en remplaçant la facettisations de boîtes englobantes par une disposition des graines le long de la courbe à subdivision. Il tire ainsi profit de la propriété de multi-résolution de ces courbes en proposant plus ou moins de points (=graines) selon le niveau de détail désiré.

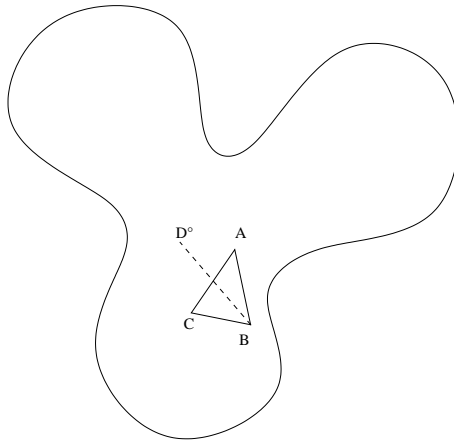
Les auteurs utilisent la cohérence temporelle et c'est donc la position de la facettisation précédente qui est utilisée pour construire la position courante. Néanmoins, la migration des graines depuis leur position initiale vers la surface exacte nécessite l'utilisation de la méthode de Newton-Raphson afin de faire converger chacun des points, ce qui devient très coûteux lorsque le niveau de détail désiré augmente.

1.3.3.3 Les Marching Triangles

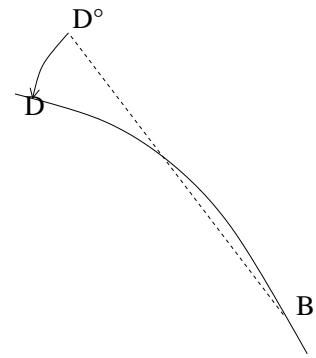
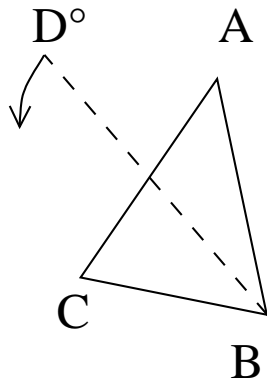
L'algorithme des Marching Triangles proposé par Adrian Hilton et al. [HSIW96] puis amélioré par Éric Galin et Samir Akkouche [GA00, AG01], réalise une triangulation de Delaunay d'une surface implicite en se basant sur la contrainte de Delaunay : *«pour chaque triangle du maillage, la plus petite sphère circonscrite au triangle ne doit contenir aucun autre point du maillage (orienté dans la même direction)»*.

Ils se basent sur un triangle initial (le germe, le triangle ABC sur la figure 1.22a) et construisent un triangle supplémentaire partageant une arête avec le germe. Pour cela, un point D^0 est construit dans le plan du triangle ABC . Ce point est ensuite projeté (par une descente de gradient par exemple) sur la surface afin de donner le point D . Si le triangle ACD satisfait la contrainte locale de Delaunay, on l'ajoute au maillage. Le germe est ainsi enrichi d'un triangle. Le procédé est réitéré jusqu'à recouvrement de la surface.

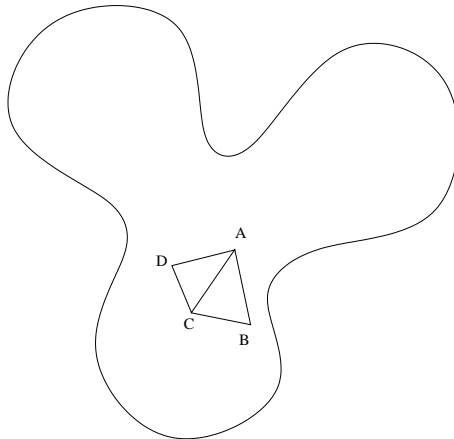
Cette facettisation nécessite moins de triangles que des Marching Cubes et les maillages obtenus tendent à être optimum. De plus la taille des triangles s'adapte à la courbure locale de la surface approximée, ce qui permet d'échantillonner plus finement les zones où la surface est fortement courbée. Cependant les résultats fournis par leurs auteurs font toujours état de performances inférieures à des Marching Cubes. Du point de vue de la complexité, la projection de chaque point sur la surface ainsi que la nécessité de satisfaire la contrainte de Delaunay semblent être deux nœuds importants de l'algorithme.



(a) Surface initiale, le «germe» ABC et le point D^0



(b) Agrandissement de la projection de D^0 sur la surface (c) Projection vue dans le plan (BD^0D)



(d) Ajout du triangle ACD au maillage

FIG. 1.22 – *Les Marching Triangles*

1.3.4 Le placage de textures

Le réalisme des objets peut être accru par l'utilisation de textures appliquées sur les surfaces. Le problème principal réside dans la manière dont on les applique. Chaque sommet du maillage de la surface doit avoir des coordonnées dans l'espace de texture. Ceci est généralement fixé par le graphiste (ou la personne qui a en charge de réaliser le design de l'objet). Dans notre cas, nous nous imposons deux contraintes particulières. D'une part, les surfaces se déforment et les maillages sont très différents d'une étape à l'autre. D'autre part, les surfaces peuvent changer de topologie, se séparer en plusieurs morceaux ou se fusionner en un seul. Il est difficile de déterminer comment texturer ce genre d'objet car dans la mesure où la texture et la surface de l'objet sont étroitement liées, on peut se demander ce que représente pour la texture un changement de topologie de la surface. Certains travaux comme [WMW87, Wyy90, Ak198, Ped95, Ped96] s'intéressent à l'application de textures sur surfaces implicites. La plupart n'est pas destinée au temps réel et ne s'intéresse généralement qu'à des objets qui ne se déforment pas.

Dans notre cas, la connaissance des vecteurs normaux aux différents points des maillages construits donnent la possibilité d'utiliser des méthodes à base d'environnement mapping. Ces méthodes apportent une solution simple à de nombreux problèmes de calculs des coordonnées de textures et sont pris en charge par les accélérateurs graphiques. De même, dans les récentes évolutions des matériels graphiques, on a vu l'apparition d'outils très puissants comme les *combiners* et les *vertex shaders* qui permettent de manipuler les polygones affichés à l'écran au moyen de petits programmes exécutés directement par le processeur de la carte graphique [LKM01].

Conclusion

Les méthodes de lancer de rayons sont encore bien trop gourmandes en calculs pour être utilisables dans un contexte de rendu temps-réel. Par ailleurs, les optimisations nécessaires pour que le rendu discret approche le temps-réel nécessitent des simplifications comme la projection orthogonale des objets à l'écran. De plus, il semble complexe d'afficher dans la même scène un objet en rendu discret et des objets représentés par exemple sous forme de facettes triangulaires.

À l'heure actuelle, l'affichage temps-réel de surfaces implicites semble donc passer nécessairement par l'utilisation des capacités offertes par les accélérateurs graphiques (en particulier le tracé de triangles et le Z-Buffer). Il est donc impératif de procéder à la facettisation de ces surfaces.

Parmi les méthodes de facettisation, les méthodes à base de graines semblent bien adaptées aux surfaces présentant une structure donnée qui ne varie pas au cours du temps. L'utilisation de certaines méthodes à graines ne se basant que sur un nuage de particules sans structure peut difficilement concilier rendu temps-réel et rendu de qualité.

En outre, la méthode des Marching Triangles ne permet pas encore d'atteindre le temps-réel et semble difficile à accélérer à cause des nombreux calculs qu'elle requiert en particulier pour satisfaire la contrainte de Delaunay.

Pour toutes ces raisons, nous allons nous concentrer sur l'algorithme des Marching Cubes qui, bien qu'il ne soit pas exempt de défauts, présente des points que l'on peut améliorer de manière tangible.

Dans le chapitre suivant, nous présentons le cadre d'application de notre travail à savoir les simulateurs chirurgicaux. Dans ceux-ci, les corps représentés peuvent non seulement bouger mais aussi (et surtout !) se déformer, voire changer radicalement de topologie comme c'est le cas pour les écoulements de liquides. Il est impératif que ces facettisations se fassent rapidement.

Chapitre 2

L’habillage de modèles mécaniques

Sommaire

2.1	Les simulateurs médicaux pédagogiques	38
2.1.1	Présentation	38
2.1.2	La simulation médicale à but pédagogique	39
2.1.2.1	L’apprentissage	40
2.1.2.2	L’évaluation	41
2.1.2.3	La formation continue et le complément de formation	41
2.1.3	Intérêts et problèmes de la simulation pédagogique	42
2.1.3.1	D’autres motivations	42
2.1.3.2	Problèmes de confiance	42
2.1.4	Classification technique des simulateurs	43
2.1.4.1	Simulation de navigation	44
2.1.4.2	Simulation avec déformations	45
2.1.4.3	Simulation avec déformations et retour d’efforts	46
2.2	Des modèles mécaniques	47
2.2.1	Modèles continus	47
2.2.1.1	Les éléments finis	47
2.2.1.2	Les splines dynamiques	48
2.2.1.3	Des modèles à nombre fini de déformations	49
2.2.2	Modèles discrets	49
2.2.2.1	Les modèles à particules	50
2.2.2.2	Les modèles masses-ressorts	50
2.2.3	L’habillage	51
2.2.3.1	Discussion	51
2.2.3.2	Dualité mécanique/géométrie	52
2.2.3.3	Définition de l’habillage	53
2.3	Notre plateforme: ALSPeMe	53
2.3.1	Présentation	53
2.3.1.1	Architecture	54
2.3.1.2	Un exemple d’utilisation d’ALSPeMe: SPIC	55
2.3.1.3	La pédagogie	56
2.3.1.4	La bibliothèque dynamique: cahier des charges	57
2.3.2	Le moteur SPORE	58
2.3.2.1	Présentation	58
2.3.2.2	Choix de l’habillage	61

Introduction

Dans ce chapitre, nous présentons le champ d'application de notre recherche sur les surfaces implicites. Il s'agit des simulateurs médicaux pédagogiques. Cette présentation est nécessaire à la compréhension de notre contexte et aux différentes problématiques liées à l'utilisation des surfaces implicites.

Nous détaillerons successivement ce qu'est la simulation à but pédagogique, différentes motivations et quelques problèmes posés. Suivra ensuite une étude d'ordre technique de différents types de simulateurs. Nous étudierons alors la nécessité de modéliser les propriétés mécaniques des corps participants à la simulation et introduirons la notion d'habillage. Enfin, nous préciserons notre cadre de travail en présentant la plateforme actuellement développée au LIFL.

2.1 Les simulateurs médicaux pédagogiques

2.1.1 Présentation

La simulation informatique appliquée au domaine chirurgical est une voie d'investigation relativement récente. Cependant, d'autres domaines d'application ont bénéficié depuis bien plus longtemps des apports des technologies de la *Réalité Virtuelle* : nous pouvons citer par exemple le pilotage d'avion qui a été la première discipline dans laquelle la simulation a fait son apparition. Peu après, ceci s'est généralisé au pilotage d'autres types de véhicules comme les poids lourds et les véhicules militaires (principalement les chars) pour récemment toucher la médecine et plus particulièrement la chirurgie.

Tout au long de ce chapitre, nous ferons souvent des analogies au pilotage d'avions car la simulation aérienne est bien connue. Sa mise en parallèle permettra de saisir plus facilement l'intérêt de la simulation chirurgicale. Cette dernière s'avérant bien plus riche par exemple au niveau des interactions fournies à l'utilisateur, nous allons en présenter les différences.

Tout comme dans le domaine du pilotage d'avions, les différents modes d'utilisation de la simulation en chirurgie peuvent être classifiés dans les catégories suivantes : l'apprentissage, l'évaluation, la formation continue et le complément de formation.

Le but de ce chapitre n'est pas de dresser une liste exhaustive des simulateurs existant à

l'heure actuelle. C'est un champ d'activités bien trop mouvant et des projets de recherche universitaire ou des produits industriels se créent tandis que d'autres disparaissent très régulièrement. Un bilan dressé à une date donnée n'est pas valable plus de quelques mois. En revanche, nous préférons présenter les différents niveaux de l'apprentissage dans lesquels la simulation peut intervenir.

Nous exposons ensuite quelques autres motivations qui feront progresser les travaux sur les simulateurs chirurgicaux et certains problèmes d'ordre éthique liés à leur utilisation. Enfin, nous présentons une classification des simulateurs en fonction de critères techniques en illustrant nos propos en présentant des exemples caractéristiques ainsi que des simulateurs réalisés ou en cours de réalisation au LIFL.

2.1.2 La simulation médicale à but pédagogique

La simulation à but pédagogique est à différencier d'autres activités connexes telles que la préparation de mission, la réalité augmentée et la télé-intervention. Décrivons brièvement ces trois disciplines afin de ne pas risquer de confusion par la suite.

La préparation de mission doit son appellation au domaine militaire. Les pilotes d'avions de chasse utilisent des simulateurs de vol pour s'entraîner et répéter jusqu'à la perfection certaines missions délicates. Dans le domaine chirurgical, la préparation de mission consiste à s'entraîner sur une opération donnée dans des conditions particulières et spécifiques au patient qui sera opéré. Préalablement mesurées par scanner, IRM ou autre, les caractéristiques physiques du patient auront été fournies au simulateur afin de mettre le chirurgien en conditions réelles. Cette activité est voisine de la simulation pédagogique dans le sens où l'utilisation d'un simulateur est indispensable. Cependant, l'aspect pédagogique est bien différent car à l'issue d'une formation, le chirurgien n'aura été formé qu'à une seule opération. Rien ne prouve qu'il sera apte à opérer d'autres patients de morphologies différentes (ossature, taille des organes, etc), et a fortiori à effectuer d'autres opérations.

La réalité augmentée consiste à fournir des informations per-opératoires au chirurgien. Il s'agit donc d'enrichir ce que perçoit le chirurgien d'éléments per-opératoires normalement imperceptibles. L'outil informatique fournira alors les informations supplémentaires et nécessaires à l'intervention. On pourra donner l'exemple de l'affichage sur un écran de contrôle du contour de certains os ou organes difficilement localisables car cachés par d'autres.

La télé-intervention ou intervention à distance a pour but de séparer géographiquement le patient et le chirurgien afin d'opérer ou d'être opéré sans avoir à se déplacer. Le chirurgien va alors opérer en manipulant des outils reliés à des capteurs de positions. Ces informations seront communiquées à un robot qui les retranscrira sur le patient. Le chirurgien aura un retour visuel par un écran et une caméra filmant l'opération. Il est possible de remplacer le système caméra/écran par un simulateur qui devra retranscrire exactement ce qui se passe du côté du patient. Cette activité en plein essor grâce aux nombreux progrès de la robotique reste cependant conditionnée par la fiabilité du simulateur.

Nous ne nous intéresserons qu'à la simulation à but pédagogique tout d'abord au niveau de l'apprentissage, ensuite de l'évaluation et enfin de la formation continue.

2.1.2.1 L'apprentissage

L'intérêt initial pour la simulation est sans conteste la formation de novices à des techniques, et l'apprentissage de techniques nouvelles ou l'utilisation de nouveaux matériels. Par ailleurs, l'apprentissage ou l'évaluation de techniques nouvelles par des chirurgiens experts peut tirer avantage des apports de la simulation.

L'apprentissage d'actes médicaux et plus particulièrement d'actes chirurgicaux, se fait habituellement par ce que l'on appelle le «compagnonnage». L'élève apprend en observant son professeur qui lui montre les gestes, et tente ensuite de les ré-exécuter.

Cette méthode du compagnonnage est malheureusement très limitée pour plusieurs raisons :

- la présence du professeur est impérative ;
- les risques encourus par le patient sont nombreux ;
- les champs opératoires sont restreints et il est difficile pour l'élève et le professeur d'avoir exactement le même point de vue, d'où des difficultés à échanger des informations lors de la démonstration ou de la restitution par l'élève ;
- lors d'une démonstration, seule une partie du savoir est transmise à l'élève et celui-ci devra tenter de restituer ce qu'il a vu alors qu'il sera en même temps en train de découvrir des choses imperceptibles lors de la démonstration comme le poids des outils, leur maniabilité⁸, etc.

8. on pense ici en particulier à l'endoscopie et aux techniques à invasion minimale

Utilisée dans le cadre de cet apprentissage, la simulation permet à l'élève de se former en toute sécurité. Ce gain en sécurité touche bien évidemment le patient qui n'intervient plus dans la formation, mais aussi le chirurgien qui ne risque plus de se mettre en danger lors de manipulations délicates, ou encore certains matériels coûteux pouvant être endommagés lors d'utilisations anormales.

2.1.2.2 L'évaluation

Les simulateurs, qu'ils soient médicaux ou non, peuvent proposer aux élèves de s'évaluer par rapport à un niveau de compétences requis pour être jugé opérationnel. Il sera alors possible de s'entraîner sur un ensemble d'exercices qui permettront d'attribuer une note à l'élève. Cette note sera un indicateur lui donnant la possibilité de juger de sa propre progression ou pourra même être employée dans la validation de sa formation.

Par ailleurs, il est possible d'évaluer aussi des méthodes opératoires: un chirurgien expert pourra en effet réaliser une opération particulière et juger de l'efficacité d'un nouveau mode opératoire. Ce n'est plus l'élève qui est évalué mais le mode opératoire proposé par le simulateur.

2.1.2.3 La formation continue et le complément de formation

Certaines interventions chirurgicales se font rarement et les praticiens ne sont pas toujours suffisamment «entraînés» à les réaliser. La simulation est alors un outil permettant de se maintenir au niveau de connaissances et d'aptitudes nécessaires au bon déroulement de l'intervention.

L'apprentissage sur patients réels souffre d'une lacune particulière: la confrontation aux pathologies rares est, évidemment, peu fréquente. De ce fait, n'y ayant pas été confrontés durant leur formation, les praticiens ne seront peut-être pas aptes à les déceler lorsqu'ils les rencontreront. Un simulateur peut proposer toutes les pathologies aussi rares soient-elles et offrir ainsi un complément de leur formation très appréciable.

Un problème voisin est la mise en situation d'urgence: lorsqu'un problème grave se pose durant une opération, le professeur reprend immédiatement la main et l'élève ne peut plus intervenir, son apprentissage de ce genre de situation se limite donc à l'observation. Ce genre de problèmes s'inscrit parfaitement dans le cadre de la simulation car ici l'élève pourra aller au bout de la manipulation, et éventuellement faire des erreurs qui seront sans doute plus formatrices qu'une simple observation de ce que pourrait faire le professeur pour venir à bout de l'incident.

L'élève pourra aussi répéter l'incident plusieurs fois et apprendre à le gérer sans jamais mettre en danger la vie d'un patient.

2.1.3 Intérêts et problèmes de la simulation pédagogique

2.1.3.1 D'autres motivations

Les pilotes d'avions ont vite compris les avantages d'un apprentissage sur simulateur, aussi bien du point de vue des coûts financiers occasionnés par des accidents que du point de vue des risques qu'ils encourent à piloter un appareil qu'ils ne connaissent pas. En revanche, les chirurgiens sont plus difficiles à convaincre.

L'apprentissage de la chirurgie débute généralement sur des animaux et sur des cadavres humains. Passé ce stade, on en vient à intervenir, toujours dans le cadre d'un compagnonnage, directement sur des patients réels. Ceci n'est pas sans poser certains problèmes puisque l'intervention de chirurgiens débutants sur des patients réels peut s'avérer dangereuse. Ce type d'apprentissage finira sans nul doute par être interdit.

Il est donc nécessaire de trouver des méthodes d'apprentissage permettant de s'affranchir de l'utilisation d'animaux et de l'intervention de débutants sur des patients.

2.1.3.2 Problèmes de confiance

La simulation chirurgicale n'est pas encore appréciée à sa juste valeur par les chirurgiens. Il est tout à fait compréhensible que certains praticiens puissent être réticents à l'égard des simulateurs tant que ce qui leur est proposé n'est pas suffisamment pertinent. Des progrès techniques sont nécessaires afin de leur proposer des produits vraiment utilisables, ceci fera alors peut-être progresser les mentalités et les chirurgiens accepteront-ils plus volontiers les simulateurs comme un véritable outil de formation.

On se heurtera sans doute encore longtemps à un sujet délicat lorsque l'on abordera l'idée d'évaluer un élève et de valider sa formation au moyen de simulateurs. Tout d'abord pour que cette évaluation ait de la valeur, les simulateurs doivent être extrêmement proches de la réalité, ce qui n'est pas évident étant données les difficultés rencontrées à tous les niveaux de leur conception. Ensuite, quel que soit le niveau de réalisme du simulateur, l'apprentissage et l'évaluation

restent fondamentalement différents d'un travail sur un patient réel. Par exemple, un élève formé exclusivement sur un simulateur devra gérer un stress nouveau lorsqu'il interviendra en conditions réelles. L'évaluation n'est en fait qu'une évaluation par rapport au simulateur et en aucun cas par rapport à une situation réelle où le stress joue effectivement une part considérable.

Une question se pose alors : un chirurgien jugé opérationnel sur un simulateur sera-t-il opérationnel lors d'une vraie opération? À l'heure actuelle, la réponse à cette question dépend du type d'intervention simulée. En effet, certains produits tels que le simulateur PixEyes dont nous reparlerons un peu plus loin, sont d'ores et déjà entièrement utilisables dans le cadre d'une formation (voir [PDR97]). Cependant, c'est encore loin d'être le cas pour des simulateurs où l'opération requiert par exemple des manipulations d'organes. Le but de notre recherche est de réussir à produire des simulateurs d'interventions complexes et dont le réalisme serait suffisant pour que la seule formation sur ces simulateurs rende opérationnel le chirurgien.

Bilan

Pour toutes ces raisons, il y a fort à parier que les mentalités seront difficiles à faire évoluer. Cependant, il reste indiscutable à l'heure actuelle que l'utilisation de simulateurs dans l'apprentissage d'actes chirurgicaux est un atout dont il faut tirer profit, cependant il faudra encore de nombreuses années de recherche, de développement, de progrès techniques, et de collaborations entre les concepteurs de simulateurs et les chirurgiens pour que la formation sur cobaye soit remplaçable (et remplacée !) par l'utilisation de simulateurs. En l'état actuel des choses, la simulation tend à être le meilleur complément de formation mais reste en général à ce stade.

2.1.4 Classification technique des simulateurs

Nous venons de présenter assez brièvement la simulation chirurgicale avec comme but de distinguer les différents niveaux d'applications auxquels on peut l'utiliser dans le cadre de l'apprentissage du geste chirurgical.

Nous allons à présent différencier les simulateurs en fonction de critères techniques. Ceci nous permettra par la suite de situer le niveau auquel notre travail intervient.

Cette classification d'ordre technique passe par une distinction du niveau d'interaction avec l'univers simulé qui est offert à l'utilisateur. Plus ce niveau d'interaction sera élevé, plus les contraintes techniques lors de la réalisation du simulateur seront importantes.

En cela, on peut distinguer trois niveaux d'interaction et donc trois classes de simulateurs. Nous présentons tout d'abord les simulateurs dans lesquels seule une observation de l'environnement est possible que l'on appelle habituellement simulateurs de navigation par analogie aux simulateurs de conduite. Ensuite nous étudierons des simulateurs qui mettent en scène les déformations dues aux contacts des outils du médecin sur les organes simulés, ces simulateurs intègrent une modélisation mécanique permettant de calculer ces déformations. Enfin nous présenterons des simulateurs qui, en plus de prendre en compte des déformations, fournissent à l'utilisateur des informations tactiles grâce à un retour d'effort sur les outils manipulés.

2.1.4.1 Simulation de navigation

Ce type de simulation propose à l'utilisateur de visualiser un environnement graphique correspondant à ce qui serait visible lors d'une phase d'observation. La forte ressemblance à la simulation de véhicule fait que les simulateurs de ce type sont généralement très aboutis et réalistes au même titre que le sont les simulateurs de vol ou de conduite de véhicules.

Contrairement à ce que l'on pourrait penser, ce genre de simulateurs est loin d'être dépourvu d'intérêt. En effet, les simulateurs de navigation permettent de former au diagnostic. Cette phase est essentielle au bon déroulement d'une thérapie car c'est elle qui conditionne tout ce qui sera décidé par la suite (traitement, type d'opération, etc). Ils permettent aussi d'apprendre à manipuler des outils dont l'utilisation n'est pas évidente tels que des sondes échographiques ou des endoscopes. Enfin, de tels simulateurs forment aussi à ce que l'on appelle le repérage anatomique qui consiste à identifier les différentes parties du corps au moyen de l'outil de visualisation utilisé.

Par ailleurs, ce type de simulateur peut aussi permettre de se former à la manipulation d'outils chirurgicaux dont le maniement peut être périlleux. À cet effet a été développé au LIFL un simulateur d'intervention coelioscopique (nommé *SPIC*) dans lequel des pinces de chirurgie sont utilisées d'une manière très différente de l'habitude [JDK97].

Dans cet esprit a été développé au LIFL un simulateur d'*arthroscopie de l'épaule*[ART] dans lequel les modélisations d'une clavicule, d'une omoplate et de la tête d'un humérus sont visualisées. Il sert principalement à reconnaître les différentes parties de l'épaule, tâche rendue difficile (pour le débutant) par l'utilisation d'une caméra de type *fish-eye* qui fournit des images très déformées. Ce simulateur propose aussi d'identifier différentes pathologies (courantes ou rares) ce qui forme les médecins au diagnostic.

Le projet initialement baptisé *Sophocle* et commercialisé par la société SIMEDGE[SIM] sous

le nom de *PixEyes* simule la photocoagulation par laser du fond d'œil[MKC⁺95a, MKC⁺95b]. Il s'agit ici d'un simulateur où l'utilisateur interagit peu avec l'environnement : en plus de déceler des pathologies du fond de l'œil, on peut apprendre à pratiquer une opération au laser nécessaire en cas de décollement de la rétine. Ce simulateur reste malgré tout classé parmi les simulateurs de navigation car les interactions avec l'environnement ne se limitent qu'à des modifications de textures pour marquer l'impact du laser sur le fond de l'œil.

La simulation de navigation appliquée au domaine médical est ancienne. Le premier exemple date du milieu des années 80 et s'intéressait à l'endoscopie [GW87]. Ce produit a été développé durant de nombreuses années [HGW92].

2.1.4.2 Simulation avec déformations

Dans ce type de simulation, un niveau d'interaction supplémentaire est fourni au praticien : il est ici possible d'entrer en contact avec différentes parties de l'environnement. Sous la pression des outils utilisés, les corps concernés vont se déformer et/ou se déplacer.

Cela nécessite une modélisation des comportements dynamiques des différents corps intervenant dans la simulation. Cette modélisation sera plus ou moins fidèle à la réalité selon le niveau de réalisme désiré et la puissance de calcul à disposition. De plus, comme le soulignent Wyvill et al. dans [WMW86a] en distinguant l'«animation» et l'«illusion de l'animation», il n'est pas nécessaire de s'appuyer sur une modélisation des propriétés physiques pour obtenir des animations réalistes.

Le projet *SPAC*⁹[SPA, Wib00] a pour but la simulation de ponction de liquide amniotique. Cette opération se fait en contrôlant par échographie la pénétration de l'aiguille dans les différents tissus abdominaux.

Une modélisation des propriétés mécaniques va permettre par exemple de déformer une partie de l'abdomen en contact avec la sonde échographique. Cette déformation est nécessaire si l'on veut rendre de manière réaliste l'image échographique déformée par la pression de la sonde.

La société SIMBIONIX[SBX] propose des simulateurs d'endoscopie (coloscopie, etc) avec éventuellement du retour d'efforts (que nous allons détailler au paragraphe suivant) au moyen d'un dispositif de type Phantom [Sen].

9. Simulateur Pédagogique d'Amniocentèse et de Cordocentèse

Dans *SPEED*¹⁰ [VCDM97, VC97], la sonde échographique introduite déforme le tube digestif, ici aussi la représentation géométrique de certains organes doit être prise en compte si l'on veut fournir des images réalistes.

2.1.4.3 Simulation avec déformations et retour d'efforts

Dans les différents types de simulateurs présentés ci-dessus aucune information tactile n'est fournie au praticien qui doit alors se fier à des indicateurs sonores ou visuels. Des simulateurs plus évolués fournissent un retour d'efforts sur les outils manipulés dans le but de restituer les mêmes sensations tactiles que dans la réalité.

L'autre partie du simulateur *SPAC* concerne l'enfoncement de l'aiguille : l'extrémité de cette aiguille est fixée à un système à retour d'efforts (un Phantom [Sen]) qui communique la résistance des différents tissus transpercés au manipulateur. L'utilisateur apprend ainsi à faire le lien entre ce qu'il voit sur l'écran de l'échographe et ce qu'il ressent par l'aiguille.

SPIC est un autre simulateur de ce type que nous présentons plus en détail dans la suite.

Jusqu'à présent, la recherche universitaire s'est concentrée sur les modèles [SBM⁺94]. La France est d'ailleurs très active sur ce sujet : Debunne propose un modèle de foie à base d'éléments finis multi-résolution [DDBC99, DDCB01], Cotin quant à lui utilise des masses-tenseurs [CDC⁺96]. Cependant peu de prototypes universitaires ont vu le jour, le prototypage concernant plus le milieu industriel [SBX, XIT].

Par ailleurs, des sociétés comme Xitact [XIT], Kismet [KIS, CK00], HT-Medical récemment rachetée par Immersion [IMM] ou des laboratoires de recherche privée comme le MERL [MER] travaillent activement des produits proposant du retour d'efforts dans la simulation.

Bilan

Nous venons de présenter différents types de simulateurs en fonction de critères techniques dépendant du niveau d'interaction que l'on veut offrir à l'utilisateur.

Parmi les trois catégories de simulateurs, deux se basent sur la physique des corps représentés.

10. Simulateur Pédagogique d'Écho-Endoscopie Digestive

Comme nous l'avons dit précédemment, les caractéristiques de la dynamique des objets simulés doivent être prises en compte pour obtenir des déplacements et des déformations réalistes.

Nous allons à présent introduire différents types de modélisations mécaniques. Puis nous décrirons le procédé qui permet de visualiser un corps que l'on anime, il s'agit de ce que nous nommerons l'*habillage*. Enfin, nous exposons les choix de notre méthode d'habillage. Nous verrons alors que les surfaces implicites présentées au premier chapitre présentent des propriétés très intéressantes pour satisfaire certaines contraintes dictées par les caractéristiques de notre moteur dynamique.

2.2 Des modèles mécaniques

Les corps déformables présentent, dans la réalité, un nombre infini de degrés de liberté. Les modéliser de manière informatique revient à rendre *fini* ce nombre, c'est-à-dire à discrétiser le corps. C'est principalement la manière dont cette discrétisation est effectuée qui caractérise le type de modèle.

Nous allons présenter ici différents types de modèles permettant de représenter des propriétés mécaniques. La classification en deux familles *modèles continus/modèles discrets*, bien que jugée discutable par certains auteurs, a le mérite d'être classique. Mon but n'est d'ailleurs pas de m'inscrire dans une quelconque polémique à ce sujet mais au contraire de dresser un inventaire de différents modèles mécaniques et des méthodes habituellement employées pour en faire l'affichage.

2.2.1 Modèles continus

Le qualificatif «continu» peut sembler contradictoire avec le principe de discrétisation évoqué ci-dessus. Cependant, les modèles dits continus présentent la particularité d'être conçus de manière à prendre en compte l'aspect continu de la matière. Nous ne parlerons ici que des modèles les plus représentatifs.

2.2.1.1 Les éléments finis

La décomposition en éléments finis s'appuie sur le principe des travaux virtuels qui consiste à dire que pour des petits déplacements, le travail des forces intérieures résistantes qui s'opposent

aux déplacements est égal au travail des forces extérieures qui créent ces déplacements. Soit, mathématiquement :

$$\int_{V_e}^t \delta \vec{\epsilon} \vec{\tau} dV_e = \int_{V_e}^t \delta \vec{u} \vec{f}^V dV_e + \int_{S_e}^t \delta \vec{u} \vec{f}^S dS_e + \sum_{i=1}^n \delta \vec{u}_i \vec{F}_i \quad (2.1)$$

où $\delta \vec{\epsilon}$ représente les petites déformations dues aux déplacements $\delta \vec{u}$, $\vec{\tau}$ représente les contraintes internes, \vec{f}^V sont les forces volumiques, \vec{f}^S sont les forces surfaciques et F_i les forces ponctuelles.

La méthode va alors recourir à une décomposition de la structure étudiée en volumes ou mailles élémentaires. Cette équation est intégrée sur le volume en sommant le calcul sur les mailles. Ces mailles peuvent être cubiques ou tétraédriques et comportent un nombre connu de nœuds. La position de chacun des points contenus dans une maille est déterminée à partir des nœuds de la maille par des fonctions d'interpolation.

Une description plus poussée des principes et de l'utilisation des éléments finis sort du cadre de ce manuscrit. On trouvera dans la littérature de nombreux ouvrages traitant de cette méthode, citons par exemple [Bat82]. Par ailleurs, l'application des éléments finis à l'animation est apparue plus récemment dans [Gou94]. Cotin [CDC⁺96] utilise les éléments finis dans la modélisation des propriétés mécaniques d'un foie pour une simulation de coelioscopie.

L'affichage de corps modélisés au moyen d'éléments finis se fait généralement en représentant les mailles élémentaires qui le composent. Cette représentation revient donc à tracer un maillage formé de surfaces paramétriques souvent polynomiales ayant pour support les nœuds de la décomposition. Il est bien souvent nécessaire d'extraire les mailles à la frontière de la structure représentée. Ceci peut poser problème lorsque l'on découpe dans la structure (lors de la découpe d'un organe par exemple), car il faut alors déterminer de nouveau l'ensemble de ces mailles.

2.2.1.2 Les splines dymaniques

L'équipe du LERI à Reims travaille sur l'animation de splines au moyen de la mécanique lagrangienne [RNN00].

Dans le formalisme lagrangien, un système est considéré sous forme de degrés de liberté indépendants. Si on connaît la valeur de tous ces degrés de liberté, le système est alors complètement caractérisé.

Les degrés de liberté d'une spline sont ses points de contrôle. Les propriétés mécaniques s'expriment sous forme énergétique en fonction des vitesses et positions des points de contrôle.

2.2.1.3 Des modèles à nombre fini de déformations

L'idée de base est de soumettre des modèles géométriques continus à un jeu de déformations connues [PW89, WW90]. Par exemple des courbures, des torsions... Ces déformations peuvent avoir une expression complexe mais que l'on connaît, et dont seule l'amplitude est paramétrable. Cette amplitude est déterminée par les lois de la physique. Ainsi, on pourra dire par exemple que l'objet peut être soumis à une courbure dont l'amplitude est quantifiée par les lois de la mécanique afin qu'il s'adapte le mieux à la contrainte qui l'oblige à se courber. L'avantage majeur de cette méthode est de réduire considérablement le nombre de degrés de liberté du système, puisque les seuls paramètres sont les amplitudes du jeu de transformations que l'on se donne (voir un exemple de déformation sur la figure 2.1). Paradoxalement, cette réduction est aussi l'inconvénient du système puisqu'il faudra avoir un jeu suffisamment important de transformations possibles pour obtenir un système ayant un comportement réaliste.

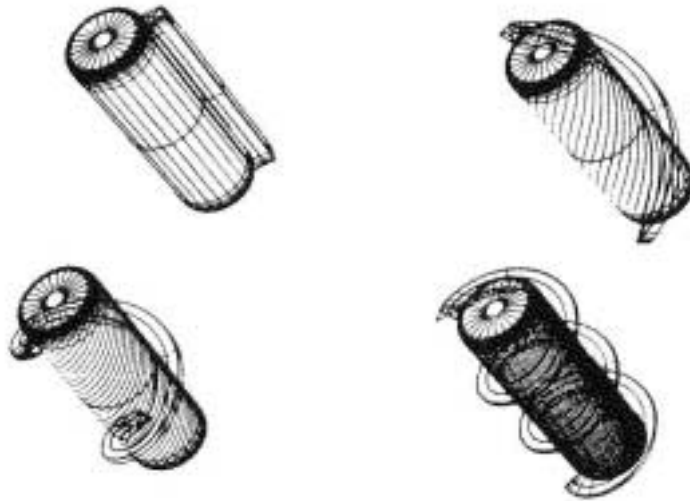


FIG. 2.1 – Un objet soumis à une torsion le long de son axe (extrait de [Bar84])

2.2.2 Modèles discrets

Ce sont des modèles qui ne s'appuient pas sur une structure continue mais qui sont basés sur un assemblage d'éléments discrets. Nous distinguerons ici les modèles à *particules* et les modèles dits *masses-ressorts*.

2.2.2.1 Les modèles à particules

Les modélisations à base de particules consistent à représenter mécaniquement un corps par un ensemble de masses ponctuelles. Les particules subissent leur propre poids, éventuellement des forces de frottement et des interactions entre elles (collisions, attraction ou répulsion entre les différentes particules). Les calculs consistent donc à faire le bilan de ces forces sur chaque particule, ce qui lui fournit une accélération. On intègre ensuite pour obtenir sa vitesse puis sa position.

Ce type de modélisation non structurée donne la liberté nécessaire à la représentation de fluides (liquides ou gaz), de solides pulvérisés (tas de sable) que l'on peut difficilement représenter avec d'autres modèles mécaniques.

L'affichage de systèmes particuliers peut se faire en affichant simplement un objet (généralement sphérique) au niveau de la particule ou alors en utilisant des surfaces implicites à squelette [DG95].

2.2.2.2 Les modèles masses-ressorts

Les réseaux de masses-ressorts sont un cas particulier des systèmes particuliers. Ces modèles mettent également en scène un ensemble de particules caractérisées par leur masse et leur position. Cependant des contraintes supplémentaires sont imposées aux particules : elles sont reliées entre elles par des ressorts.

Hormis les forces supplémentaires provoquées par les ressorts, les calculs à effectuer sont donc les mêmes que pour un modèle particulaire simple.

L'affichage de maillages masses-ressorts peut se faire de différentes manières selon le corps représenté. On pourra par exemple dessiner le graphe en représentant les particules et les ressorts qui les relient ou tracer des triangles dont les arêtes sont portées par les ressorts. Ou encore on pourra considérer les particules comme étant des points de contrôle de splines (des courbes ou des patches) voire les utiliser pour gouverner des FFD [SP86].

Citons le cas particulier de [LC86] où il est proposé d'ajouter des points dits géométriques, sans masse, dont la position se détermine par des contraintes géométriques (volume, surface, distance constante). Ces points géométriques ajoutent des nouveaux points de contrôle pour des

splines formant le contour des objets représentés (voir figure 2.2).

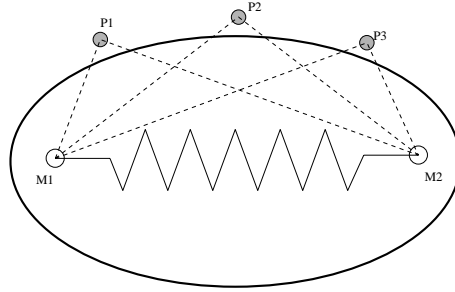


FIG. 2.2 – *Modélisation par points mécaniques et points géométriques*

2.2.3 L’habillage

2.2.3.1 Discussion

Les modèles de déformations présentés ci-dessus peuvent être visualisés de nombreuses manières. On pourra utiliser de simples maillages polygonaux, l’assemblage de surfaces paramétriques, des splines, des primitives géométriques à déformations globales, des FFD, des surfaces implicites (à squelette ou non), etc.

Tous ces outils de visualisation sont les briques fondamentales des modèles complexes utilisés en modélisation géométrique.

On voit donc que dans chacun des modèles présentés ci-dessus, on associe un modèle mécanique servant aux calculs des déformations et un modèle géométrique servant à la visualisation. La présentation que l’on en a fait tend volontairement à dissocier les deux modèles. Cependant, concevoir un modèle de déformations revient toujours à les associer.

Historiquement, on remarquera que dans les modèles les plus anciens, les modélisations partaient de l’utilisation d’outils de la mécanique et des mathématiques pour mettre en place un schéma de calculs du comportement. Les chercheurs en synthèse d’images se sont seulement ensuite posé le problème de leur visualisation.

Pourtant les modèles les plus récents tendent à faire l’inverse en allant piocher dans les outils géométriques de haut niveau (FFD, splines, surfaces implicites, etc) puis trouver le modèle calculatoire que l’on va lui appliquer pour l’animer. Cette tendance vient principalement du fait que l’on privilégie l’aspect visuel en cherchant à utiliser un modèle d’affichage détaillé et

complexe, puis à l'animer en mettant au point le modèle mécanique qui lui convient le mieux. P. Meseure a étudié les combinaisons entre les différents modèles géométriques et mécaniques [Mes97]. Les travaux du Leri sur les splines lagrangiennes (paragraphe 2.2.1.2) peuvent à présent compléter son inventaire.

2.2.3.2 Dualité mécanique/géométrie

Un modèle de corps déformables est donc obtenu par l'association de deux modèles (mécanique et géométrie).

Ces modèles, bien que liés, doivent satisfaire des contraintes antagonistes. D'une part, il est nécessaire de simplifier la modélisation mécanique afin non seulement de diminuer la quantité de calculs nécessaires aux animations mais également de fiabiliser ces calculs. En effet, plus on discrétise (au niveau mécanique) un objet, plus la masse par nœud est faible. Or la fréquence d'oscillation du système mécanique est en $\sqrt{k/m}$ où m est la masse du point mécanique et k caractérise les forces mises en jeu (la raideur d'un ressort par exemple). Démultiplier les points mécaniques réduit la masse moyenne e par nœud et augmente donc cette fréquence d'oscillation, ce qui peut causer des problèmes de convergence très gênants lors de l'animation.

D'autre part, nous voulons que la représentation graphique soit de très bonne qualité. Ceci implique que le modèle géométrique soit détaillé.

L'utilisation de modèles de déformations «intimement liés» (masse-ressorts, éléments finis, etc) où la géométrie du corps est directement obtenue par la position des points mécaniques ne permet pas de satisfaire ces contraintes. En effet, si l'on décide de favoriser la qualité des calculs de la mécanique (en utilisant un modèle mécanique simple), on hérite alors d'une représentation géométrique trop rudimentaire. Seuls les calculs sont satisfaisants. Par ailleurs, si l'on augmente suffisamment le nombre de points pour que la visualisation soit de qualité suffisante, on est alors contraint à travailler sur un modèle mécanique trop complexe pour du temps-réel et soumis à de nombreux problèmes d'instabilité numérique. La solution intermédiaire consistant à se donner un nombre «moyen» de points aboutit à un modèle mécanique moins coûteux mais dont les calculs restent souvent trop coûteux et imprécis. En plus de cela, le modèle géométrique reste trop simple pour fournir une visualisation de qualité acceptable.

L'utilisation de modèles très liés présente donc l'inconvénient de ne pas pouvoir satisfaire en même temps les besoins des deux modèles car il y a toujours la nécessité de favoriser l'un au détriment de l'autre.

L'idée qui consiste à fournir une «enveloppe» détaillée à un modèle mécanique simple permet de tirer profit de cette dissociation dont nous avons parlé précédemment.

2.2.3.3 Définition de l'habillage

Par le terme habillage on entend ici le fait de fournir un modèle géométrique à un modèle mécanique. Il s'agit d'établir un échange entre les deux couches du modèle afin que l'état du premier permette de construire le second (voir figure 2.3). Ce formalisme étudié dans [GVP91, Gas93, DDCB01] s'inscrit dans un cadre plus large : il s'agit de la problématique «forme/mouvement» étudiée à l'ACROE [Hab97].

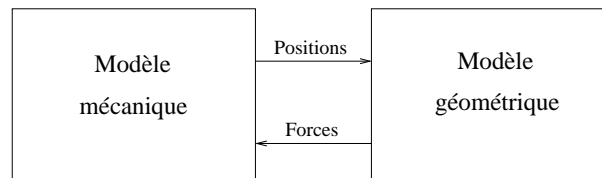


FIG. 2.3 – *Communication entre géométrie (forme) et mécanique (mouvement)*

Dans cette problématique on cherche à mettre en évidence les échanges entre la forme et le mouvement des objets. On s'intéresse d'une part à l'incidence des déplacements d'un objet sur sa forme en mettant en évidence la déformabilité des corps. D'autre part, la forme de l'objet va influencer sur son mouvement. En se déformant, il pourra par exemple entrer en contact avec d'autres corps, ou voir sa surface de contact avec un support diminuer ou augmenter et sa matrice d'inertie changer. La forme de l'objet affectera donc directement ses déplacements.

2.3 Notre plateforme : ALSPeMe

2.3.1 Présentation

Mon travail a pour contexte la plateforme *ALSPeMe* (pour Atelier Logiciel pour la Simulation Pédagogique Médicale). Cette plateforme est réalisée en collaboration avec la société SIMEDGE. Dans cette plateforme logicielle ont été formalisés les différents éléments conceptuels de la simulation. Elle vise à proposer un ensemble d'outils et de bibliothèques permettant de réaliser un simulateur pédagogique d'intervention chirurgicale.

2.3.1.1 Architecture

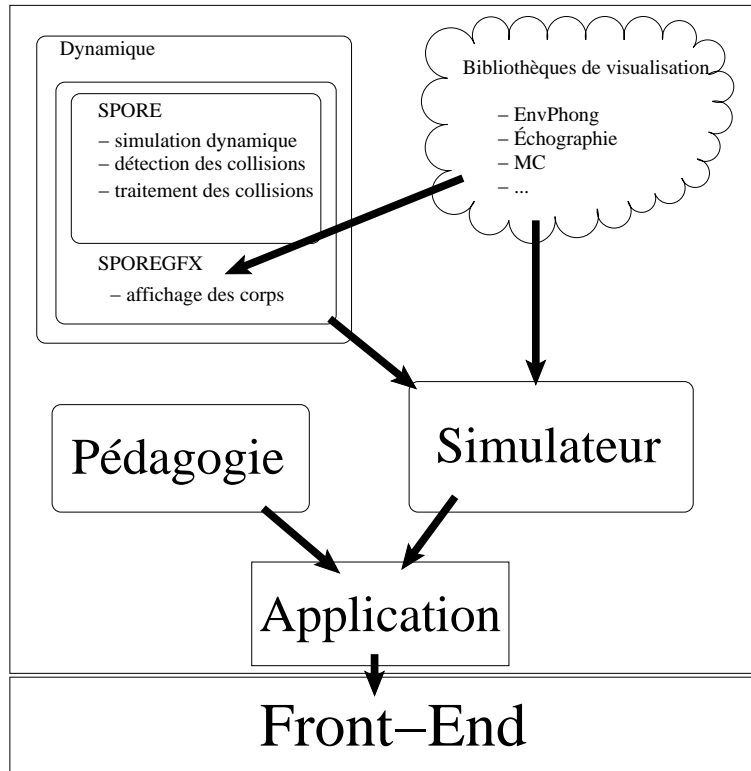


FIG. 2.4 – Architecture d'ALSPeMe

Nous trouvons dans cette architecture une *front-end* permettant de lancer une "Application" de simulation. Cette application dialogue avec deux parties du système: la "Pédagogie" et le "Simulateur". L'application dialogue avec ces deux sous-systèmes afin entre autres de déterminer si un exercice est effectué correctement.

Le noyau pédagogique contient un panel d'exercices (exercices de localisation, de manipulation d'outils comme présenté au paragraphe 2.1.4.1, etc) mais également des éléments d'aide comme des avertissements sonores, visuels ou tactiles. Ces routines sont génériques et indépendantes de l'opération simulée.

Les informations spécifiques à un simulateur donné (à savoir les descriptions d'organes, le mode opératoire, etc) sont stockées dans la partie "Simulateur" qui s'appuie sur des bibliothèques spécifiques d'affichage et éventuellement de calculs de la dynamique.

Parmi ces bibliothèques d'affichage, nous trouvons des outils permettant d'afficher des objets brillants telles les structures organiques, des images échographiques, et la librairie *MC* qui est le résultat synthétique des travaux de cette thèse et qui permet d'afficher des surfaces implicites à

squelette grâce à notre implantation de l'algorithme des Marching Cubes. D'autres bibliothèques de visualisations viendront sans doute se greffer à celles déjà existantes.

Concernant la dynamique, une bibliothèque appelée *SPORE* (Simulation Physique d'Objets virtuels avec Retour-d'Efforts) prend en charge effectue les calculs de la simulation, la détection et le traitement des collisions. Nous reviendrons sur cette bibliothèque au paragraphe 2.3.2. *SPORE* propose ses fonctions d'affichage au travers de *SPOREGFX*.

2.3.1.2 Un exemple d'utilisation d'ALSPeMe : SPIC

SPIC est un Simulateur Pédagogique d'Intervention Coelioscopique en gynécologie développé au LIFL depuis 1995. La coelioscopie est un mode opératoire utilisé en remplacement d'une ouverture de l'abdomen : on gonfle la cavité abdominale avec un gaz inerte afin de «détacher» les organes les uns des autres, on insère ensuite des outils par des orifices pratiqués sur les côtés de l'abdomen et une caméra par un troisième orifice se situant au niveau du nombril.

Le choc opératoire dû à ce mode d'intervention est nettement plus faible qu'une intervention classique car il présente l'avantage de ne pas nécessiter l'ouverture de l'abdomen. Les cicatrices sont quasiment inexistantes et les risques d'infection presque nuls.

Le problème de ce mode opératoire réside en deux points : la manipulation des outils et de la caméra d'une part, et celle des organes d'autre part.

Les outils et la caméra sont en effet manipulés depuis l'extérieur et tournent autour d'un point rotule par lequel ils ont été introduits. Ceci fait que le déplacement de la main qui manipule et de l'outil manipulé sont inversés. L'opération se fait en regardant sur un écran de contrôle ce que l'on filme au moyen de la caméra. Une difficulté est alors de reconstruire mentalement l'univers tridimensionnel dont on ne perçoit que lors d'une partie grâce à l'image bidimensionnelle de l'écran. De plus l'image que retransmet la caméra est dilatée. Enfin, on peut faire pivoter la caméra le long de son axe d'insertion ce qui peut vite faire perdre la notion de haut et de bas.

Il est parfois nécessaire de déplacer des organes encombrants (l'intestin par exemple). L'opération se réalisant *in vivo*, il n'est pas possible d'écarter les organes comme on peut le faire dans les modes opératoires invasifs. C'est au moyen des outils que l'on doit faire toutes ces manipulations qui peuvent être plus ou moins dangereuses selon la fragilité de l'organe et de son environnement.

2.3.1.3 La pédagogie

Les exercices proposés mettent en scène ou non des déformations et utilisent ou non le retour d'efforts. Selon l'exercice pris en compte, le simulateur pourra donc faire partie de l'une ou l'autre des catégories présentées au paragraphe 2.1.4.

Un premier exercice consiste à manipuler le dispositif de visualisation (une caméra, une sonde échographique, etc) en pointant différents objets ajoutés dans la cavité abdominale. Il s'agit d'en approcher la caméra sans toucher aux autres organes et en maintenant la caméra (ou tout autre dispositif de visualisation) orienté correctement. Des indicateurs sonores et visuels avertissent l'utilisateur de sa réussite et le préviennent en cas de mauvaise manipulation. Un autre type d'exercice consiste à faire suivre une trajectoire bien définie aux outils manipulés afin de bien appréhender les distances et les vitesses de déplacement pour mieux contrôler l'amplitude des mouvements. Pour ces raisons, on peut considérer que le simulateur impliqué ici est un simulateur de navigation car il n'y a aucune interaction «physique» entre les outils et l'environnement de simulation (paragraphe 2.1.4.1).

Des complications opératoires peuvent également être orchestrées, dans lesquelles des hémorragies peuvent survenir. Il est important de savoir les contrôler. L'exercice consiste alors à utiliser comme outil une sorte d'aspirateur pour vider le sang qui s'est écoulé dans la cavité abdominale. Cet exercice nécessite la prise en compte du mouvement de l'aspirateur, le sang est alors animé et déformé en fonction de ce que fait l'utilisateur. Cet exercice correspond donc à ce que nous avons présenté au paragraphe 2.1.4.2.

Certains exercices consistent à mesurer et évaluer le geste lors de la manipulation d'organes. Ceci a pour but de ressentir leur poids et en particulier l'effet de levier que l'on peut avoir en faisant pivoter un outil autour de son point d'insertion : le retour d'efforts est indispensable et les déformations des corps permettent d'en évaluer la déformabilité. On est ici dans le type de simulation présentée au paragraphe 2.1.4.3.

Le retour d'efforts peut également être utilisé dans la pédagogie afin d'augmenter le réalisme. Il peut en effet servir à remplacer la main du chirurgien expert qui guide habituellement celle de l'élève. Le retour d'efforts pourra ainsi être une aide à la manipulation lors d'exercices dits de navigation.

2.3.1.4 La bibliothèque dynamique : cahier des charges

La bibliothèque dynamique a pour rôle de fournir un environnement permettant la conception d'applications nécessitant la simulation d'objets dont le comportement doit être dicté par les lois de la mécanique. L'environnement simulé doit présenter les propriétés suivantes :

- Simulation de tout type de corps mécaniques, qui doivent interagir entre eux. Les forces calculées doivent permettre la commande de systèmes à retour d'efforts ;
- Simulation temps-réel, permettant de s'adapter aux fréquences nécessitées par le retour d'effort. Le *temps simulé* doit correspondre au temps réellement écoulé pour effectuer les calculs de la simulation ;
- Simulation interactive, afin de permettre à un utilisateur d'intervenir en temps-réel sur le comportement des objets.

Il n'existe pas à notre connaissance d'environnements capables de traiter ensemble plusieurs types de corps physiques différents en temps-réel.

La forme et le mouvement d'un objet sont deux caractéristiques très liées : la *forme* des objets permet de déterminer les collisions avec l'environnement et de fournir des forces pour créer du mouvement tandis que le *mouvement* calcule les déplacements et communique des positions et des déformations à la forme.

Les calculs de collisions se font en déterminant les interpénétrations éventuelles entre les corps. C'est la représentation géométrique des corps qui sert à effectuer ces calculs. On teste donc les éventuelles intersections entre les représentations géométriques des différents corps. Dans un souci de simplicité de traitement (et de développement) il est important d'*unifier* ces tests d'intersection. Pour cela on effectue les calculs sur les facettes qui composent la géométrie des corps.

Cependant, le modèle géométrique des corps doit être détaillé si l'on veut une représentation de qualité. De ce fait, les calculs de collisions ne peuvent généralement pas être effectués en temps-réel sur des objets un peu trop détaillés et/ou nombreux.

La partie qui suit présente les caractéristiques à la base de *SPORE* qui vont permettre de satisfaire le cahier des charges présenté ci-dessus.

2.3.2 Le moteur SPORE

Dans le moteur SPORE, on cherche à fournir un habillage à un modèle mécanique sans que cet habillage ne renvoie d'information vers la mécanique (figure 2.5). Il ne s'agit ici que d'une représentation graphique des objets.

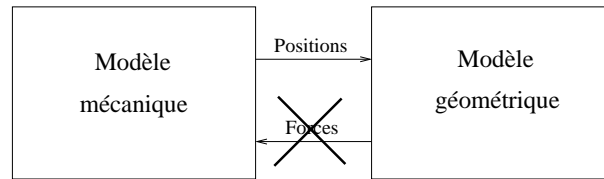


FIG. 2.5 – Communication entre géométrie (forme) et mécanique (mouvement) dans SPORE

2.3.2.1 Présentation

Comme nous l'avons dit précédemment, le moteur *SPORE* désigne une bibliothèque pour la Simulation basée sur la Physique d'Objets virtuels pour le Retour d'Efforts. Cette bibliothèque a été développée en C++ pour diverses plateformes (WIN32, Irix, Linux, etc). L'approche adoptée dans SPORE a été structurée autour des points suivants :

- ◇ Un noyau de simulation responsable :
 - de la détection unifiée des interactions entre corps ;
 - des calculs de la réponse à ces interactions ;
 - de la résolution des équations différentielles du mouvement.
- ◇ Une bibliothèque présentant un large éventail de corps simulables en temps-réel, afin de couvrir un maximum de comportements mécaniques ;
- ◇ De méthodes permettant de transmettre les résultats de la simulation à un système à retour d'efforts quelconque.

En outre, *SPORE* est complétée par une seconde librairie, appelée *SPOREGFX*, et capable de fournir un affichage à tous les corps traités par *SPORE*.

Remarque : *Sous certains aspects, SPOREGFX peut être considérée comme une bibliothèque d'effets spéciaux. Nous rangeons dans cette catégorie tout un panel d'effets visuels reposant normalement sur des concepts de la mécanique mais dont la représentation peut être*

grandement simplifiée en s'accordant quelques simplifications. La représentation de la fumée en est un exemple : simuler une cautérisation réaliste nécessite de représenter la fumée qui émane de la brûlure. Or, les calculs basés sur un modèle particulière de la fumée sont complexes et coûteux à utiliser tandis que l'animation de quelques textures correctement dessinées peuvent suffire à «bluffer» l'utilisateur. Dans le même ordre d'idée un fluide modélisé par un ensemble de particules pourra être avantageusement remplacé par un simple polygone sur lequel on appliquera une texture animée et du Bump-mapping pour simplifier les traitements spécifiques à une plaque.

Le but de ces effets spéciaux n'est pas de remplacer la mécanique par des «bidouilles» visuelles mais au contraire de fournir certaines alternatives simples à des problèmes que la mécanique pourrait avoir des difficultés à résoudre et qui, pour autant, ne sont pas nécessaires à l'entraînement du chirurgien. Nous utilisons par exemple au chapitre 4 une méthode permettant de propager des déformations sur les surfaces qui entrent en contact. Ces déformations sont purement géométriques alors que leur prise en compte au niveau mécanique complexifierait grandement la modélisation mécanique.

Les différents types de modèles à simuler doivent être suffisamment simples, afin d'être utilisables en temps-réel. Cette simplicité concerne à la fois la résolution des équations et de leur comportement (évoqués au paragraphe 2.2.3.2) et la facilité de détection de leurs interactions avec l'environnement. Cette simplicité ne doit cependant pas se manifester lors de la représentation graphique du corps.

Par ailleurs, afin de simplifier la gestion des collisions entre corps (détection et calcul de la réponse), il a été choisi de la centraliser au niveau du noyau : tous les corps doivent donc être traités de façon identique et pour cela, être représentés par une structure de données unique. Le modèle de collisions unifié choisi est l'approximation du volume des objets par un ensemble de sphères. Ces sphères sont les nœuds de communication entre l'objet et l'extérieur. Toutes les interactions mécaniques passent par ces nœuds. Les sphères ont été retenues car elles présentent les qualités suivantes : elles reposent sur le concept de nœud, commun à tous les modèles, elles sont invariantes par rotation (moins de calculs de positionnement) et surtout, il est très facile de déterminer si deux sphères s'intersectent. Le calcul de collision entre deux objets revient à vérifier s'il existe des sphères approximant leur volume qui s'interpénètrent.

Pour concilier les problèmes de représentation graphique de qualité et de gestion unifiée des collisions par représentation en sphères, tous les corps simulés dans *SPORE* se présentent en

trois couches superposées (figure 2.6) :

- Un modèle mécanique propre au corps et responsable du calcul de son évolution ;
- Un modèle géométrique de qualité, fabriqué à partir du modèle mécanique ;
- Un modèle de collisions constitué de sphères pour la gestion des interactions.

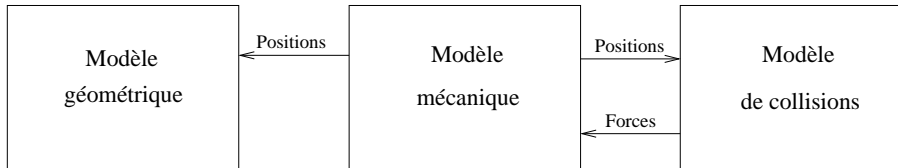


FIG. 2.6 – *Communication entre modèles*

Nous allons illustrer ce principe de superposition des modèles sur deux exemples : la figure 2.7 présente les trois couches d'un corps rigide (un cube) et la figure 2.8 montre le cas d'un fluide.

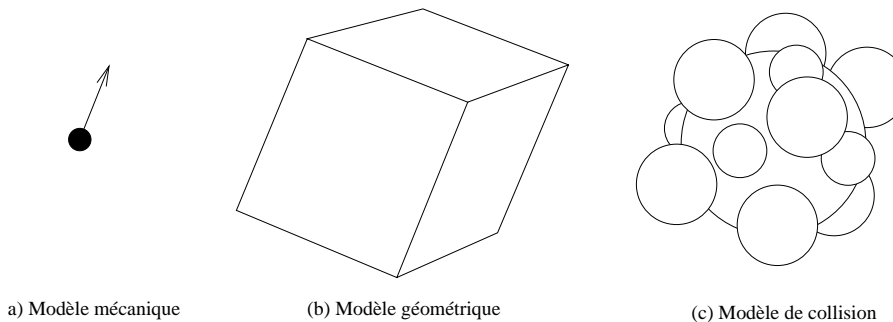


FIG. 2.7 – *Exemple de décomposition en couches d'un corps rigide*

Le modèle mécanique d'un cube rigide est composé d'un point, la position du centre de masse, et d'une orientation (une matrice de rotation, un quaternion ou des angles d'Euler). Le modèle géométrique est composé d'un ensemble de primitives géométriques, comme des facettes par exemple. Le modèle de collisions approxime le cube par des sphères. Elles peuvent s'interpénétrer ou rester tangentes, être de tailles différentes, et même surdimensionner le volume.

Cette modélisation en trois couches s'adapte également très bien à des corps non structurés comme les fluides (figure 2.8).

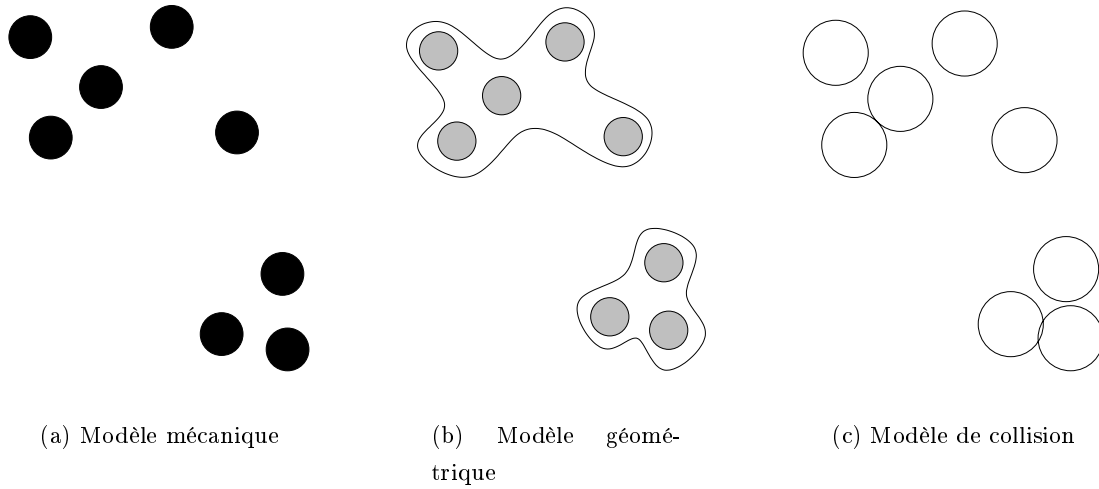


FIG. 2.8 – Exemple de décomposition en couches d'un fluide

Mon travail se situe à la jonction entre la géométrie et la mécanique : mon but est de construire le modèle d'affichage d'un fluide à partir des informations de positions fournies par le modèle mécanique.

2.3.2.2 Choix de l'habillage

L'idée directrice est d'habiller le modèle mécanique grâce à un modèle géométrique sophistiqué. Différentes solutions s'offrent à nous pour faire cet habillage. Il existe différents types de primitives géométriques dont l'utilisation est suffisamment simple pour permettre bâtir les surfaces que l'on désire.

L'utilisation de *patches* comme les surfaces de Bézier ou les NURBS est suffisamment simple pour se faire en temps réel : de simples fonctions d'interpolation fournissent directement la position de tous les points de la surface en fonction des points de contrôle. Cependant, bien que cette utilisation soit très simple (les accélérateurs graphiques actuels prennent même en charge le tracer de ces surfaces), il est difficile de les contrôler «efficacement». En effet, pour obtenir des formes complexes on a généralement besoin d'un nombre important de points de contrôle décrivant la surface. De même, certains problèmes se posent aux raccords entre les différentes surfaces : obtenir une continuité d'ordre un ou deux entre deux patches est parfois une tâche ardue et toujours coûteuse.

Les surfaces implicites à squelette forment une alternative très intéressante aux patches. Effectivement, elles présentent l'avantage considérable de fournir un contrôle précis de la forme et de la topologie des surfaces tout en ne requérant qu'un nombre raisonnable de points «de contrôle»¹¹. Cet avantage est d'autant plus flagrant lorsque l'on se place dans un contexte d'animation où les formes et surtout la topologie des objets évoluent avec le temps. Une modification de topologie entre des patches (leur fusion ou leur séparation) est extrêmement difficile à réaliser. Enfin, on obtient facilement des formes «organiques», c'est-à-dire des surfaces lisses qui ne présentent pas de contours anguleux.

Les deux images qui suivent (figure 2.9) présentent deux captures d'écrans du simulateur que nous développons actuellement avec *ALSPeMe*. La première image présente un carré de tissu modélisé au niveau de la mécanique par un réseau de masses-ressorts (ressorts linéaires et angulaires). On peut manipuler en temps-réel cet objet au moyen de la pince. La représentation géométrique est assurée par un pavage de patches de Catmull-Rom. La seconde image présente un écoulement sanguin modélisé par un système particulaire animé par des forces de Lennard-Jones et représenté par des surfaces implicites à squelette.

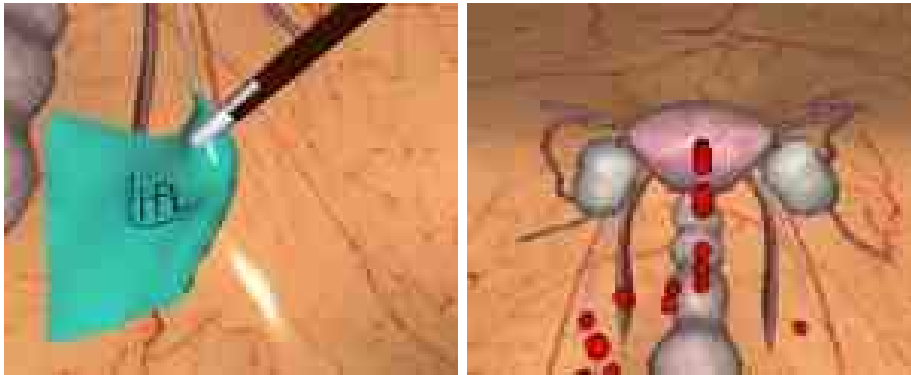


FIG. 2.9 – Captures d'écrans

Conclusion

Les différents simulateurs chirurgicaux présentés ici marquent la nécessité de prendre en compte le comportement des corps entrant en jeu dans la simulation. Pour que les propriétés mécaniques puissent être prises en considération, il est nécessaire de les représenter au sein du

11. par la suite, nous ne parlerons plus de points de contrôle comme pour les patches, mais plutôt de primitives, accordons-nous ici cet abus de langage

simulateur. Dans la plateforme *ALSPeMe* définie pour automatiser la conception des simulateurs, une bibliothèque, nommée *SPORE*, a pour rôle la simulation mécanique des corps.

D'après les expérimentations que nous avons pu faire, nous avons vérifié que les principes du moteur dynamique *SPORE* sont viables et que la distinction de trois couches (collisions, mécanique et géométrique) est pertinente.

Pour toutes les raisons évoquées ci-dessus, nous avons trouvé intéressant de pouvoir d'habiller certains modèles mécaniques au moyen de surfaces implicites à squelette. Ces surfaces vont venir compléter d'autres modèles géométriques dédiés à d'autres modèles mécaniques (par ex. les maillages de triangles pour les solides indéformables). Comme vu précédemment, l'affichage de surfaces implicites à squelette pose certains problèmes que nous allons étudier au chapitre suivant.

Nous tenons le pari que l'algorithme des Marching Cubes, réputé pour être très lent, peut être amélioré à tel point que l'on pourra l'utiliser dans un contexte temps-réel. Ces améliorations sont présentées au chapitre suivant.

Chapitre 3

L’Affichage temps réel de surfaces implicites à squelette

Sommaire

3.1	Quelques faiblesses des techniques actuelles	66
3.2	Les choix concernant le potentiel	67
3.2.1	La fonction de potentiel	67
3.2.2	Accélération de la sommation	69
3.3	Les améliorations des Marching Cubes	71
3.3.1	Le suivi de surface	71
3.3.2	La table de facettisation	73
3.3.3	La détection des calculs redondants	75
3.4	Résolution des cas de facettisation incorrecte	76
3.4.1	Rappels	76
3.4.2	Une première solution	77
3.4.3	Notre solution	78
3.5	Bilan	81
3.5.1	Besoins en mémoire	81
3.5.1.1	Les tableaux dynamiques	81
3.5.1.2	Les tables de hashage	83
3.5.1.3	Comment économiser de la mémoire	84
3.5.2	Utilisation de la cohérence temporelle	85
3.5.3	Les Marching Cubes adaptatifs	86

Introduction

Comme convenu à la fin du premier chapitre, nous nous intéressons à présent à l’algorithme des Marching Cubes. Bien qu’il ait une mauvaise réputation en termes de rapidité d’exécution, nous allons voir que de nombreuses améliorations peuvent lui être appliquées.

Convention

Dans la suite du document, lorsqu’un cube sera représenté, la couleur de ses sommets aura toujours la même signification : un sommet plein (noir) sera considéré à l’intérieur de la surface alors qu’un sommet vide (blanc) sera à l’extérieur.

3.1 Quelques faiblesses des techniques actuelles

L’algorithme présenté au paragraphe 1.3.3.1 peut s’exprimer sous la forme suivante :

```
POUR chaque cube de la grille FAIRE
  construire l’octet INDEX:
  POUR chacun des 8 sommets du cube FAIRE
    évaluer  $F$  pour le  $i^{\text{e}}$  sommet
    SI  $F(i^{\text{e}} \text{sommet}) < \text{SEUIL}$  ALORS
      mettre à 1 le  $i^{\text{e}}$  bit de INDEX
    SINON
      mettre ce bit à 0
  FINSI
FINPOUR
SI INDEX  $\neq 0$  ET INDEX  $\neq 255$  ALORS
  calculer les coordonnées des points d’intersection
  entre la surface et les arêtes du cube,
  générer l’ensemble des facettes correspondant à la
  portion de surface présente dans le cube
FINSI
FINPOUR
```

En s’intéressant de près à cet algorithme on se rend compte qu’en plusieurs points il souffre de certaines faiblesses. En particulier, on peut exhiber deux classes d’améliorations possibles. D’une part, nous allons proposer certains choix de simplifications concernant la fonction de potentiel

permettant d'accélérer les calculs de potentiels et de gradients. D'autre part nous allons proposer des améliorations visant à réduire le nombre de cubes inspectés par les traitements. Tout ceci va être mis en œuvre avec des structures de données optimisées permettant un codage automatique des actions à effectuer et des détections de calculs redondants.

Ces améliorations ont été présentées dans [TMC01]. Nous détaillons tous ces points dans les paragraphes qui suivent.

En plus des choix et des optimisations proposées, nous donnons une solution au problème bien connu de facettisation ambiguë ou incorrecte qui mène en général à l'apparition de trous dans la surface facettisée.

3.2 Les choix concernant le potentiel

Reprenons l'expression du potentiel pour le cas particulier des surfaces implicites à squelette (voir le chapitre 1) :

$$F(x,y,z) = \sum_1^N f_i(x,y,z) \quad (3.1)$$

où N est le nombre de primitives traitées, et f_i est la fonction de potentiel associée à la $i^{\text{ème}}$ primitive et qui varie selon la distance entre le point considéré (x,y,z) et la primitive. L'évaluation de F pourra donc être améliorée en simplifiant l'expression des f_i et en tentant de réduire la somme.

3.2.1 La fonction de potentiel

Nous avons choisi d'utiliser une fonction de potentiel à rayon d'action fini. D'une part parce que la modélisation avec des primitives à rayon d'action fini est parfois plus intuitive dans la mesure où le déplacement, l'apparition ou la suppression d'une primitive n'affecte que localement les surfaces. D'autre part, l'utilisation de primitives à rayon d'action fini va nous permettre l'optimisation présentée ci-après (au paragraphe 3.2.2).

De nombreuses fonctions ont été proposées dans la littérature [Bli82, NHK⁺85, WMW86b, MI87, Gas93, BS95]. Le tableau qui suit présente certaines de ces fonctions ainsi que les temps de calculs nécessaires à 10^7 évaluations sur un Athlon 800MHz. Un paramètre p intervient dans l'expression de certaines fonctions. Sa valeur caractérise la pente de la fonction et donc les formes

obtenues par blending (voir les illustrations du tableau 1.2 en page 12). Toutes les fonctions n’utilisent pas ce paramétrage, c’est pourquoi pour nos mesures de temps nous avons fixé arbitrairement p à 1 (qui donne des formes de blending esthétiques). Les expressions données ici ne sont donc pas exactement celles de leurs auteurs.

Auteur	Expression	Temps en secondes
[Bli82]	$f(d) = e^{-Ad^2}$ $d \in \mathbb{R}$	4.06
[NHK ⁺ 85]	$f(d) = 4/3 - 4d^2$ $f(d) = 2(1 - d)^2$ $0 \leq d \leq 1/3$ $1/3 \leq d \leq 1$	1.33
[WMW86b]	$f(d) = 1 - 22/9d^2 + 17/9d^4 - 4/9d^6$ $d < 1$	0.63
[MI87]	$f(d) = (1 - d^2)^2$ $0 \leq d \leq 1$	0.58
[Gas93]	$f(d) = 1/4(3 - 2d)$ $f(d) = (-1 + 8d - 2d)(1 - d)^2$ $d < 1/2$ $1/2 \leq d < 1$	1.13
[BS95]	$f(d) = 1 - \frac{(3d^2)^2}{1+0.5d^2}$ $f(d) = \frac{(1-d^2)^2}{-0.25+4.5d^2}$ $d^2 < 1/4$ $d^2 < 1$	0.87

Nous avons choisi d’utiliser la fonction proposée par Murakami [MI87], car cette fonction est la plus rapide à évaluer. C’est un simple polynôme bi-carré de degré 4 :

$$F_i(p) = \begin{cases} d_i^2(x,y,z) = (x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 \\ A_i * \left[1 - \left(\frac{d_i(x,y,z)}{R_i} \right)^2 \right]^2 & \text{si } \|d_i\| < R_i \\ 0 & \text{sinon} \end{cases} \quad (3.2)$$

où d_i est la distance entre le point de coordonnées (x,y,z) et la i^e primitive, R_i est le rayon d’influence de cette primitive et A_i un facteur de mise à l’échelle. Ce polynôme par morceau est identiquement nul au-delà de R_i .

Comme c’est un polynôme de degré pair sur sa variable (la distance), il n’est pas nécessaire d’extraire de racine carrée. Ceci n’est pas le cas pour toutes les fonctions proposées dans les articles cités ci-dessus

Lors de l’affichage, on a besoin de connaître des vecteurs normaux à la surface afin de lui fournir un éclairage. Ces vecteurs sont obtenus en calculant le gradient de la fonction de potentiel :

$$\overrightarrow{\text{grad}} F_i(x,y,z) = \begin{cases} \frac{4 * A_i}{R_i} * \left[1 - \left(\frac{d_i(x,y,z)}{R_i} \right)^2 \right] * \overrightarrow{d}_i(x,y,z) & \text{si } \|d_i(x,y,z)\| < R_i \\ \overrightarrow{0} & \text{sinon} \end{cases} \quad (3.3)$$

La fonction de potentiel de Murakami a un coût très faible, elle n'utilise que des opérations simples (additions et multiplications), en particulier elle permet d'éviter les exponentielles et les racines carrées. De plus elle approxime relativement bien la gaussienne et les formes de blending obtenues sont visuellement aussi satisfaisantes que celles obtenues grâce aux autres fonctions plus coûteuses (voir le tableau 1.2 en page 12 qui présente différentes fonctions de potentiels et les blendings obtenus).

3.2.2 Accélération de la sommation

Pour obtenir le potentiel d'un point donné de l'espace, on effectue la somme des potentiels apportés par chacune des primitives du squelette. Pour citer les auteurs de [WMW86b] : « *C'est la manière la plus simple pour combiner les effets de plusieurs points et il semble que cela fonctionne bien* ».

Cependant comme nous venons de le voir, nous avons choisi des primitives à rayon d'action fini, qui par définition n'influent pas sur le potentiel des points situés au-delà de leur rayon d'influence. Pour une partie des primitives l'évaluation du potentiel va donc se résumer à calculer la distance¹² entre le point et la primitive puis conclure qu'étant donnée cette distance le potentiel est nul. Il n'est donc pas nécessaire de sommer les potentiels de toutes les primitives du squelette et l'on peut se restreindre aux primitives suffisamment proches.

Pour cela nous avons mis en place une grille de *voxels* (figure 3.1). Chacun de ces voxels contient la liste des primitives influentes. Les listes sont remplies au début de chaque appel de facettisation. L'évaluation du potentiel dans un voxel ne requiert que les contributions des primitives de ce voxel. Ainsi cette structure permet de ne considérer que les primitives « locales » lors des évaluations de potentiels mais aussi lors de la construction des vecteurs normaux aux surfaces.

12. c'est en fait une comparaison entre le carré de la distance et le carré du rayon d'influence, ce qui évite le calcul d'une racine

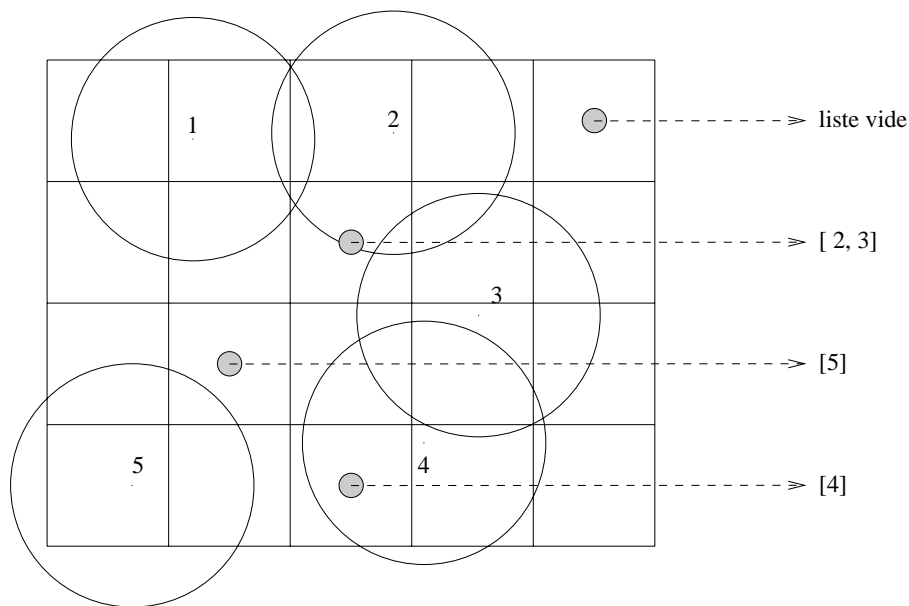


FIG. 3.1 – Grille de voxels : les primitives numérotées de 1 à 5 et leur sphères d’influence interviennent dans la composition des différentes listes, il s’agit des sphères d’influence et non pas des surfaces engendrées par ces primitives

Il est important de bien différencier la grille de « cubes » et celle de « voxels ». L’algorithme des Marching Cubes est fondé sur la première : on génère des triangles au sein de ces cubes et ce sont donc eux qui permettent de discrétiser la surface. Par ailleurs, notre implantation a recours à une toute autre grille de résolution et de position éventuellement différentes de la grille de cubes. Il s’agit de notre grille de voxels, qui correspond également à une discrétisation particulière de l’espace, et qui contient des listes de primitives. On voit par exemple sur la figure 3.1 certains voxels ne subissent l’influence que d’une ou deux primitives, tandis que l’on aurait normalement dû sommer les potentiels de chaque primitive du squelette.

La taille des voxels influe bien évidemment sur les performances de l’ensemble. En effet, si les voxels sont trop grands, on en perd tout l’intérêt puisqu’ils vont contenir beaucoup de primitives, éventuellement non-influents, et en revanche des voxels trop petits vont demander une initialisation très coûteuse.

Le choix de la taille de ces voxels est laissée à l’utilisateur : il est difficile (pour ne pas dire impossible) de déterminer de manière automatique quelle doit être la taille des voxels pour obtenir les meilleures performances. Il doit s’agir d’un ratio entre la taille des voxels, la taille des cubes et le rayon d’influence des primitives (qui peuvent être différents pour chacune d’elles !). Pour un squelette donné, des heuristiques peuvent certainement déterminer une grille de voxels permettant d’obtenir de « bonnes » performances. Une étude approfondie de ce problème me

semble intéressante. Néanmoins, il est important de souligner que cette structure de donnée est actuellement fixée une fois pour toutes. Lorsque la taille des primitives augmente ou diminue la grille choisie n'est plus nécessairement «bonne», il est alors nécessaire d'en déterminer une nouvelle.

3.3 Les améliorations des Marching Cubes

3.3.1 Le suivi de surface

L'algorithme initial des Marching Cubes [LC87] est utilisé pour reconstruire des surfaces à partir de données tridimensionnelles. La surface occupe alors la totalité de la grille et il n'est donc pas aberrant de traiter tous les cubes un par un.

Notre contexte est très différent. Notre but est d'afficher des surfaces implicites animées par des lois de la mécanique. Ces surfaces vont pouvoir évoluer dans un espace donné (se déplacer, se déformer, etc). Les surfaces vont donc rarement être présentes dans l'ensemble de la grille des Marching Cubes. Certains tests nous ont en effet montré que lors de nos animations, il est fréquent que les surfaces occupent moins de 25% des cubes de la grille.

C'est pourquoi notre implantation ne va s'occuper que des cubes coupés par la surface (appelons-les «cubes utiles»). Cette idée, proposée par J. Bloomenthal [Blo94], consiste à traiter un premier cube puis à étendre le traitement uniquement à ses voisins utiles. Ceci est fait récursivement et s'arrête lorsque toute la surface a été traitée.

Notre implantation prend en charge la récursivité en proposant sa propre pile (comme le font [Blo94, ZJ91]) : un cube à traiter sera empilé et lorsqu'il aura été facettisé, on empilera les voisins utiles. Le traitement principal consiste à recommencer ce processus en prenant un autre cube sur la pile. Le principe peut être exprimé en pseudo-code comme suit :

```
initialiser la pile de cubes à traiter
TANT QUE la pile n'est pas vide FAIRE
    depiler un cube
    facettiser ce cube
    POUR chacun des voisins utiles FAIRE
        empiler ce cube voisin
    FINPOUR
FINTANQUE
```

Plusieurs problèmes se posent alors :

- comment ne pas traiter plusieurs fois un même cube ?
- comment trouver le cube de départ ?
- comment être sûr de traiter toute la surface, sachant qu’elle peut être composée de nombreuses parties disjointes ?

Une fois le cube empilé on le «marque» pour qu’il ne puisse plus être empilé dans le reste de la facettisation. C’est le mécanisme de *date* présenté au paragraphe 3.3.3 qui va nous permettre de «marquer» les cubes empilés.

Les deux problèmes suivants sont liés : le traitement s’effectuant d’un cube à son voisin, on ne pourra traiter deux surfaces disjointes que si l’on a au moins un cube de départ sur chaque composante. Ainsi on pourrait reformuler les deux problèmes comme suit : comment trouver au moins un cube sur chaque composante connexe ?

Notre méthode s’appuie sur le fait que toute primitive peut générer une surface disjointe de toutes les autres. Pour chacune des primitives du squelette nous recherchons donc un cube sur la (les) surface(s) engendrée(s). Cette recherche s’effectue depuis l’intérieur de la surface (le centre de la primitive concernée) et dans une direction arbitraire (selon les abscisses croissantes) comme indiqué sur la figure 3.2. Elle va parer au pire des cas, c’est-à-dire celui où il faut avoir un cube de départ par primitive.

[Blo94, ZJ91] font également du suivi de surface. Cependant leurs méthodes ne prennent pas en compte les surfaces disjointes : il est nécessaire d’effectuer autant de facettisations distinctes qu’il y a de surfaces disjointes. Nous aurions pu nous contenter d’un seul cube de départ par composante connexe, cependant nous avons préféré notre méthode exhaustive car elle permet d’éviter une recherche de connexité entre les surfaces.

Même si cette optimisation est vraiment très efficace dans le contexte de l’animation (où la grille n’est pas intégralement remplie de primitives), elle reste intéressante même dans d’autres domaines comme la reconstruction (voir le paragraphe 5.6) où l’on doit traiter presque la totalité de la grille. Le surcoût occasionné par les *push/pop* reste négligeable devant le reste des opérations à effectuer.

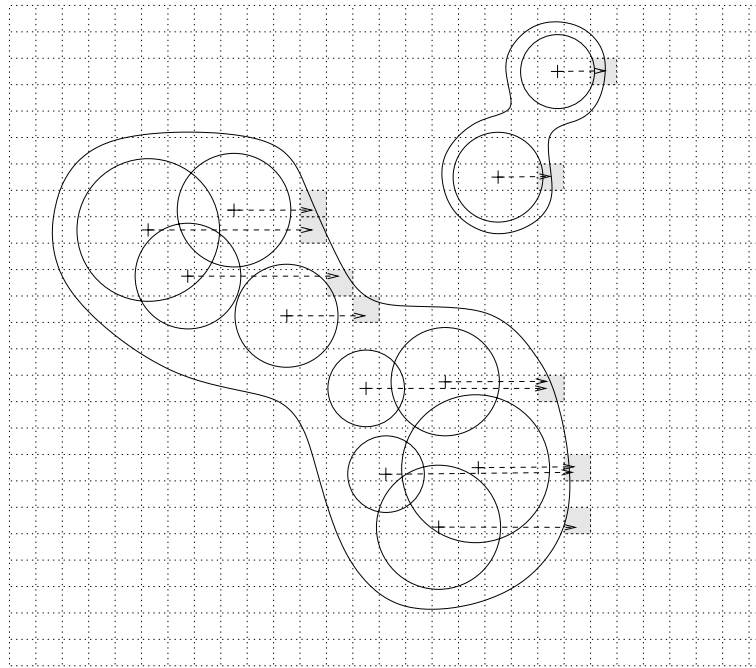


FIG. 3.2 – Recherche de cubes de départ pour le suivi de surfaces, les cubes choisis sont grisés.

3.3.2 La table de facettisation

La position des points d'intersection entre la surface et les arêtes des cubes dépend de la configuration du cube traité. Dans [Blo94], J. Bloomenthal effectue une recherche ordonnée au sein de chaque face du cube et qui suit la surface (voir figure 3.3). L'ordre d'énumération des arêtes est choisi pour que la surface soit décrite dans le sens horaire.

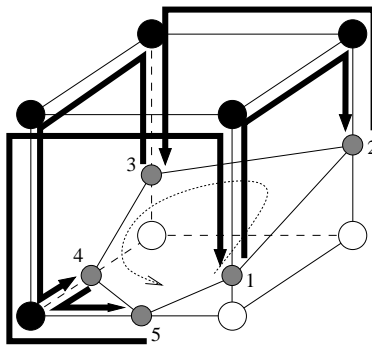


FIG. 3.3 – Facettisation dans un cube par Bloomenthal. En gras, l'ordre de parcours des sommets, en gris, les points d'intersection avec la surface et en pointillés l'ordre des sommets

Les auteurs de [LC87] proposaient une méthode moins procédurale en utilisant une table contenant pour chaque configuration possible la description des triangles à construire.

Nous avons repris cette idée classique qui consiste à construire un octet dont chacun des 8 bits indique la position du sommet correspondant par rapport à la surface : 0 pour un sommet à l’extérieur de la surface et 1 à l’intérieur. La numérotation est présentée en figure 3.4 et donne pour le cube de la figure 3.3 l’octet 11110001 (= 241).

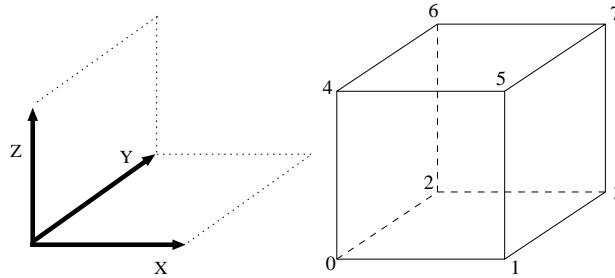


FIG. 3.4 – Numérotation des sommets d’un cube

À chaque entrée, notre table contient la liste des triangles décrits par les numéros des arêtes (du cube) qui portent les sommets (voir figure 3.5). Pour le même exemple que précédemment, nous avons la description suivante¹³ : { (0, 6, 5) (0, 3, 6) (3, 7, 6) } .

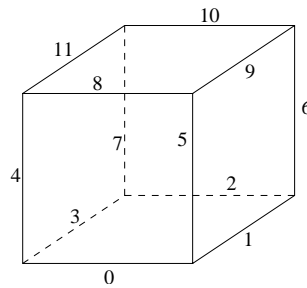


FIG. 3.5 – Numérotation des arêtes d’un cube

À cette description nous ajoutons la liste des cubes voisins qu’il est nécessaire de visiter. Cette liste nous permet de traiter de manière systématique le suivi de surface par voisinage présenté au paragraphe 3.3.1. Les cubes voisins sont identifiés eux aussi par un numéro (voir figure 3.6). Dans l’exemple, nous avons comme voisins utiles : { 1, 2, 3, 4, 5 }.

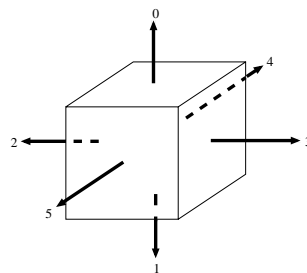


FIG. 3.6 – Numérotation des cubes voisins

13. nos triangles sont décrits dans le sens direct (anti-horaire)

3.3.3 La détection des calculs redondants

Notre but est d'éviter de calculer plusieurs fois une même valeur durant une boucle principale de l'algorithme. La solution apportée sera appliquée par exemple au calcul du potentiel d'un point de la grille commun à plusieurs cubes. Pour cela, nous stockons le résultat dans un tableau, à chaque valeur de ce tableau correspond une marque indiquant si la donnée est valide ou si elle doit être calculée.

Une méthode classique consiste à attribuer un booléen à chaque valeur. Ce booléen indique si la valeur est correcte (**VRAI**) ou non (**FAUX**). Malheureusement une telle méthode requiert d'effacer (en les mettant à **FAUX**) tous les booléens à chaque facettisation.

C'est pourquoi nous avons préféré remplacer le booléen par un nombre que l'on compare au numéro de la facettisation. Appelons dès maintenant ce compteur une *date* (*timestamp* en anglais).

À chaque facettisation, cette date est donc incrémentée et lorsque l'on doit évaluer un potentiel on compare la date principale à la date du potentiel. Lorsque les deux dates coïncident il suffit de lire la valeur dans le tableau et dans le cas contraire il faut la calculer, la mémoriser et la marquer de la date courante.

Cette méthode permet de ne pas avoir à effacer l'intégralité des dates à chaque boucle du programme puisqu'à la boucle suivante la date globale sera incrémentée et toutes les dates du tableau seront considérées comme invalides car différentes de la date globale. Bien entendu il est nécessaire de les effacer au moment où la date principale arrive à la limite de capacité. À ce moment, on pourrait «confondre» des valeurs actuelles avec d'autres ayant été calculées avant le bouclage (et n'ayant pas été modifiées depuis). Dans [TMC99, TMC01] nous utilisons un entier codé sur 32 bits. Cette date pouvait donc prendre plus de 4 milliards de valeurs différentes. Pour une animation à 70 facettisations par secondes, les dates n'atteignaient leur valeur limite qu'au bout de 23 mois sans interruption¹⁴...

Après avoir pris en compte certains aspects liés à la quantité de mémoire utilisée, nous avons jugé préférable de manipuler des dates codées simplement sur 16 bits. Les gains en mémoire sont loin d'être négligeables pour des grilles de taille importante et la durée d'une animation sans effacement complet de toutes les dates du système reste tout à fait acceptable. En effet il faut 21 minutes pour que la date principale atteigne sa limite de capacité lors d'une animation à 50

14. $\frac{2^{32}}{70 \cdot 3600 \cdot 24 \cdot 30} \approx 23$ ou $2^{32} * (1/70 \text{ }^e\text{me de seconde}) \approx 23 \text{ mois}$

images par seconde.

Ce mécanisme d’«horodatage» est utilisé à plusieurs niveau de notre implantation :

- le potentiel des nœuds de la grille ;
- la position des points du maillage de triangles (qui sont les points d’intersection entre les arêtes de la grille et la surface) ;
- le gradient du potentiel au niveau de ces points ;
- les listes chaînées contenues dans les *voxels* ;
- les cubes empilés ;
- les cubes traités (facettisés) ;
- ...

3.4 Résolution des cas de facettisation incorrecte

3.4.1 Rappels

Comme nous l’avons vu au premier chapitre (page 23 et suivantes), la position des huit sommets d’un cube par rapport à la surface ne suffit pas toujours pour déterminer la topologie de la surface dans le cube.

Certaines configurations peuvent alors être facettisées de manières différentes, ce qui peut causer des discontinuités dans les maillages construits.

Certaines méthodes tendent à fournir des maillages dont la topologie est en accord avec la topologie de la surface théorique alors que d’autres se contentent de bâtir des maillages continus ne présentant pas de trou.

Comme nous avons pu le constater aucune méthode n’est totalement fiable ni pleinement satisfaisante. En plus de cela elles nécessitent toutes des traitements supplémentaires comme l’évaluation du potentiel d’autres points ou bien multiplient grandement le nombre de triangles générés.

3.4.2 Une première solution

Nous avons essayé une première méthode consistant à insérer dans le maillage des facettes au niveau des faces pouvant poser problème (voir figure 3.7).

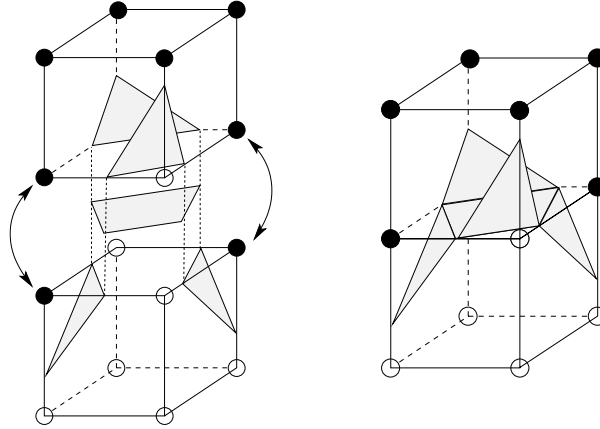


FIG. 3.7 – Ajout de facettes

Lorsque l'on traite un cube qui présente une face ambiguë, il n'est pas toujours nécessaire d'ajouter de facette. Il faut connaître la configuration de l'autre cube avec lequel on partage cette face pour pouvoir faire cette décision. La figure 3.8 illustre une situation dans laquelle il ne faut pas ajouter de facette.

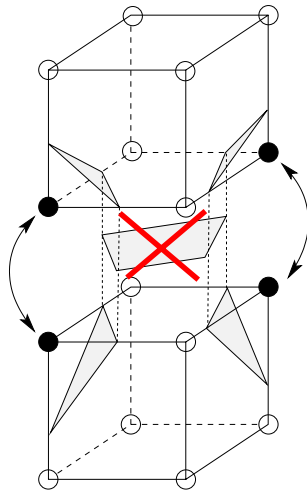


FIG. 3.8 – Ajout de facette erroné

Le traitement nécessiterait la mise en attente du cube qui pose problème jusqu'à ce que le cube voisin soit traité. On pourrait ainsi ensuite décider d'ajouter ou non la face nécessaire. Cependant, ce second cube dont on doit attendre le traitement peut également avoir besoin d'un troisième cube pour être construit (lorsqu'une autre face pose problème). Cette mise en attente

risque dans certaines conditions de conduire à des files d’attente très longues et éventuellement à des inter-blocages.

Cette méthode peut donner des résultats visuels satisfaisants cependant elle n’a pas été retenue car sa mise en œuvre est délicate.

La solution que nous allons proposer reste une méthode basée sur une décomposition de l’espace en cubes. Elle présente l’avantage de ne nécessiter aucun traitement supplémentaire.

3.4.3 Notre solution

Afin de bien cerner les raisons pour lesquelles des trous apparaissent dans les maillages, nous sommes partis de la construction des 256 configurations de cubes proposée par Lorensen et Cline [LC87]. La table initiale composée de 14 cas «de base» à partir desquels on construit tous les autres cas, ne présente que des cubes ayant au plus 4 sommets intérieurs à la surface (voir la figure 1.9 en page 22). Les configurations présentant plus de 5 sommets intérieurs sont obtenues en inversant les polarités. Une inversion de polarité a pour effet d’intervertir les sommets à l’intérieur et à l’extérieur de la surface (les sommets blancs deviennent noirs et les noirs deviennent blancs). La figure 3.9 montre l’exemple d’un cube avec 3 sommets intérieurs et son dual obtenu ainsi.

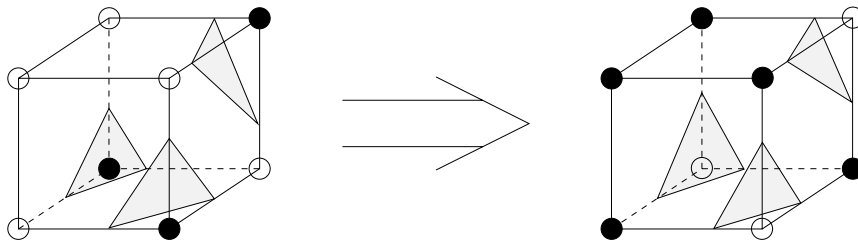


FIG. 3.9 – Inversion de polarité permettant d’obtenir un cube avec 5 sommets intérieurs

Au niveau des faces ambiguës, on constate que les topologies des maillages sont différentes. De ce fait lorsque deux cubes voisins partagent ce type de face, il se peut que les facettes de l’un et de l’autre ne partagent que les sommets mais pas les arêtes au niveau de cette face.

L’ensemble des 14 cubes de base sont cohérents entre eux. Cependant, le problème de l’inversion de polarité est qu’elle ne respecte pas le critère de cohérence que nous nous sommes fixé, à savoir que deux cubes voisins doivent présenter la même topologie au niveau de leur face commune. Sur la figure 1.12a en page 24, le cube du dessus est obtenu à partir de celui du dessous auquel on a fait subir une inversion de polarité et une rotation. C’est par ce type de construction

que l'on obtient des incompatibilités car l'inversion de polarité a pour conséquence de modifier la topologie du maillage au niveau des faces ambiguës.

La table des 256 cubes possibles doit présenter des facettisations compatibles entre elles, ceci n'est pas le cas avec la méthode de Lorensen et Cline.

C'est pourquoi il est nécessaire de s'affranchir des inversions de polarités pour construire l'intégralité des cubes. Pour cela il est donc nécessaire de considérer des cubes de base supplémentaires présentant de 5 à 7 sommets intérieurs à la surface. Ces cubes ajoutés doivent par ailleurs être cohérents topologiquement avec les 14 initiaux et ne peuvent donc pas être obtenus autrement qu'en les ajoutant.

Les cubes ajoutés présentant de 5 à 7 sommets intérieurs à la surface sont habituellement obtenus à partir des 7 cubes présentant de 1 à 3 sommets intérieurs. C'est la raison pour laquelle il y a 7 cubes ajoutés aux 14 initiaux.

Les surfaces facettisées au moyen de ces 21 cubes ne présentent pas de trous car les cubes sont construits de manière à ce que toute arête présente au sein de la face d'un cube soit utilisée pour construire un triangle dans chaque cube partageant cette face. Il n'y a donc pas d'arête «dans le vide» : toute arête du maillage fait obligatoirement partie de deux triangles exactement.

À partir de ces 21 cubes de base on peut donc construire l'intégralité des configurations nécessaires aux Marching Cubes sans avoir recours aux inversions de polarités (uniquement rotations et symétries). Il est à noter que comme certaines décompositions en tétraèdres ou comme la méthode consistant à évaluer le potentiel au centre des faces, notre méthode ne fournit pas forcément une facettisation dont la topologie coïncide exactement avec la réalité (le critère d'exactitude n'est pas pris en compte). Notre méthode présente également la particularité d'être globale à la surface : ceci peut être considéré comme un désavantage si l'on veut privilégier la fusion dans une zone de l'espace et la séparation ailleurs.

Néanmoins, certains détails propres à nos choix sont à remarquer :

- il n'y a plus aucun problème de continuité C^0 dans les surfaces triangulées (la cohérence est assurée) ;
- la méthode proposée ne nécessite aucun traitement supplémentaire afin de «boucher les trous» ;
- la «disparition» de facettes lors d'animations est bien plus visible et bien plus gênante que ces problèmes de topologie qui restent tous comptes faits, de l'ordre du détail dans la

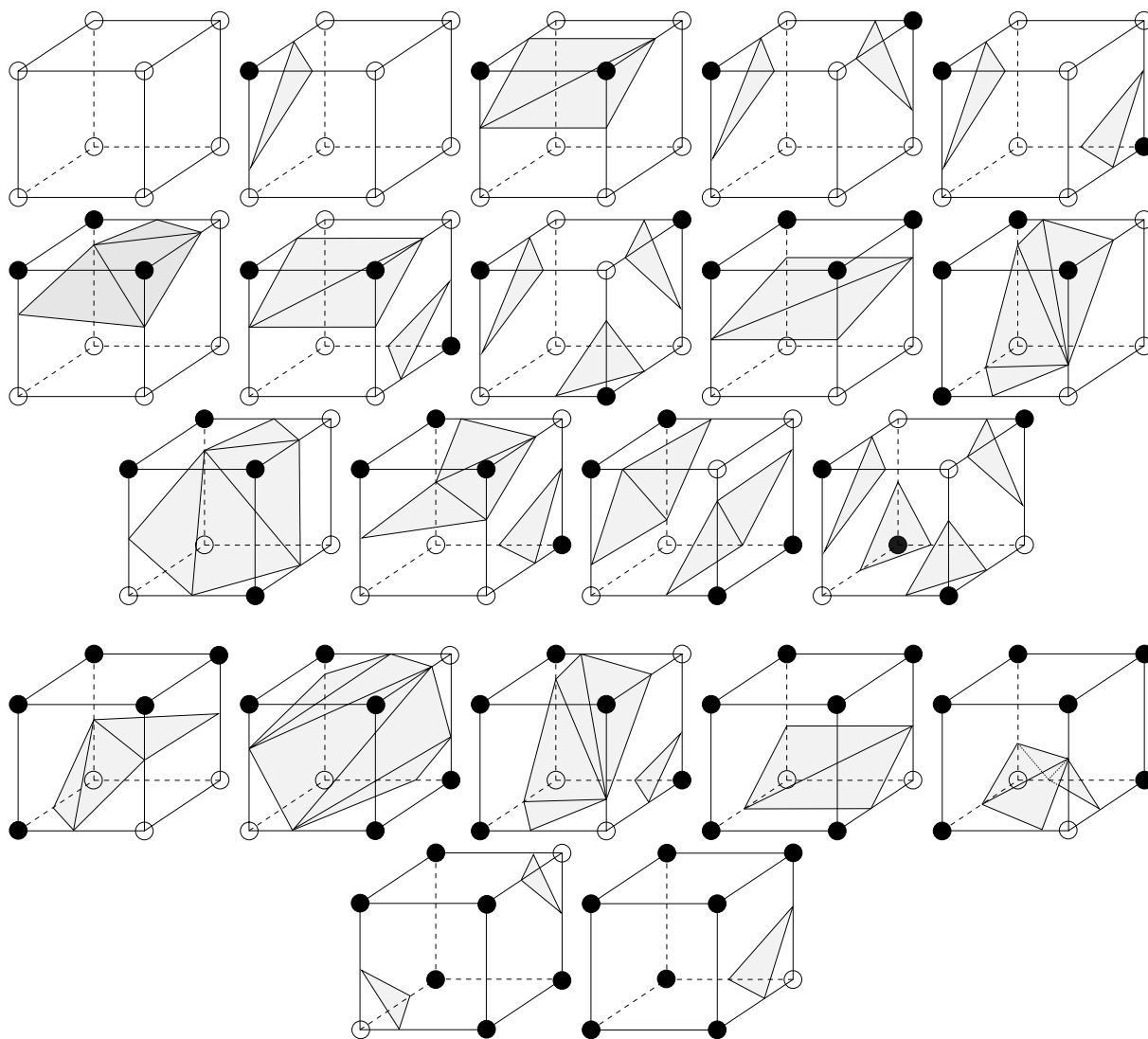


FIG. 3.10 – L’ensemble de nos cas de base

mesure où le but est de réaliser des animations présentant des surfaces qui bougent et se déforment.

Il est très gênant de voir des trous se former dans les surfaces. En revanche, étant dans un contexte d'animation et de rendu temps-réel où les surfaces bougent et se déforment beaucoup, il semble moins handicapant que les facettisations construites ne coïncident pas toujours exactement avec la topologie de la surface implicite théorique. Ceci n'est en fait acceptable que parce que la topologie des surfaces n'est affectée que *très* localement.

Remarque : *L'ensemble des cubes de base que nous avons présentés tendent à privilégier la séparation des surfaces au sein des cubes. Vous trouverez en annexe A une autre table qui tend à favoriser la fusion des surfaces. Le choix de l'une ou de l'autre des tables est laissé à l'utilisateur.*

3.5 Bilan

Toutes ces améliorations se concrétisent par une implantation sous forme de librairie. Notre librairie est utilisée comme outil de facettisation de surfaces implicites à squelette et peut donc servir à facettiser des corps très différents. Dans nos simulateur, nous l'utilisons aussi bien pour l'affichage d'écoulements sanguins que pour représenter certains corps organiques.

3.5.1 Besoins en mémoire

La plupart des optimisations présentées ci-dessus permettent de diminuer grandement les temps de calculs (voir le chapitre 5) principalement en mémorisant les résultats de certaines opérations. Il y a malheureusement un revers à cette médaille : avec une gestion un peu trop laxiste, les besoins en mémoire peuvent devenir prohibitifs.

3.5.1.1 Les tableaux dynamiques

Il y a de nombreux points où l'on a à gérer des problèmes d'allocation de mémoire :

- la grille de marching cubes contient les potentiels au niveau des nœuds de la grille ;
- la grille de voxels contient des listes chaînées de numéros de primitives ;

- la liste des triangles est décrite par une liste de points de l’espace et une liste de nombres décrivant l’ordre de ces sommets pour décrire la surface.

En plus de cela, nous avons recours au mécanisme d’horodatage pour de nombreuses données : en plus de la donnée elle-même nous devons mémoriser la date qui lui est associée.

À cause de tout ceci les besoins en mémoire peuvent devenir très pénalisants : une grille composée de 160^3 cubes nécessite 100 méga-octets de mémoire et une de 260^3 cubes environ 400 méga-octets ! Il est donc nécessaire de gérer efficacement la mémoire manipulée.

Dans l’implémentation de Bloomenthal parue dans les *Graphic Gems* [Blo94], on constate que certaines structures de données sont allouées de manière statique (à la compilation) et que d’autres sont allouées (et libérées) dynamiquement tout au long du programme.

De nombreuses structures de données ont des tailles très variables et dont l’évolution est imprévisible (lors de la compilation). C’est par exemple le cas pour les listes contenant les points de la surface ou la description des triangles. L’allocation statique présente alors l’inconvénient majeur contre lequel nous luttons à savoir cette forme de «laxisme» ou de gaspillage consistant à réserver plus de mémoire que nécessaire puisque l’on ne sait pas exactement de combien on a besoin.

Néanmoins, une gestion entièrement dynamique de la mémoire est, quand à elle, inconcevable dans un contexte temps réel. Les résultats présentés au chapitre 5 confirment cette idée que les fonctions `malloc` et `free` sont trop pénalisantes pour être utilisables dans notre contexte.

Il est donc indispensable de trouver un juste milieu entre ces deux extrêmes. Parmi toutes les structures de données que nous utilisons, aucune n’est allouée de manière statique. Ceci a été dicté par l’aspect modulaire que nous nous sommes imposés (voir le paragraphe 5.6 qui traite de la présentation sous forme de librairie).

Toutes les structures de données sont stockées dans des tableaux. Ces tableaux sont initialement vides et leur taille est fixée à quelques éléments¹⁵. Lorsque l’on y ajoute de nouvelles données et que la limite de capacité est dépassée, nous «multiplions» sa taille par deux¹⁶ afin de se doter d’une marge suffisante pour les besoins ultérieurs de l’algorithme.

15. 16 éléments

16. en utilisant la fonction `realloc` qui permet d’étendre une zone de mémoire réservée tout en conservant intact son contenu

Cette gestion est appliquée à tout tableau dont la taille n'est pas connue a priori. Le doublement à chaque fois que c'est nécessaire permet d'augmenter rapidement la taille et de vite atteindre un «régime de croisière». Ainsi les allocations (réallocations) de mémoire sont peu nombreuses car chaque structure de données arrive assez vite à sa taille «normale» et les appels au système sont en général regroupés au début de l'animation.

Notre proposition met en place une gestion de la mémoire à mi-chemin entre des allocations statiques et des allocations dynamiques d'un élément lorsque c'est nécessaire. Nous réservons plus de mémoire que nécessaire mais le nombre des allocations qui sont coûteuses en temps est minimisé.

3.5.1.2 Les tables de hashage

Dans [Blo94], la grille des Marching Cubes, c'est-à-dire la grille des potentiels, n'est pas stockée sous forme d'un tableau tridimensionnel et l'auteur a préféré utiliser une table de hashage.

Il existe deux formes de tables de hashage. La première repose sur une connaissance a priori de la quantité de données que l'on va y stocker : la table est alors surdimensionnée et la clé est choisie afin d'éviter les collisions entre les données. Ce type de table de hashage ne nous intéresse pas ici car nous cherchons à diminuer la quantité de mémoire nécessaire et une telle implémentation requiert en général une table dont la taille est le double du nombre d'éléments que l'on va y mettre.

C'est l'autre forme d'implémentation qui a été utilisée dans [Blo94]. Celle-ci utilise un tableau dont chaque entrée est la tête d'une liste chaînée d'éléments. La clé de hashage devra répartir les données le plus uniformément possible sur l'ensemble des entrées. Ici la taille du tableau importe moins que dans la première approche. Lors d'une collision durant un ajout dans la table, on va ajouter la donnée à la liste chaînée.

Le remplissage d'une telle table n'est pas particulièrement coûteux en soi (bien qu'un peu plus lent que l'accès à un simple tableau). Cependant nous n'utilisons pas ce type de structure de données car les accès en lecture sont intolérables : une lecture ou une mise à jour d'une donnée requiert un parcours de liste non ordonnée ! Certains essais nous ont démontré que le surcoût occasionné par ces parcours de listes est trop pénalisant au vu des gains en mémoire, c'est la raison pour laquelle nous utilisons pour l'instant un tableau permettant de mémoriser l'intégralité de la grille de cubes.

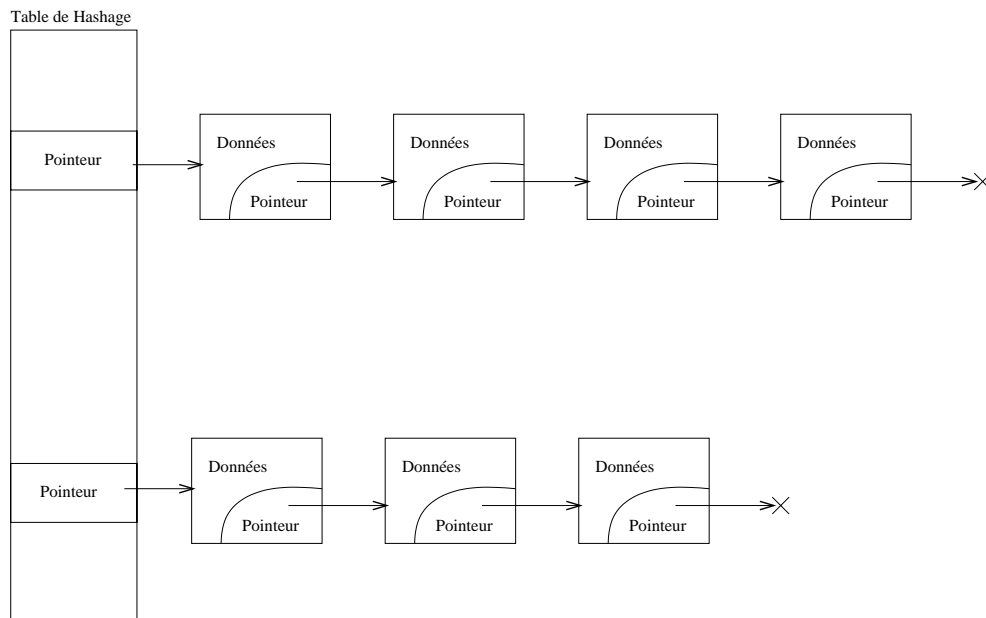


FIG. 3.11 – Le deuxième type de table de hashage

3.5.1.3 Comment économiser de la mémoire

Nous prévoyons de modifier notre implantation pour que le stockage en mémoire de l’intégralité de la grille ne soit plus nécessaire.

On peut voir ici un parallèle avec le remplissage de surface dans les logiciels de dessin. Notre méthode actuelle s’apparente à du *flood fill* qui consiste à colorier un pixel d’une image puis à propager récursivement le traitement aux pixels voisins. Les problèmes de mémoires que nous rencontrons sont sensiblement les mêmes que ceux rencontrés il y a 20 ans dans les logiciels de dessins. Une solution à ce problème a été le remplissage *scanline* : une seule ligne de l’image est stockée en mémoire et l’on procède à du flood fill sur cette ligne.

On ne gardera alors en mémoire qu’un seul plan de cubes (par exemple tous les cubes d’une cote donnée). Le traitement de la grille devra donc s’effectuer «tranche par tranche» (selon les cotes croissantes par exemple).

Le parcours des cubes se fera par voisinage (comme dans la version actuelle) mais en restreignant l’exploration des voisins utiles aux cubes du même plan. Les autres voisins utiles (ceux contenus dans les plans de cote différente) seront mémorisés dans une autre pile pour être traités à l’étape suivante (voir figure 3.12).

Il faudra, avec cette méthode, s'assurer qu'il ne reste aucun cube à facettiser dont la cote serait supérieure à celle du plan en cours de traitement. Pour cela, nous devons trouver et ordonner les cubes «de départ» (paragraphe 3.3.1, figure 3.2) de manière à ce qu'aucun «retour en arrière» ne doive avoir lieu. Les algorithmes de remplissage de surfaces de type scanline sont naturellement sujets à ce type de problèmes et la mise en parallèle des solutions sera intéressante à effectuer.

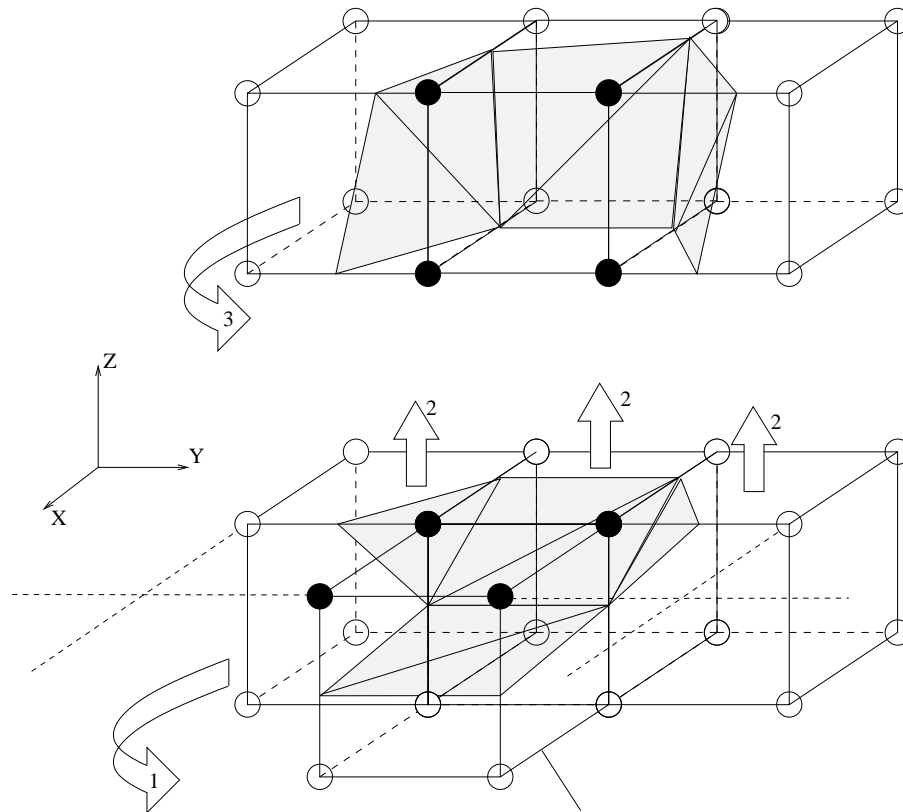


FIG. 3.12 – *Facettisation par «étages» : un cube de départ est trouvé «le plus bas possible» (cube nommé START), la facettisation est propagée à ses voisins utiles (flèche 1), à chaque fois on empile les cubes voisins dans la tranche du dessus pour un traitement ultérieur (flèches 2), une fois la tranche courante achevée, on traite les cubes qui viennent d'être empilés (flèche 3). Pour des raisons de clarté, les deux étages de cubes adjacents ont été disjoints sur la figure et seuls quelques cubes ont été représentés.*

3.5.2 Utilisation de la cohérence temporelle

Certains travaux tels que [DTG95, FV97, GA00, FCG00] s'appuient sur la détection de certaines parties du squelette n'ayant pas bougé entre deux appels de facettisation pour ne refacettiser que si c'est nécessaire. Les endroits où le squelette a été déplacé sont facettisés à nouveau alors que l'on réutilise les triangles de la facettisation précédente pour les portions de

surface immobiles.

Nous ne nous sommes pas beaucoup intéressés à ce type d’optimisation car notre but initial était d’animer des surfaces qui bougent et se déforment beaucoup au fil du temps. Nous nous sommes donc placés volontairement dans le cas où les surfaces doivent être intégralement facet-tisées à chaque fois. Les résultats obtenus étant satisfaisants, nous n’avons pas encore étudié en détail ce type d’optimisation.

Néanmoins il pourra être intéressant d’étudier ce problème pour adapter notre implémenta-tion au cas particulier de surfaces peu mobiles. La grille de voxels fournira alors des informations utiles pour déterminer rapidement si des mouvements de primitives ont eu lieu.

Les mécanismes à base de graines tels que [WH94, DTG95] sont fondés sur des traitements incrémentaux pour échantillonner les surfaces : une solution initiale est améliorée par étapes successives jusqu’à satisfaction d’un critère de qualité défini arbitrairement. Lors de petites dé-formations de la surface, il est donc possible de faire évoluer la facettisation précédente vers sa nouvelle position.

L’algorithme des Marching Cubes quant à lui ne tire profit d’aucun fonctionnement incrémental et la facettisation au sein d’un cube est à refaire entièrement même lorsque les potentiels ont très peu varié.

De ce fait il n’est pas impossible que les traitements supplémentaires pour détecter les portions de surfaces à re-facettiser ne nous coûtent plus en temps de calculs que ce qu’ils ne nous feraient gagner. Une étude approfondie de ce problème me semble intéressante et passera à mon avis par une adaptation des travaux de Max Froumentin et d’Éric Varlet [FV97] où il est question d’actualiser les potentiels des nœuds de la grille d’après les déplacements de primitives.

3.5.3 Les Marching Cubes adaptatifs

Les surfaces obtenues dans les simulateurs (voir le paragraphe 5.5) pour les écoulements sanguins ou les intestins ne présentent pas de zones où la courbure semble suffisamment faible pour permettre de réduire les maillages sans perte que la perte ne soit perceptible.

Il semble à première vue que l’utilisation actuelle que nous faisons des surfaces implicites dans notre simulateur ne tirerait pas beaucoup profit des raffinements présentés au paragraphe présenté en page 22. C’est pourquoi nous ne nous sommes pas intéressés en détail à des facettisations

basées sur des grilles adaptatives. Une étude du problème serait néanmoins intéressante dans le but d'ouvrir notre bibliothèque à cette amélioration de l'algorithme des Marching Cubes.

Bien que toutes ces améliorations seront étudiées prochainement nous avons préféré nous concentrer sur d'autres problèmes tels que le contrôle du blending.

Conclusion

Notre implémentation présente de nombreux points où l'algorithme des Marching Cubes a pu être optimisé voire rendu plus robuste en ce qui concerne les surfaces mal facettisées présentant des trous.

Les temps de facettisation obtenus (voir le chapitre 5) nous permettent d'habiller des modèles mécaniques relativement complexes. Par exemple, le traitement d'un squelette composé de 200 primitives prend moins de 25ms sur un ordinateur de gamme moyenne.

Dans le chapitre suivant, nous présentons les problèmes liés au blending des surfaces et plus particulièrement au contrôle de ce blending. Dans le cadre de l'algorithme des Marching Cubes, nous proposons une méthode permettant de contrôler les zones du squelette où les primitives peuvent fusionner entre elles et une méthode inspirée de [Gas93, OC97] afin de propager les déformations causées par la compression des surfaces les unes contre les autres.

Chapitre 4

Le contrôle du blending

Sommaire

4.1	Exposé du problème	90
4.2	Désactivation du blending	91
4.2.1	Solutions de la littérature	92
4.2.1.1	Structuration du squelette	92
4.2.1.2	Altérations des calculs de potentiels	92
4.2.1.3	Des objets disjoints	93
4.2.1.4	Le cas des auto-collisions	94
4.2.2	Solutions intuitives	96
4.2.2.1	Le cas des auto-collisions	96
4.2.3	Notre solution pour les Marching Cubes	99
4.2.4	Limites de la solution	101
4.3	La propagation des déformations	103
4.3.1	Solutions de la littérature	104
4.3.2	Notre solution pour les Marching Cubes	106
4.3.3	Limites de la solution	107

4.1 Exposé du problème

Le blending, cette particularité qu'ont les surfaces implicites à squelette de fusionner entre elles (présentée au paragraphe 1.2.3), peut s'avérer gênante lorsque certaines surfaces fusionnent alors qu'il en serait autrement dans la réalité. L'exemple le plus couramment employé est celui du bras d'un personnage qui ne devrait normalement fusionner qu'avec l'épaule mais qui fusionne aussi avec la cuisse ou le tronc (voir figure 4.1).

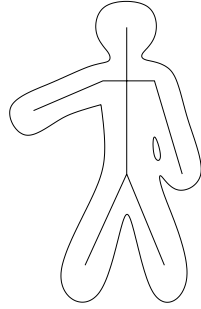


FIG. 4.1 – *Blending involontaire entre le bras et la cuisse*

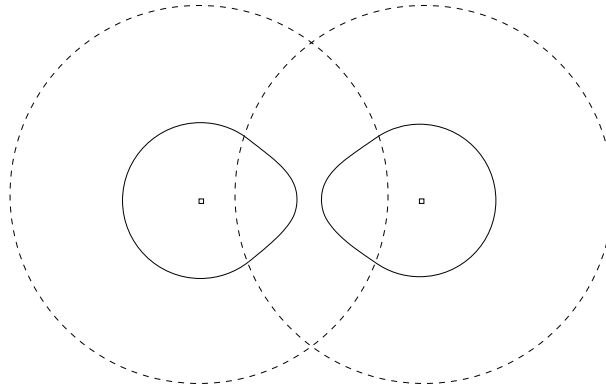
Ce problème est lié à l'utilisation de surfaces implicites pour lesquelles on somme les potentiels générés par différentes primitives. Il est nécessaire de contrôler quelles primitives peuvent fusionner entre elles.

Dans notre cas, ce contrôle du blending intervient à deux niveaux. D'une part, il s'agit de prendre en compte des corps dont la topologie reste la même au cours du temps, généralement des organes. D'autre part, la topologie de certains corps peut évoluer au cours du temps : c'est le cas des fluides formant des flaques qui se scindent et se réunissent à nouveau.

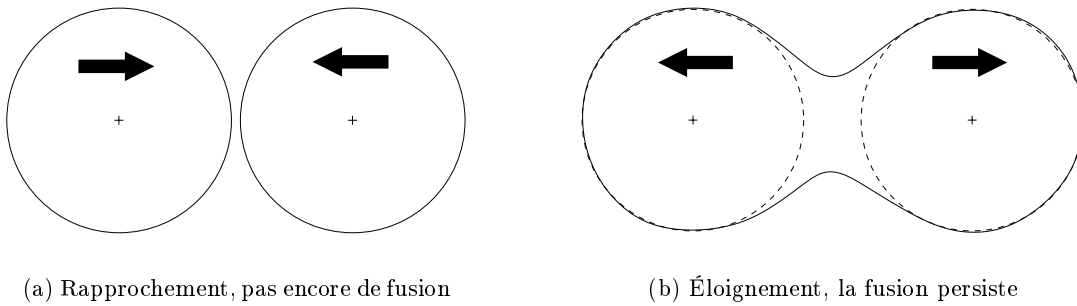
D'une part il s'agit donc d'empêcher un corps de fusionner avec lui-même en dehors des endroits où cela a été autorisé lors de la modélisation : ceci est nécessaire lors de la représentation d'organes tels que des intestins, des trompes ou tout autre organe très déformable.

Et d'autre part, lorsque des corps de même nature et dont la topologie est très variable (ils peuvent fusionner ensemble) se rapprochent et que la surface de l'un entre dans la zone d'influence de l'autre, cette surface est déformée et semble «attirée» (figure 4.2). Cette attraction n'a aucune justification dans notre contexte et nous voulons donc l'éviter.

Ici ce n'est plus la modélisation initiale qui règle le blending : c'est le moteur mécanique prenant en charge les déplacements qui va préciser entre quels couples de primitives peuvent avoir lieu les fusions.

FIG. 4.2 – *Déformation de deux surfaces avant le contact.*

Les positions des primitives ne sont d'ailleurs pas suffisantes pour déterminer correctement le comportement de la fusion. Un exemple concret est celui de deux gouttes de liquide qui, lorsqu'elles se rapprochent l'une de l'autre, ne doivent pas commencer à fusionner (et donc se déformer) tant qu'elles ne sont pas en contact, alors que lorsqu'elles s'éloignent, la capillarité fait qu'elles fusionnent encore pendant un certain laps de temps (voir figure 4.3).

FIG. 4.3 – *Fusion lors d'une animation de fluide*

4.2 Désactivation du blending

Nous présentons ici des solutions apportées au problème de contrôle du blending. Nous commençons par ce que différents auteurs ont proposé et nous étudions ensuite les différentes approches sur lesquelles nous nous sommes attardés. Ces approches sont présentées ici sous le qualificatif «intuitives», non pas dans le sens où elles seraient dépourvues d'intérêt mais plutôt car ce sont les premières idées que nous avons eues. Elles permettent de bien cerner l'ensemble des problèmes. Enfin nous présentons une solution plus robuste que nous avons retenue.

4.2.1 Solutions de la littérature

4.2.1.1 Structuration du squelette

Dès 1989, Brian et Geoff Wyvill abordent le problème dans [WW89] : leur idée est alors de regrouper les primitives en leur associant un numéro de groupe. Seules les primitives d'un même groupe sont autorisées à fusionner ensemble. Ceci permet donc d'éviter la fusion entre des surfaces normalement disjointes.

Cette solution n'est cependant pas suffisante car elle ne se résume en fait qu'à dissocier les primitives représentant des objets disjointes en les plaçant dans des groupes différents. On obtient exactement le même résultat en traitant d'abord un objet puis le second lors d'une seconde facettisation.

Le personnage de la figure 4.1 étant d'un seul morceau, il est impossible de le représenter correctement : les primitives des bras, du tronc et des jambes doivent toutes appartenir au même groupe pour que l'on puisse avoir les fusions au niveau des épaules et des hanches, et donc le bras continue de fusionner avec la cuisse.

L'idée de Wyvill, fondée sur une répartition des primitives dans des groupes disjointes, a été approfondie dans [OM93, GW95] pour évoluer vers l'utilisation d'un graphe de voisinage décrivant la structure de l'objet : les nœuds du graphe sont les primitives et les arcs relient les primitives autorisées à fusionner ensemble.

4.2.1.2 Altérations des calculs de potentiels

La structure que l'on a donnée au squelette de la surface doit être prise en compte dans l'évaluation du potentiel. On ne peut effectivement pas sommer la contribution de chaque primitive du squelette pour obtenir le potentiel d'un point.

Dans [DTG95], l'évaluation du potentiel en un point P devient :

1. calculer au point P le potentiel engendré par chaque primitive ;
2. choisir le potentiel prépondérant (maximal) f_i associé à la $i^{\text{ème}}$ primitive ;
3. lui ajouter la contribution maximale des groupes de primitives qui fusionnent ensemble et qui, dans le graphe, sont voisines de la i^{e} primitive ;

4. renvoyer la valeur obtenue.

Lorsque l'on passe de la zone de prépondérance d'une primitive à la zone de prépondérance d'une primitive voisine, les groupes de primitives qui fusionnent ne sont pas nécessairement les mêmes. De ce fait, ce calcul provoque des discontinuités dans le champ de potentiel. On peut alors observer des discontinuités dans les surfaces.

Pour corriger ceci, l'évaluation du potentiel a été améliorée dans [CD97, Can98] : c'est le maximum des potentiels des groupes qui est retenu. Il n'y a plus de discontinuité dans les surfaces obtenues [GW95], mais la méthode ne garantit pas de continuité C^1 en tout point de la surface.

Dans ces deux articles [DTG95, CD97], l'affichage des surfaces se fait grâce à la méthode des graines présentée sur la figure 1.21 en page 31 (méthode décrite dans l'article).

4.2.1.3 Des objets disjoints

Un premier cas de figure envisageable est celui où deux corps disjoints entrent en contact. Afin d'empêcher la fusion des deux surfaces, nous devons éviter de sommer les potentiels de leurs primitives respectives.

On effectue ce type de déformation en agissant directement sur le modèle implicite [Gas93]. Une surface a que l'on veut creuser au moyen d'une surface b est obtenue en modifiant la fonction de potentiel comme suit :

$$G_{ab}(x,y,z) = F_a(x,y,z) - F_b(x,y,z) + \text{isovaleur} \quad (4.1)$$

où F_a (respectivement F_b) est la fonction de potentiel associée à la surface a (resp. b) et G_{ab} celle associée à la surface désirée (c'est-à-dire la surface a déformée par b). L'idée d'utiliser des primitives dont la contribution est négative pour «creuser» une surface avait été proposée initialement dans [Bli82].

Les calculs des potentiels sont alors modifiés afin d'obtenir les compressions nécessaires au niveau des zones de contact.

Les deux surfaces étant disjointes, nous nous trouvons dans le même cas de figure que celui auquel se ramenaient les frères Wyvill dans [WW89]. Ils plaçaient les primitives formant des objets disjoints dans des groupes différents.

L'idée la plus simple et la plus efficace pour traiter de telles surfaces est de procéder à autant de facettisations qu'il y a de surfaces disjointes. Dans cette configuration, nous allons donc facettiser la première surface dont le potentiel sera altéré par la présence de la seconde, puis lors d'une seconde facettisation nous inverserons les rôles (en facettisant la seconde surface creusée par la première). On est ainsi assuré que les facettisations d'objets disjointes sont bien dissociées.

Une telle configuration ne pose pas véritablement de problème et la solution apportée n'alourdit que très peu les traitements.

4.2.1.4 Le cas des auto-collisions

Dans le cas d'un objet entrant en contact avec lui-même, étudions les résultats obtenus en appliquant les calculs de potentiels de [CD97] dans le cadre d'un affichage par Marching Cubes.

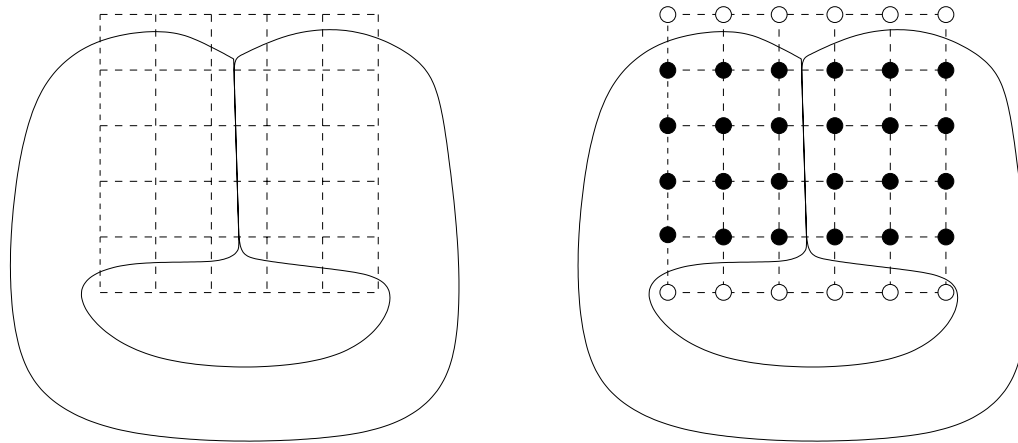
Plaçons-nous dans le cas d'une surface connexe allongée dont les deux extrémités entrent en contact (et qui ne fusionnent pas ensemble). La figure 4.4 montre la surface réelle, les potentiels des sommets, la facettisation que l'on obtient et celle que l'on voudrait obtenir. Comme le montre la figure 4.5 sur un agrandissement, la facettisation obtenue diffère de celle désirée au niveau des cubes où les deux surfaces sont présentes.

Étant donnée la configuration de ce cube, il est impossible d'avoir la facettisation voulue. En effet, comme on retient le potentiel prépondérant, on attribue aux sommets de gauche les potentiels générés par la portion de gauche du squelette. De même, le potentiel des sommets de droite dépend de la partie de droite du squelette.

Ainsi, sur la figure 4.5b, les deux sommets du bas sont considérés à l'intérieur de la surface et l'algorithme n'a pas à chercher de point d'intersection le long de l'arête du bas.

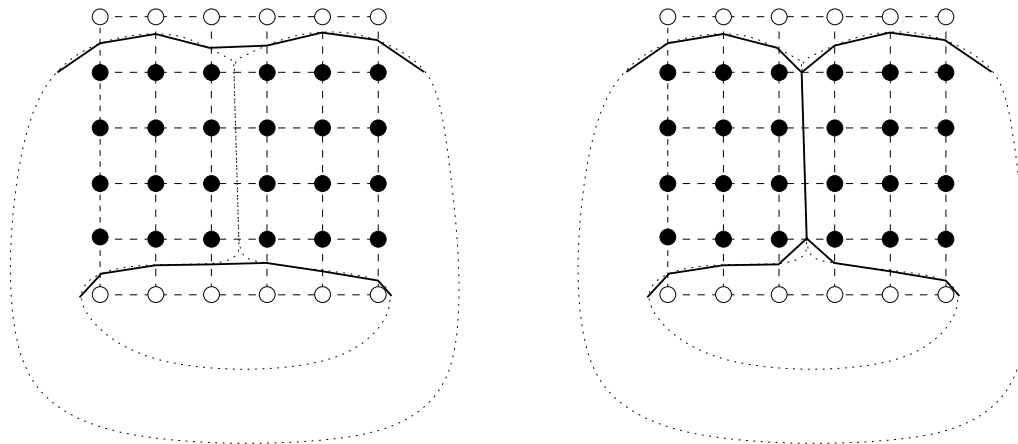
Bilan

Les différentes solutions présentées attestent que le contrôle du blending est étroitement lié à la méthode d'affichage. De plus, nous venons de constater que ces solutions, bien qu'efficaces lors d'un lancer de rayons ou d'une facettisation par graines, ne s'appliquent pas bien à l'algorithme des Marching Cubes. C'est pourquoi nous avons cherché une méthode spécifique.



(a) Surface réelle et grille de cubes

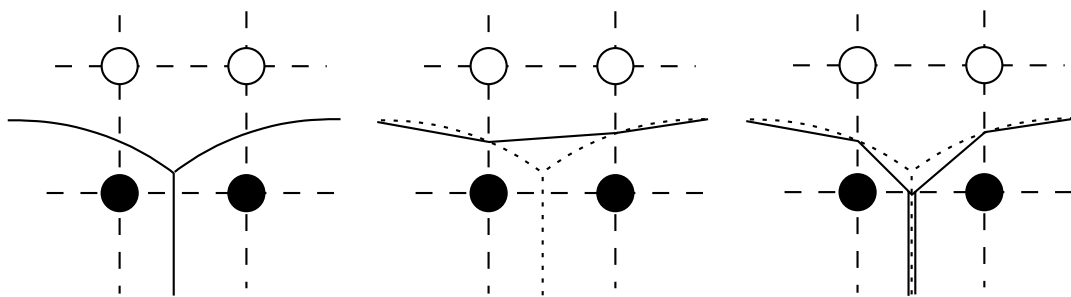
(b) Cubes utiles avec les potentiels calculés aux sommets



(c) Facettisation obtenue

(d) Facettisation désirée

FIG. 4.4 – Contrôle du blending avec des Marching Cubes



(a) Surface réelle

(b) Facettisation obtenue

(c) Facettisation désirée

FIG. 4.5 – Agrandissement du problème

4.2.2 Solutions intuitives

Dans cette partie, nous présentons différentes solutions visant à contrôler le blending dans le cas particulier des Marching Cubes.

4.2.2.1 Le cas des auto-collisions

La difficulté majeure réside dans le traitement de corps d'un seul morceau et dont une certaine topologie doit être préservée comme c'est le cas pour le personnage présenté en figure 4.1.

Dans le cas du personnage il est possible cependant de se ramener à une telle décomposition. En scindant le personnage au niveau de l'épaule par exemple, on arrive à dissocier le bras du reste du corps. On parvient ainsi à retrouver à peu près une décomposition comme au paragraphe précédent.

Ayant cette pseudo-décomposition on peut alors tenter une facettisation par morceaux voisine de celle proposée au paragraphe précédent. Le problème des raccords entre ces surfaces reste épineux. Dans [CGS98], B. Crespin propose, dans le cadre d'une facettisation par graines (présentée au paragraphe 1.3.3.2), une solution permettant de construire les portions de facettisation qui manquent au niveau des zones de blending.

Malheureusement, le nombre de primitives que nous animons est important et les squelettes manipulés ont souvent des topologies complexes ce qui crée de nombreuses zones de blending. Ce traitement très calculatoire n'est donc pas à notre portée dans un contexte temps-réel.

Une approche que nous avons envisagée pour réaliser la facettisation par morceaux est différente : l'idée que nous avons élaborée consiste à définir certaines primitives comme étant des « pivots » c'est-à-dire l'endroit où a lieu la fusion entre deux (ou plus) objets étant disjoints ailleurs (voir figure 4.6). Ce pivot est affecté à chacune des parties du squelette qu'il met en relation : la primitive de l'épaule est à la fois dans le groupe de primitives composant le bras *et* dans celui composant le tronc. On arrive ainsi à retrouver le même type de partitionnement du squelette que chez Wyvill. On peut alors facettiser séparément les différents « morceaux » du squelette (figures 4.6b et 4.6c).

Les objets que nous représentons sont opaques et constitués de surfaces fermées. En d'autres termes, on peut se permettre certaines libertés concernant la facettisation tant que cela n'est pas visible car caché à l'intérieur du volume (voir figure 4.6).

Nous laissons tels quels les maillages obtenus car ils tendent à se fondre correctement l'un dans l'autre : en «superposant» les maillages on obtient en effet les résultats attendus lors de la modélisation (figure 4.6d). Cette méthode basée sur le recouvrement de maillages a également été adoptée par Marie-Paule Cani et Samuel Hornus [CH01] dans le cadre des surfaces de convolution (voir page 14) construites autour de courbes à subdivision. Ils facettent séparément chaque squelette, dans les zones de branchements multiples entre plusieurs squelettes les facettes se mélangent correctement les unes avec les autres.

Dans cette première solution que nous pourrions proposer, les raccords entre les différentes parties du squelette sont donc effectués de manière assez peu élégante (que ce soit au niveau de la cohérence entre la véritable surface et la facettisation que l'on en fait¹⁷ ou tout simplement d'un point de vue algorithmique) mais qui ne requiert aucun traitement supplémentaire.

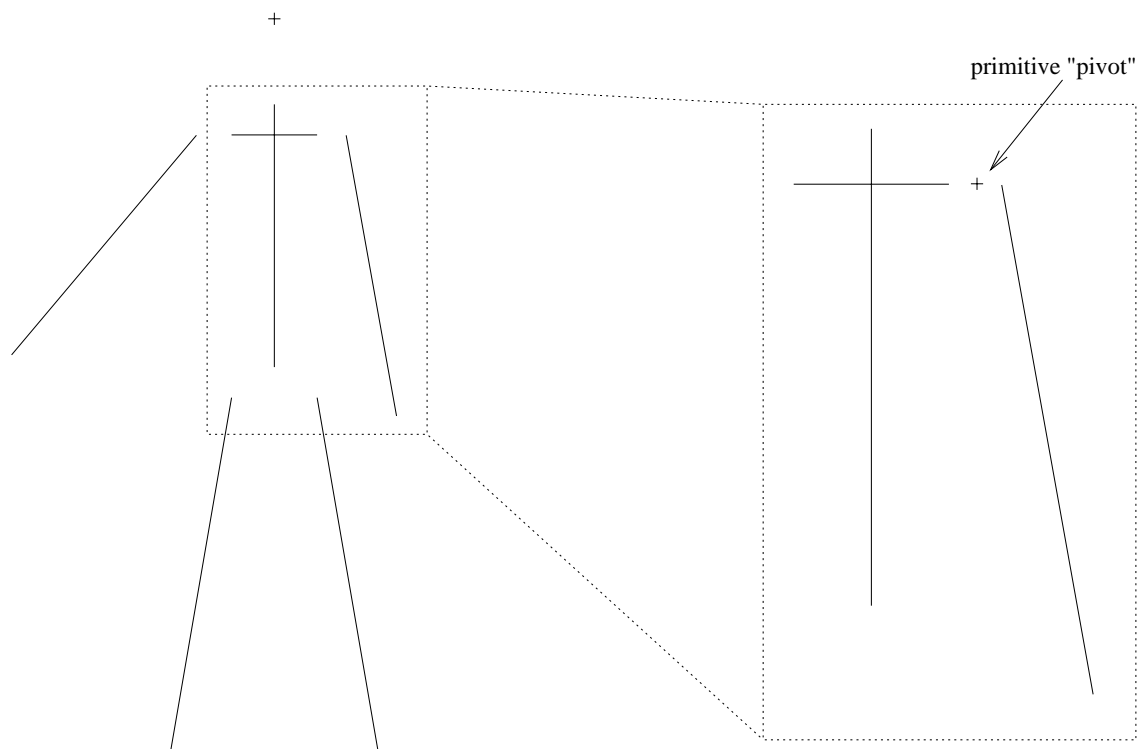
Pour que le personnage soit correctement représenté il faudra déterminer dans le squelette plusieurs endroits au niveau desquels effectuer ce type de fractionnement : au niveau des épaules afin d'isoler les bras du reste du corps, et au niveau du bassin (figure 4.6e).

Les résultats obtenus pour des cas simples comme un personnage sont satisfaisants. Cependant, c'est une solution extrêmement limitée qui ne donne de bons résultats que pour ce genre de cas simples.

Cette méthode présente de gros défauts. Nous avons décidé de ne pas poursuivre dans cette voie pour les raisons suivantes :

- les pivots doivent soit être connus à l'avance (lors de la modélisation) soit être déterminés «manuellement», notre but est d'avoir un traitement moins exigeant que cela ;
- selon le corps représenté, une seule primitive pivot n'est pas forcément suffisante et le nombre de pivots nécessaires (au niveau d'une même zone) pour construire un lien entre deux parties de squelette doit lui aussi être déterminé ;
- l'utilisation de trop peu de pivots (au niveau d'une même zone) conduit à de graves discontinuités C^1 dans le maillage final ;
- l'utilisation de trop de pivots conduit à ce que des primitives fusionnent alors qu'elles ne le devraient pas ;
- les corps modélisés ne présentent pas nécessairement de pivot, c'est par exemple le cas lorsque l'on replie un intestin sur lui-même (voir figures 5.12 et 5.17 en pages 122 et 129).

17. avec une telle facettisation il ne suffit pas de sommer l'aire de chaque triangle pour obtenir une approximation de l'aire de la surface



(a) Le squelette et un agrandissement où l'on a inséré une primitive «pivot» au niveau de l'épaule

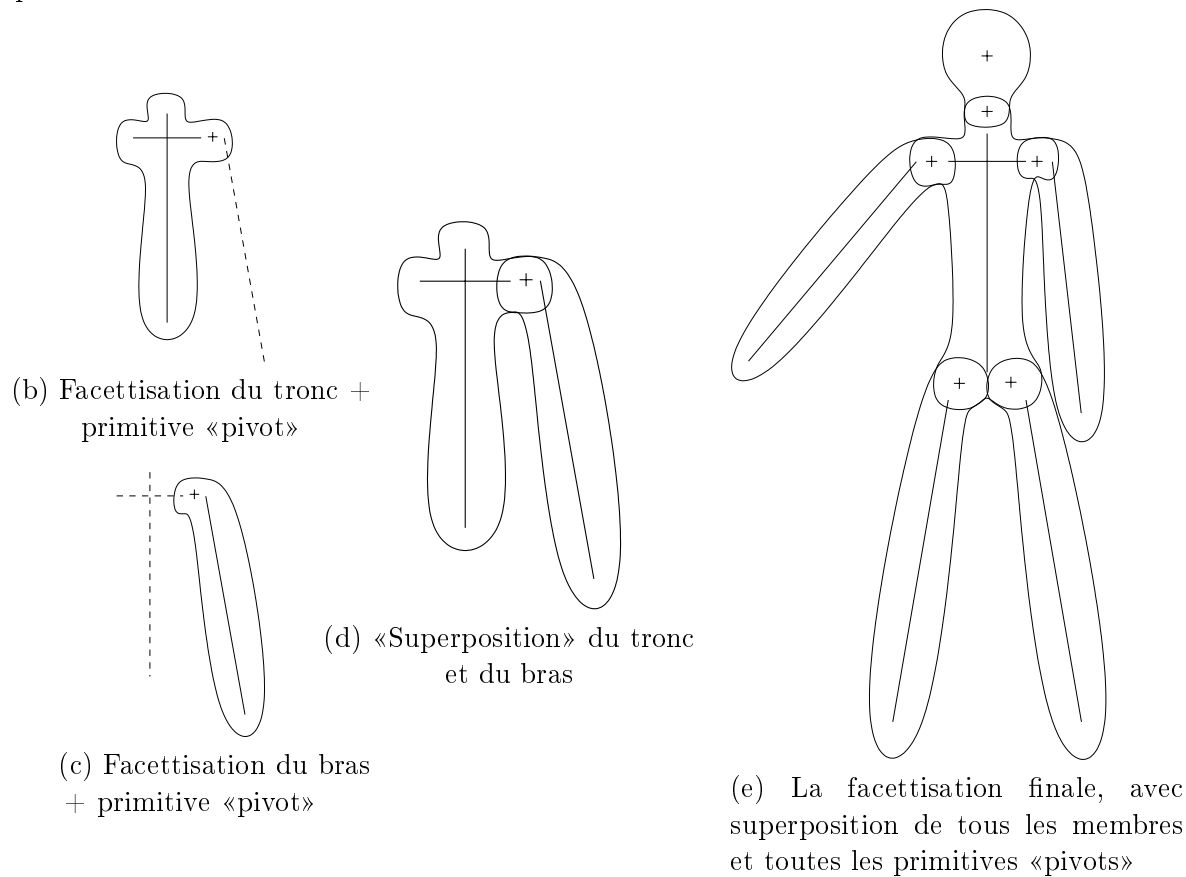


FIG. 4.6 – Facettisation par morceaux pour contrôler le blending

Nous voulons contrôler le blending aussi bien pour un simple personnage (peu déformable) dont le bras ne doit pas fusionner avec la cuisse, que pour un organe très déformable (un intestin par exemple) dont la topologie est fixe, que pour une flaque de sang dont les gouttes constamment en mouvement ne peuvent fusionner que lorsque le modèle mécanique sous-jacent détermine qu'elles sont suffisamment proches.

Pour les méthodes à base de graines où l'on sait à quelle primitive une graine se rapporte, on peut déterminer facilement la valeur de la fonction de potentiel. Cependant, le problème lors de l'évaluation d'un potentiel dans la grille de Marching Cubes est que l'on ne sait pas a priori à quelle primitive on se rapporte. En plus de cela nous avons vu sur la figure 4.5 qu'il est impossible d'obtenir une facettisation correcte si les sommets du bas de la figure sont considérés tous les deux à l'intérieur de la surface. Ces deux sommets sont pourtant *tous les deux* à l'intérieur de la surface !

Il y a une nuance à apporter à cette affirmation : le sommet de gauche est effectivement à l'intérieur du morceau de gauche mais il se trouve à l'extérieur de la surface de droite (et inversement pour le sommet de droite). Selon la portion de la surface à laquelle on s'intéresse, un sommet peut donc être à la fois intérieur *et* extérieur !

Il est donc impératif de prendre ceci en compte à un moment donné du traitement des Marching Cubes. C'est ce que nous allons détailler dans la partie qui suit.

4.2.3 Notre solution pour les Marching Cubes

Il nous semble incontournable de nous appuyer sur une structure de graphe dont les arcs représentent la fusion entre les primitives du squelette (concept décrit dans [OM93, GW95] et présenté au paragraphe 4.2.1). Reste à savoir comment utiliser les informations qu'il fournit afin de traiter correctement les cubes pour obtenir la facettisation désirée.

Nous avons souligné précédemment qu'il était impossible d'obtenir une facettisation correcte pour la configuration des figures 4.4 et 4.5; ceci parce que l'on affecte une unique valeur de potentiel aux sommets des cubes alors que selon la portion de surface considérée ces points peuvent être à la fois à l'intérieur et à l'extérieur.

Nous devons donc savoir par rapport à quelle portion de surface les calculs de potentiels sont effectués. De plus, nous devons élire tour à tour chaque portion de surface afin d'attribuer aux sommets des cubes leurs différentes valeurs de potentiels.

Pour cette raison, nous devons traiter (facettiser) plusieurs fois un même cube lorsque cela est nécessaire, c'est-à-dire lorsque le cube contient des portions de surfaces différentes.

Nous procédons à un partitionnement du squelette selon la relation de mélange. Localement à un cube, on peut généralement considérer que le mélange entre primitives est une relation d'équivalence¹⁸. La facettisation d'un cube se fait ainsi :

- construire les classes d'équivalence des primitives influentes sur le cube ;
- pour chacune de ces classes, facettiser le cube en considérant cet ensemble de primitives comme principal.

Les optimisations que nous avons apportées à l'algorithme des Marching Cubes (voir le paragraphe 3.3) vont nous permettre de simplifier ces traitements particuliers et surtout de ne pas trop ralentir la facettisation.

La construction des classes d'équivalence que nous effectuons passe par la construction d'une liste de primitives. Cette liste contient une et une seule primitive par classe d'équivalence.

La liste de ces primitives est construite grâce au contenu des voxels (voir le paragraphe 3.2.2 pour la présentation de ces voxels). Elle contient l'ensemble des primitives susceptibles d'engendrer une portion de surface dans le cube. À partir des informations contenues dans nos voxels nous pouvons donc construire cette liste qui est composée de toutes les primitives ayant de l'influence sur le cube.

Quand nous construisons cette liste (au début du traitement d'un cube), nous n'y ajoutons que des primitives qui ne fusionnent pas avec des primitives déjà présentes. La liste est donc minimale, ce qui évite de facettiser deux fois le cube pour deux primitives qui fusionnent (et pour lesquelles une seule facettisation est nécessaire puisqu'elles appartiennent à la même classe d'équivalence).

L'optimisation de l'horodatage consistant à éviter d'effectuer plusieurs fois les mêmes calculs en les stockant et en leur attribuant une date (paragraphe 3.3.3) nécessite une attention particulière. En effet cette optimisation a été mise en place pour éviter de calculer plusieurs fois un même potentiel ou la position d'un point d'intersection entre la surface et les arêtes des cubes. De ce fait, il est indispensable d'*invalid*er les dates du cube après le traitement de chaque primitive de la liste. Cette invalidation se fait très simplement en remettant à zéro les différentes dates concernées par le cube.

18. ce n'est cependant pas toujours vrai et nous présentons plus loin les problèmes que cela pose

4.2.4 Limites de la solution

Bien qu'elle donne de très bons résultats (voir figure 5.15 en page 126), notre proposition présente un défaut que nous allons expliquer ici.

Prenons le cas d'un squelette formé de 4 primitives A , B , C et D . Chacune de ces primitives ne fusionne qu'avec son ou ses voisins directs comme cela est illustré sur la figure 4.7

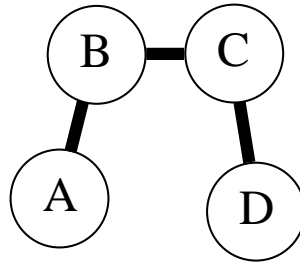


FIG. 4.7 – Squelette composé de 4 primitives A , B , C et D : les arcs du graphe indiquent quelles primitives fusionnent ensemble

Lorsque ces quatre primitives sont présentes dans un même cube, la relation de mélange n'est plus transitive. De ce fait, ce n'est plus non plus une relation d'équivalence. Le partitionnement du squelette ne peut donc plus être effectué

Lors du traitement d'un cube où ces quatre primitives sont présentes nous allons construire la liste des primitives pour notre traitement spécifique en y ajoutant tour à tour A , B , C et enfin D . Plus précisément, nous n'ajouterons ces primitives que si la liste ne contient pas d'autre primitive avec laquelle une fusion est autorisée par le graphe.

La primitive B ne sera pas ajoutée car elle fusionne avec la primitive A déjà présente dans la liste. En revanche, C sera ajoutée car il n'y a pas de fusion entre A et C . Pour la même raison que pour B , la primitive D n'est pas ajoutée à la liste puisqu'elle fusionne avec C .

La liste obtenue est donc finalement : [A , C]

L'ordre dans lequel sont définies les primitives ne dépend que de l'utilisateur. Il est donc tout à fait possible que l'objet soit modélisé exactement de la même manière mais en énumérant les primitives dans un autre ordre. Le tableau qui suit montre trois exemples d'énumération et le résultat obtenu dans la liste.

Ordre d'énumération des primitives	Contenu de la liste finale
A B C D	A C
A B D C	A D
B A C D	B D

Le contenu de la liste finale n'est pas le même pour les différents ordres proposés dans l'exemple et les calculs de potentiels en sont affectés. Par exemple, le premier cas donnera lors du traitement de la primitive A comme fonction de potentiel :

$$f = f_A + f_B - f_C - f_D \quad (4.2)$$

car A fusionne avec B mais pas avec C ni D .

Le tableau suivant présente les différentes fonctions de potentiel utilisées lors du traitement de chaque primitive de la liste en fonction de leur ordre d'énumération.

Ordre d'énumération des primitives	Contenu de la liste finale	Fonctions de potentiel	
		1 ^{re} primitive	2 ^e primitive
A B C D	A C	$f_A + f_B - f_C - f_D$	$f_C + f_B + f_D - f_A$
A B D C	A D	$f_A + f_B - f_C - f_D$	$f_D + f_C - f_B - f_A$
B A C D	B D	$f_B + f_A + f_C - f_D$	$f_D + f_C - f_B - f_A$

On se rend compte que parmi ces trois exemples aucun ne va produire exactement les mêmes calculs de potentiels. De ce fait les facetisations générées peuvent être différentes.

À la question «quelle facetisation est correcte?» on pourrait répondre par cette autre question: «existe-t-il vraiment une facetisation correcte?». On peut en effet se demander s'il est possible de déterminer si l'une ou l'autre des facetisations correspond effectivement à la véritable surface.

Certains tests nous ont montré que ces problèmes de facetisations ne sont perceptibles que lorsque la surface est sous-échantillonnée (voir figure 4.8): certaines portions de surfaces correspondant à la primitive choisie ont une largeur inférieure à la taille d'un cube. La résolution choisie pour la grille des Marching Cubes n'est pas suffisante et il est nécessaire de l'augmenter si l'on veut s'affranchir de ces problèmes.

Selon l'utilisation que l'on veut faire des surfaces facetisées, ceci peut être plus ou moins

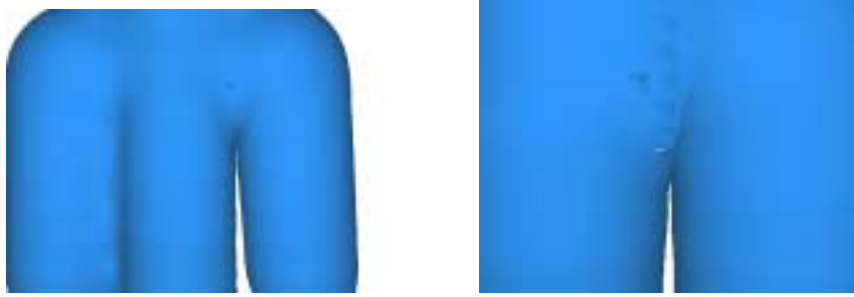


FIG. 4.8 – *Zoom sur les discontinuités de potentiels et donc de surface*

gênant. Lors d’animation de liquides qui bougent et se déforment, le simple mouvement continu des primitives représentant les gouttes va faire que ces artéfacts très localisés ne seront visibles que durant de très brefs instants.

Pour finir, il est important de noter que ces problèmes ne se posent que pour des configurations très particulières dans lesquelles on veut rendre compte de formes complexes au moyen de primitives très proches les unes des autres. Les problèmes que l’on a sont dus au fait que des primitives qui ne fusionnent pas ensemble se retrouvent trop proches les unes des autres : le modèle mécanique sous-jacent aurait dû contrôler cela et empêcher le rapprochement ou bien autoriser le blending pour certains couples de primitives.

4.3 La propagation des déformations

Étudions à présent un problème connexe à celui du contrôle du blending : lorsque deux objets déformables entrent en contact leurs surfaces ne doivent pas s’interpénétrer et au contraire il faut qu’elles se déforment afin de s’adapter l’une à l’autre. De plus, dans le cas de corps peu compressibles il se forme généralement un renflement autour des zones compressées. C’est la mise en œuvre de ces déformations particulières que nous allons présenter ici.

Lorsque l’on applique une pression sur un corps mou comme un organe (très peu compressible), il tend à conserver son volume initial en se bombant autour de la zone de compression. Ce type de déformation est typiquement du ressort du moteur mécanique. Toutefois, nous allons proposer une méthode géométrique fournissant des déformations réalistes pour un coût inférieur à leur prise en compte au niveau mécanique. Ce que nous apportons ici est à classer dans la catégorie des «effets spéciaux» présentés en page 58.

4.3.1 Solutions de la littérature

Marie-Paule Cani [Gas93] donne une expression implicite du contact entre surfaces et un modèle de calculs permettant d'obtenir des renflements autour des zones comprimées.

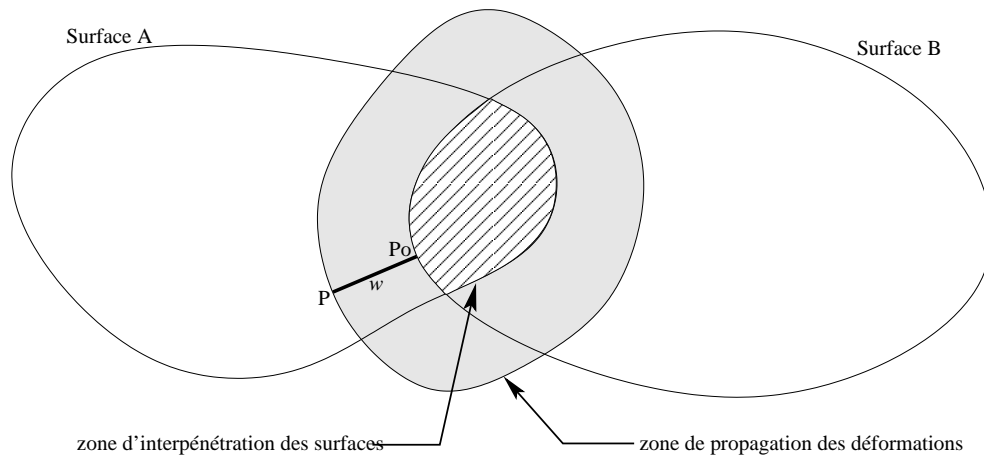
Lorsqu'une interpénétration est détectée, on déforme chaque objet concerné en fonction de l'ensemble des corps entrant en jeu. Dans la zone d'interpénétration une surface de contact est déterminée en prenant l'équation 4.1 et son symétrique en a et b . Cette construction par symétrie fournit donc une surface de contact définie par les points où $F_a = F_b$. Autour de cette zone d'interpénétration les potentiels des corps sont augmentés d'un certain terme qui dépend de plusieurs variables : l'épaisseur w de cette zone de propagation des déformations, un facteur α d'atténuation déterminant l'épaisseur du renflement. Le paramètre de ce terme de déformation est la distance entre le point considéré et la zone d'interpénétration. Cette fonction de déformation est définie sur l'intervalle $[0, w]$ et représentée sur la figure 4.10 : sa forme particulière permet de conserver les continuités C^1 au niveau des surfaces déformées (voir la figure 4.9). Cette fonction dépend de paramètres physiques des corps modélisés tels que son élasticité et qui permettent aussi de déterminer une force de réponse à la collision.

Ce dernier point ne sera pas abordé dans notre travail : nous faisons exclusivement de l'habillage (voir paragraphe 2.2.3) et notre but n'est pas d'effectuer des calculs de la mécanique ni de détecter les collisions. Ceci a été fait au sein du moteur prenant en charge la mécanique. De plus, n'ayant aucune connaissance *a priori* du modèle mécanique sous-jacent, il se peut que notre modèle implicite ne soit pas du tout adapté aux calculs à y appliquer.

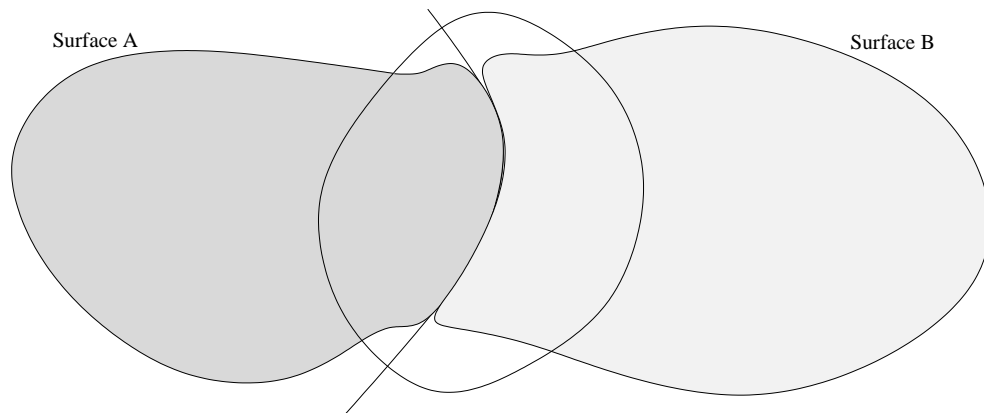
Remarque : *En poussant ce raisonnement un peu plus, on peut également conclure que le modèle de déformations que nous allons proposer n'est pas non plus adapté à la mécanique sous-jacente. Il faudrait normalement prendre en compte l'élasticité des corps plutôt que de déformer empiriquement les surfaces. Néanmoins, nous avons tenu à fournir cette fonctionnalité de l'ordre des «effets spéciaux» (voir le paragraphe 2.3.2.1) afin de construire simplement des déformations complexes sans pour autant complexifier la modélisation mécanique.*

L'idée initiale de [Gas93] consiste donc à calculer la distance entre le point (dont on veut le potentiel) et la zone d'interpénétration des surfaces. Dans le cas général, ce calcul est malheureusement compliqué à réaliser. Dans [OC97], les auteurs proposent d'utiliser le potentiel de la primitive qui engendre la surface avec laquelle on entre en collision (par exemple la surface B sur la figure 4.9) pour construire ce terme correctif¹⁹ qui sera ajouté au potentiel de la primitive

¹⁹. le potentiel étant une fonction décroissante de la distance, il n'est donc pas surprenant que les courbes 4.10 et 4.11 semblent symétriques



(a) Les surfaces initiales



(b) Les surfaces après déformation

FIG. 4.9 – *Contact et déformations de deux surfaces implicites*

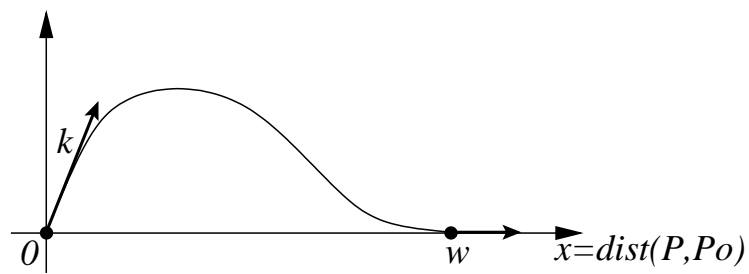


FIG. 4.10 – *Fonction de propagation des déformations [Gas93]*

dont on déforme la surface (par exemple, la surface A sur la figure 4.9). Ce traitement étant symétrique on effectuera également une déformation de la surface B au moyen d'un terme correctif apporté par la surface A. La fonction de déformation est positive, les surfaces affectées par la déformations sont donc *dilatées* en certains points (autour de la zone d'interpénétration).

Dans la zone de déformation, l'expression du potentiel pour la surface A devient donc :

$$F_{A_2}(p) = F_A(p) + \text{deform}(F_B(p)) \quad (4.3)$$

où la fonction *deform*, définie sur $[c_1, c]$ vérifie les propriétés suivantes :

- $\text{deform}(c) = 0$ continuité C^0 de la surface
- $\text{deform}'(c) = -1$ déformation au niveau du contact
- $\text{deform}(c_1) = 0$ continuité C^0 de la surface
- $\text{deform}'(c_1) = 0$ continuité C^1 de la surface au bord de la zone déformée

Par ailleurs, deux autres paramètres interviennent dans la définition de cette fonction : le point m pour lequel la fonction atteint son maximum et la valeur h de ce maximum (voir figure 4.11).

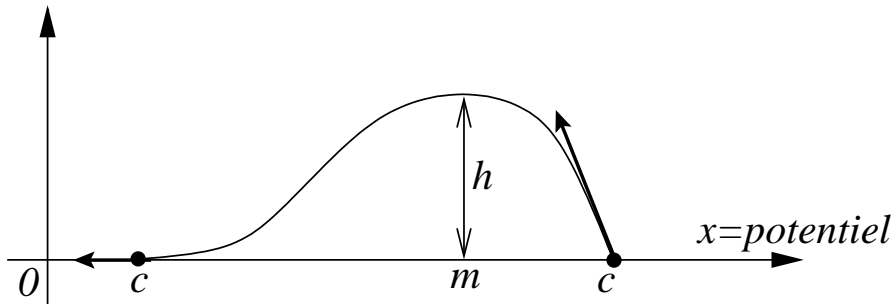


FIG. 4.11 – Fonction de propagation des déformations [OC97]

4.3.2 Notre solution pour les Marching Cubes

La méthode employée ici s'appuie sur les travaux de Marie-Paule Cani et Agata Opalach [Gas93, OC97]. Nous avons adapté les Marching Cubes pour qu'ils puissent bénéficier des apports de ces travaux.

Les déformations autour des zones de contact ne se produisent que lorsque des surfaces entrent en contact et s'écrasent l'une contre l'autre. Il faut donc déterminer quelles sont les surfaces qui s'interpénètrent afin d'ajouter ce terme correctif uniquement lorsque c'est nécessaire. En effet, cette modification des potentiels ne doit pas avoir lieu si les surfaces ne s'interpénètrent pas.

Dans le cas simple de deux primitives, détecter l'interpénétration des deux surfaces peut se résumer à la comparaison entre la somme des deux rayons des surfaces²⁰ et la distance entre les primitives. Cependant ce test n'est plus valide dès que d'autres primitives sont ajoutées : les surfaces ont tendance à grossir légèrement lorsqu'elles fusionnent comme le montre la figure 5.13 en page 124. Le test d'interpénétration va donc être plus complexe dès lors que l'on manipulera plus de deux primitives.

Les points situés à l'intérieur de l'objet implicite ont leur potentiel supérieur au *seuil* servant à déterminer la surface. Ainsi si des points échantillonnés sur la surface B ont un potentiel plus grand que le seuil, alors cela signifie qu'une partie de la surface B se trouve à l'intérieur de la surface A. Cette méthode de détection d'interpénétration est classique et est utilisée par les deux articles déjà cités dans cette partie.

La solution que nous proposons va s'appuyer sur notre méthode permettant d'éviter le *blending* (paragraphe 4.2.3). Nous effectuons une première passe «à blanc» de l'algorithme des *Marching Cubes*. Lors de cette première étape, aucun triangle n'est généré. Cette première passe sert à déterminer quelles portions de surfaces sont en collision en repérant dans chacun des cubes traités quelles sont les surfaces potentiellement en collision. Une liste des interpénétrations est construite pour être utilisée lors de la facettisation proprement dite : elle nous permet de savoir quelles surfaces doivent subir la déformation due aux collisions.

Cet ensemble de couples de surfaces en interpénétration est donc utilisé dans la seconde passe (durant laquelle les triangles sont effectivement construits) lors des évaluations de potentiel : il nous indique quand il est nécessaire d'ajouter aux potentiels les termes correctifs permettant de créer les renflements des surfaces.

4.3.3 Limites de la solution

La figure 4.12 présente une configuration difficile à gérer et pour laquelle notre méthode fournira un résultat légèrement erroné. En effet, une interpénétration va être détectée au niveau de la zone hachurée et les surfaces A et B seront donc considérées en interpénétration. Dès qu'un point est sous l'influence de la surface A et de la surface B, son potentiel sera donc évalué par notre méthode de déformations. Les potentiels vont donc être altérés autour de cette zone hachurée. La déformation dans le bas de la figure se produit bien comme prévu. Malheureusement, lors du traitement des autres extrémités de ces surfaces, le terme correctif sera ajouté aux potentiels. Les portions de surfaces étant sous l'influence l'une de l'autre vont donc être légèrement dilatées

20. qu'il ne faut pas confondre avec les *rayons d'influence* des primitives

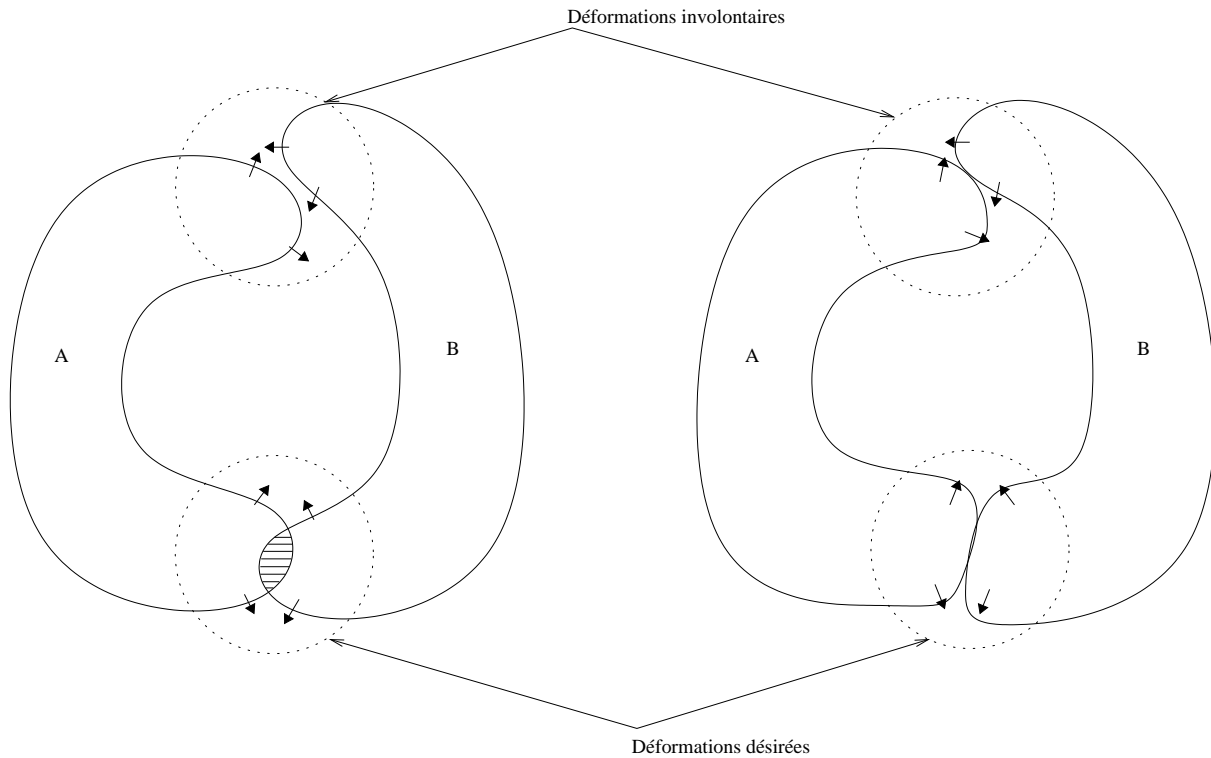


FIG. 4.12 – *Problème de contrôle du blending : une interpénétration est détectée entre A et B, les déformations désirées apparaissent dans le cercle du bas (les surfaces sont dilatées lorsqu'elles sont proches l'une de l'autre), cependant la méthode de calcul crée également des déformations dans le cercle du haut à cause de la proximité des surfaces*

et des déformations auront lieu alors qu'il n'y a pas de contact entre les surfaces à ce niveau.

Résoudre ce problème nécessiterait, comme pour les auto-collisions, de scinder les surfaces en portions disjointes, ce qui nous ramène à la solution proposée en figure 4.6 et aux problèmes énoncés au paragraphe 4.2.2.1 dans le cas de squelettes complexes.

Une autre possibilité consisterait à s'appuyer sur les résultats de la première phase «à blanc» de nos Marching Cubes. On y détecte quels sont les cubes au sein desquels des interpénétrations ont lieu. Sachant que les déformations se propagent autour de ces zones de collisions, il pourrait être intéressant de marquer les cubes autour de ces zones. Ce marqueur indiquerait alors que les calculs de potentiels doivent être altérés par le terme correctif. Néanmoins, deux problèmes subsistent : d'une part comment déterminer quels cubes sont à marquer, et d'autre part si les zones encerclées de la figure 4.12 se rapprochent il y aura de nouveau des déformations dans le cercle du haut.

Conclusion

Nous avons réussi à résoudre partiellement mais de manière simple les problèmes de contrôle de blending. Notre solution s'applique aussi bien au cas des objets disjoints qu'à celui plus complexe des auto-collisions. Les coûts en calculs sont tout à fait tolérables étant donné ce que l'on arrive à faire avec des «simples» Marching Cubes : l'algorithme initial n'est absolument pas prévu pour traiter ce genre de problème.

Concernant la propagation des déformations, là non plus l'algorithme initial ne semblait pas pouvoir être utilisable. Nous avons malgré tout réussi à obtenir des résultats valables dans de nombreuses configurations.

Dans le chapitre qui suit, nous présentons différents résultats montrant les performances de notre implantation. Nous voyons également que sa mise sous forme de librairie, simple à utiliser, lui vaut d'être utilisée dans d'autres laboratoires de recherche.

Chapitre 5

Résultats expérimentaux

Sommaire

5.1	Présentation et analyse des performances	112
5.1.1	Incidence du nombre de primitives	112
5.1.2	Incidence du nombre de cubes traités	114
5.2	Comparaisons avec d'autres implantations	117
5.2.1	L'implantation de Bloomenthal [Blo94]	118
5.2.2	VTK et GTS	119
5.3	Performances avec contrôle du blending	121
5.4	Quelques captures d'écrans	123
5.5	Utilisation dans SPIC	127
5.6	Librairie et collaborations	130

Introduction

Dans ce chapitre nous présentons des résultats propres à notre travail : courbes de performances et captures d'écrans. Nous comparons ensuite d'autres implantations à la nôtre. Enfin, nous présentons différentes collaborations mises en place au cours de ma thèse.

Remarque : *Toutes les mesures de temps ont été effectuées sur un ATHLON à 800 MHz. À l'heure actuelle, c'est une machine de gamme moyenne pour le grand public.*

5.1 Présentation et analyse des performances

5.1.1 Incidence du nombre de primitives

Comme le montre la courbe de performances en figure 5.1, notre implantation permet de facettiser des surfaces implicites à squelette complexes en un temps très satisfaisant. Par exemple, nous parvenons à traiter un ensemble de 200 primitives en moins de 25 ms. La figure 5.2 montre un exemple de surface obtenue grâce à un squelette de 200 primitives. L'animation qui a servi à faire ces mesures de temps consiste à déplacer les primitives dans des directions aléatoires en les faisant rebondir sur les parois de la grille des Marching Cubes. De manière générale, on constate que les temps de calcul croissent linéairement en fonction du nombre de primitives. Ces résultats ont été obtenus pour la précision de facettisation (c'est-à-dire la résolution de la grille par rapport à la taille des primitives) utilisée dans notre simulateur.

La résolution de la grille y est réduite pour minimiser les temps de traitements mais maintenue suffisamment élevée pour que les résultats visuels soient satisfaisants et donc que les corps représentés ne montrent pas de contours anguleux.

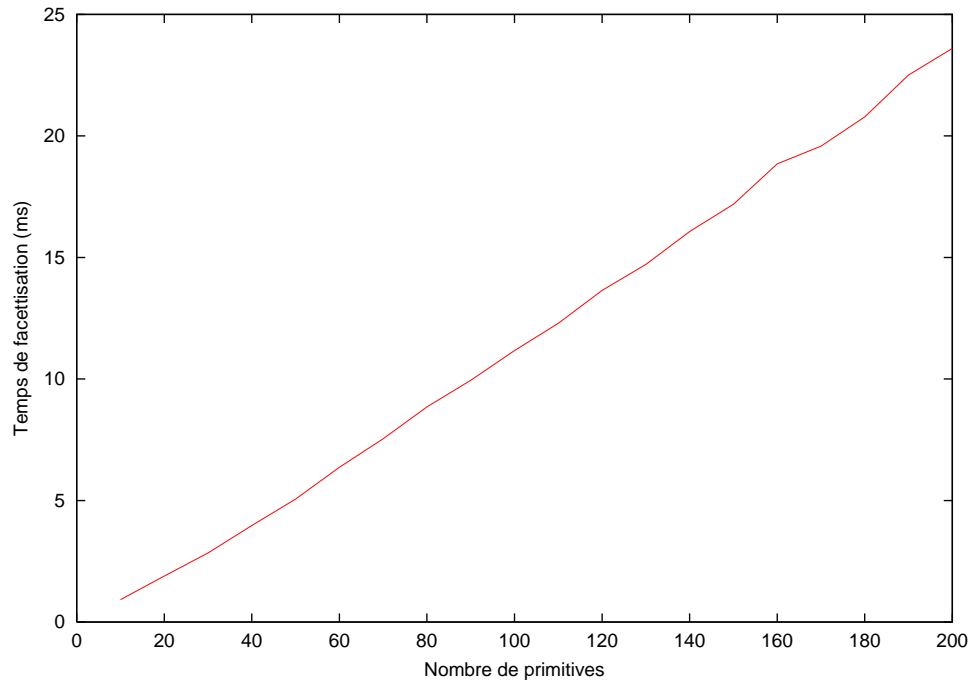


FIG. 5.1 – Temps de facettisation en fonction du nombre de primitives

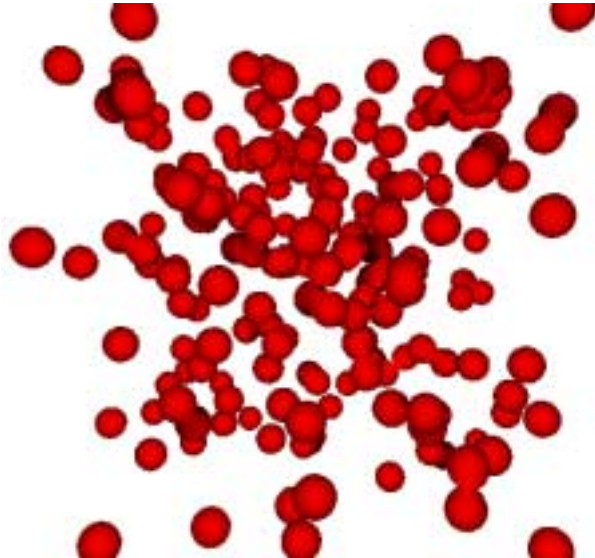


FIG. 5.2 – 200 primitives

5.1.2 Incidence du nombre de cubes traités

Le nombre de primitives n'est effectivement pas le seul paramètre faisant varier la vitesse de facettisation d'une surface. Il faut également prendre en compte la résolution de la grille. Certains tests nous ont montré que le nombre moyen de triangles par cube²¹ est de 1.8 . Cette moyenne est constante quels que soient le type de surface et la précision de facettisation. Un moyen pour prendre en considération l'influence de la résolution des cubes est d'exprimer les temps de calculs en fonction du nombre total de triangles obtenus. Le tableau 5.3 présente l'évolution du nombre de ces triangles en fonction du nombre de primitives à facettiser. Ces valeurs ont été obtenues dans le cadre d'une utilisation «normale» (i.e. comme c'est le cas dans le simulateur). Par ailleurs, le tableau 5.4 présente cette même évolution dans le cas où la précision de la grille est plus élevée (les cubes ont des arêtes 3 fois plus petites, la grille est donc composée de 27 fois plus de cubes). Une même primitive facettisée à ces deux résolutions est présentée en figure 5.5. On constate sur les deux tableaux que la progression du nombre de triangles obtenus est une fonction linéaire du nombre de primitives, de ce fait, les temps de facettisation varient linéairement par rapport au nombre de triangles générés (et donc du nombre de cubes parcourus).

Remarque : *Dans les paragraphes qui suivent, cette résolution de facettisation est parfois appelée «haute résolution» dans le sens où cette qualité de facettisation est bien supérieure au minimum de précision requis pour représenter les corps de manière suffisamment détaillée dans le simulateur.*

Nombre de primitives	10	20	30	40	50	60	70	80	90	100
Nombre de triangles obtenus	554	1108	1663	2222	2777	3334	3890	4450	5006	5561
Nombre de primitives	110	120	130	140	150	160	170	180	190	200
Nombre de triangles obtenus	6118	6677	7233	7794	8346	8908	9461	10019	10589	11139

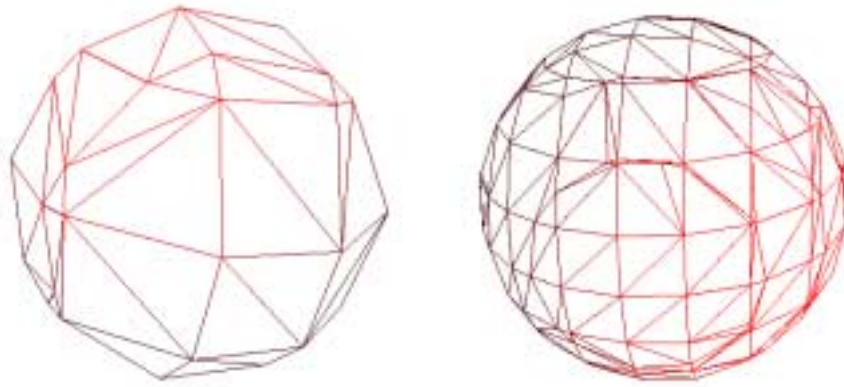
FIG. 5.3 – *Nombre de triangles en fonction du nombre de primitives, facettisation de qualité normale*

Le nombre de triangles construits étant nettement supérieur dans le second tableau, les temps de calculs sont donc naturellement plus importants lorsque l'on augmente la résolution de la grille (voir figure 5.6). Ces résultats ont été obtenus en facettisant exactement les mêmes surfaces, seule la précision de facettisation change.

²¹. en ne prenant en compte que les cubes coupés par la surface, que nous avons appelés «cubes utiles»

Nombre de primitives	10	20	30	40	50	60	70	80	90	100
Nombre de triangles obtenus	4776	9556	14317	19088	23869	28609	33375	38104	42827	47511
Nombre de primitives	110	120	130	140	150	160	170	180	190	200
Nombre de triangles obtenus	52127	56808	61401	66070	70678	75343	80003	84648	89328	93828

FIG. 5.4 – Nombre de triangles en fonction du nombre de primitives, facettisation de qualité supérieure



(a) Résolution moyenne

(b) «Haut» résolution

FIG. 5.5 – Facettisations aux deux résolutions

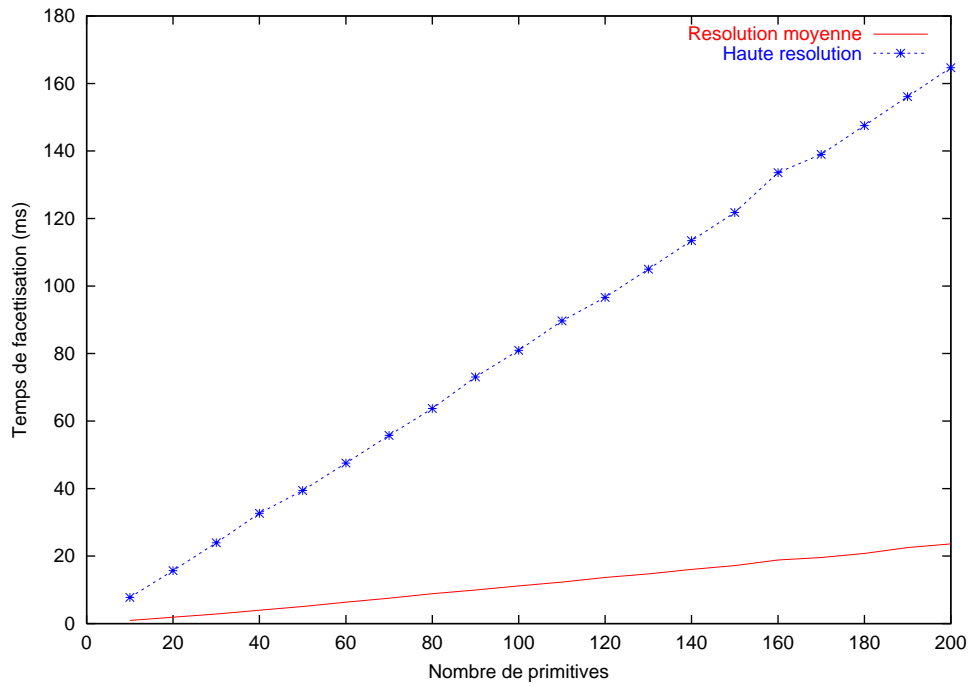


FIG. 5.6 – Temps de facettisation à des résolutions de grilles différentes

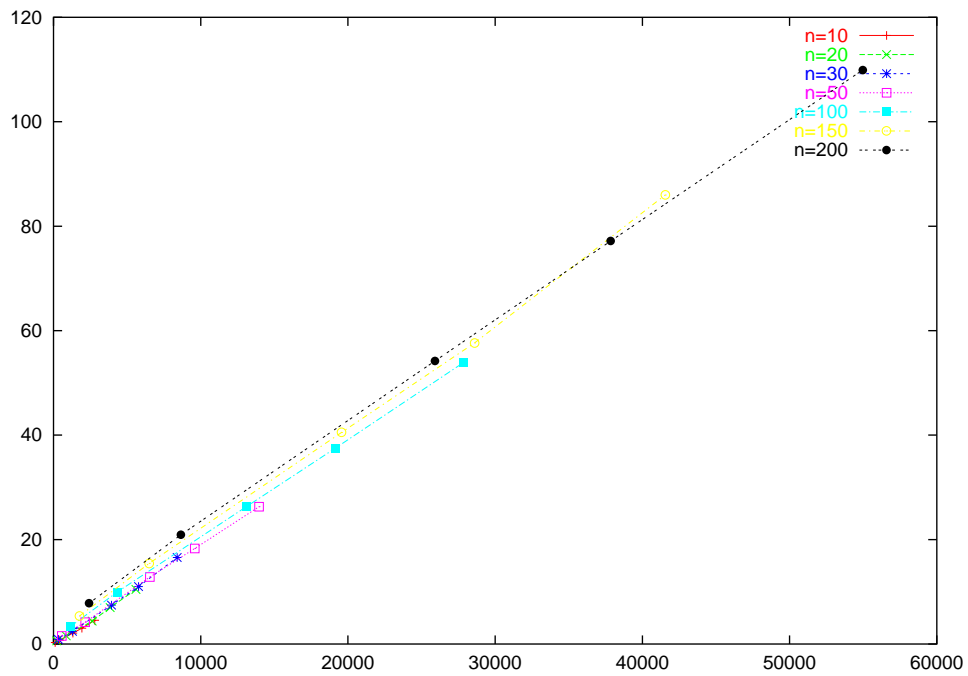


FIG. 5.7 – Temps de facettisation en fonction du nombre de triangles obtenus, le nombre de primitives restant constant

La figure 5.7 présente l'évolution des temps de facettisation en fonction du nombre de triangles construits. Chacune des courbes traduit ces variations pour un nombre constant de primitives. On constate que ces courbes sont à peu près confondues. La conclusion que l'on peut en tirer est qu'avec notre méthode, le facteur prépondérant dans les temps de facettisation est le nombre de triangles construits. Comme les courbes passent par l'origine du repère, on peut alors conclure qu'un changement de résolution de grille pour doubler le nombre de triangles d'une surface aura pour effet de doubler le temps de facettisation.

5.2 Comparaisons avec d'autres implantations

Malgré le succès des surfaces implicites à squelette dans le domaine de l'animation, il est difficile de trouver des implantations d'algorithmes dont le but serait une visualisation temps-réel de surfaces complexes.

Afin de comparer les performances de diverses implantations, il est nécessaire d'une part de les tester rigoureusement dans les mêmes conditions d'utilisation et d'autre part d'effectuer des mesures de temps précises. Pour ces deux raisons, il est nécessaire de travailler directement sur le code source de l'application. C'est pourquoi nous ne présentons ici que des comparaisons avec des implantations dites *Open-Source*²² et aucune autre implantation de type industriel.

Il existe de nombreuses implantations des Marching Cubes. Pour la plupart, leurs auteurs n'ont pas cherché à optimiser et les performances ne méritent pas d'être présentées en détail ici.

Toutefois, trois implémentations sortent quelque peu du lot. Il s'agit de l'implantation de Jules Blomenthal dans les Graphic Gems [Blo94] et des deux bibliothèques *GTS* et *VTK*. À l'heure actuelle, le facettiseur de surfaces implicites d'Alexander Pasko²³ n'est pas encore distribué en open-source, cependant une comparaison me semble intéressante et sera réalisée dès que les sources seront disponibles.

Afin de comparer les performances des différentes implantations, nous avons facettisé des surfaces aux squelettes identiques dans des grilles également identiques.

22. c'est-à-dire dont le code source est disponible gratuitement

23. <http://www.hyperfun.org>

5.2.1 L'implantation de Bloomenthal [Blo94]

Le code présenté par Jules Bloomenthal propose une implantation des Marching Cubes relativement simple et très didactique. Cette implantation est très connue dans la communauté graphique et peut être considérée comme une référence dans le domaine de la facettisation de surfaces implicites.

Néanmoins, elle souffre de sa simplicité car pour rester facilement compréhensible, le code n'a pas été particulièrement optimisé. On remarquera en particulier l'utilisation massive d'allocations dynamiques de mémoire (`malloc/free`) qui, à chaque appel, utilisent des ressources du système d'exploitation. On y retrouve cependant l'optimisation du «suivi de surface» qui permet de ne traiter que les cubes coupés par la surface (voir le paragraphe 3.3.1 en page 71). Il reste à noter que cette implantation ne parvient pas à traiter des surfaces disjointes : c'est l'utilisateur qui doit faire une recherche de connexité afin de fournir des points de départ sur les différentes portions de surfaces.

La courbe 5.8 montre nettement les différences de performances entre l'implantation de Bloomenthal et la nôtre. Un ratio d'environ 40 peut être mesuré sur les temps de calculs !

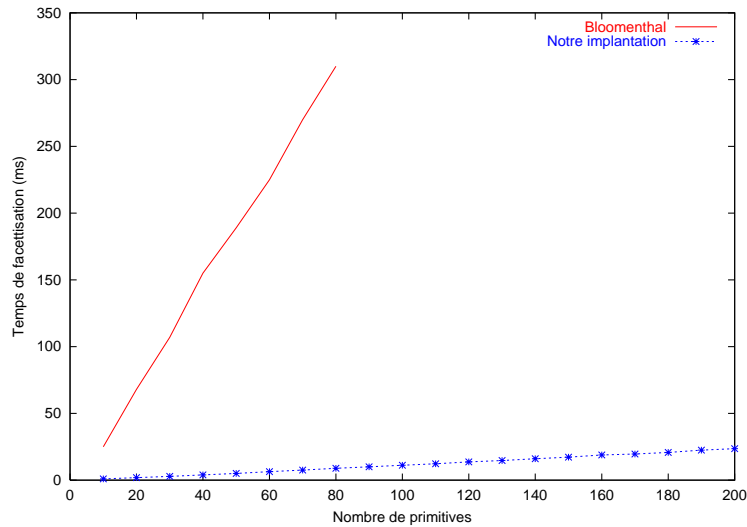


FIG. 5.8 – Comparaison des performances avec l'implantation de J. Bloomenthal

Il est possible de réduire ce ratio en utilisant notre évaluation de potentiel (expression de la fonction *et* sommation basée sur les voxels, voir paragraphe 3.2 en page 67). Les résultats sont présentés en figure 5.9 et font état d'un gain en performances très important. L'optimisation de l'évaluation du potentiel basée sur les voxels que nous utilisons est donc tout à fait

pertinente. Toutefois notre implantation demeure nettement plus rapide (on constate encore des performances 5 fois meilleures pour notre implantation). Toutes les autres optimisations n'ayant pas trait aux évaluations de potentiel que nous proposons ont donc un impact bien réel sur les performances globales du système.

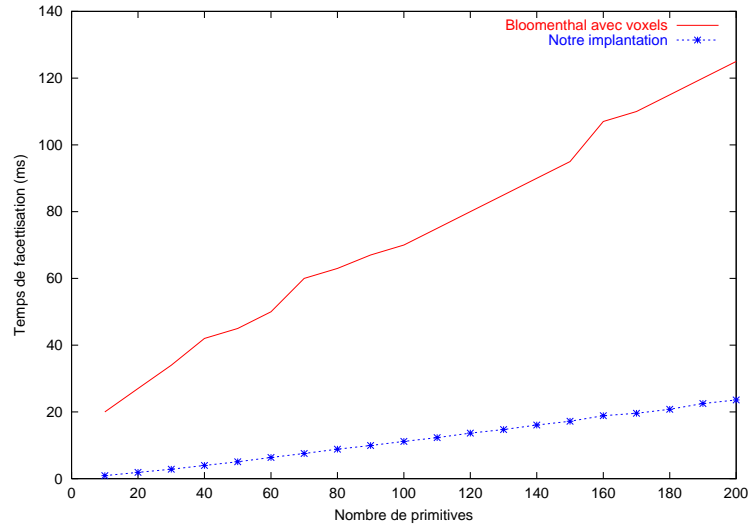


FIG. 5.9 – Amélioration de l'implantation de Bloomenthal grâce à notre évaluation de potentiel

5.2.2 VTK et GTS

Les bibliothèques *Visual ToolKit* [VTK] et *GNU Triangulated Surface*[GTS] sont deux bibliothèques de manipulation de surfaces. Leurs fonctionnalités sont variées. Elles permettent par exemple l'application d'opérateurs de CSG²⁴ sur des maillages de triangles, la réduction du niveau de détail de surfaces, etc.

Hormis ces fonctionnalités, elles proposent également la possibilité de facettiser un champ scalaire, ce qui nous a amenés à les comparer à notre implémentation. Ces deux bibliothèques sont à ma connaissance les seuls projets dits *Open-Source* proposant de facettiser des champs scalaires (et qui peuvent donc être utilisées pour facettiser nos surfaces implicites à squelette).

Toutefois, comparer leurs performances à celle de notre implantation n'est pas véritablement très pertinent car aucune de ces bibliothèques n'utilise de suivi de surface. L'intégralité de la grille est donc traitée et les coûts en temps de calculs en sont gravement affectés. Par ailleurs, les structures de données employées mettent en œuvre des mécanismes parfois compliqués et coûteux au vu de

²⁴. Constructive Solid Geometry : opération ensemblistes appliquées aux volumes

nos besoins en temps de traitement : en particulier ces bibliothèques maintiennent des informations topologiques sur les maillages.

Ces deux bibliothèques nécessitent que l'utilisateur remplisse préalablement la grille des potentiels.

Les résultats présentés sur la figure 5.10 ont été obtenus pour des grilles de taille minimale. Le nombre de cubes inutilement traités est ainsi réduit. Par ailleurs, l'évaluation des potentiels (c'est-à-dire le remplissage de la grille de potentiels) étant laissée à la charge de l'utilisateur, les temps de calculs de ces potentiels n'apparaissent pas dans les résultats présentés ici. Les courbes ne présentent donc que les temps de calculs liés à la facetisation.

En conséquence, la complexité du squelette influe peu sur les temps de facetisation puisque les temps de remplissage de la grille de potentiels ne sont pas comptabilisés. Par ailleurs, même pour un très faible nombre de primitives (0 étant ici le cas extrême), les temps de traitements sont relativement élevés (au moins 80 ms!). Ceci est dû au traitement intégral de la grille de cubes qui est effectué quel que soit le contenu de la grille.

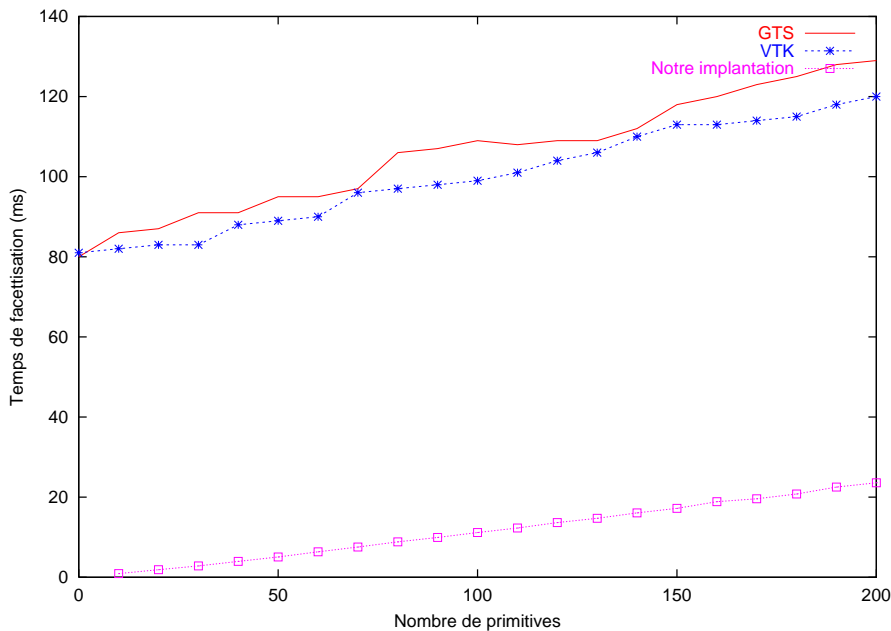


FIG. 5.10 – Performances des bibliothèques GTS et VTK comparées aux nôtres

Bilan

Comparer ces différentes implantations [Blo94, GTS, VTK] à la nôtre peut paraître relativement peu intéressant car elles ne sont pas destinées au temps réel.

Cependant l'habitude de considérer que l'algorithme des Marching Cubes est trop lent pour être utilisable en temps réel sur des surfaces complexes vient essentiellement du fait que l'on se base sur ces implantations et principalement sur celle de Bloomenthal. Son but était de proposer un code simple et effectivement il n'atteint pas de performances intéressantes.

Nous venons de montrer que notre implantation est nettement plus efficace et que le rendu temps-réel de corps construits au moyen de surfaces implicites à squelette relativement complexes est réalisable en temps réel.

5.3 Performances avec contrôle du blinding

Le contrôle du blinding nécessite des traitements supplémentaires qui affectent les performances (voir la figure 5.11). Les temps de calculs peuvent être jusqu'à 8 fois plus importants que lorsque l'on ne contrôle pas le blinding ! Cette perte de performances a trois origines :

- notre méthode requiert que certains cubes soient traités plusieurs fois ;
- le mécanisme d'invalidation des dates (nécessaire au traitement multiple des cubes) occasionne de nombreux recalculs qui sont habituellement évités ;
- il est nécessaire de construire la liste des primitives nécessaires au traitement de chaque cube (voir en page 100).

Ces trois points particuliers à notre méthode influent plus ou moins sur les performances selon le squelette à traiter. En effet, plus les zones où le blinding doit être contrôlé sont nombreuses et/ou étendues, et plus le nombre de cubes traités plusieurs fois sera important. Les recalculs dus aux invalidations de dates seront également plus nombreux.

La figure 5.12 présente la surface utilisée pour nos mesures de temps. L'affichage obtenu fait penser à un ressort que l'on comprimerait afin que ses différents «anneaux» entrent en contact. Les résultats décrits ici montrent une situation «extrême» dans laquelle nos traitements spécifiques sont énormément sollicités. Les zones de contacts sont très étendues et presque tous les cubes considérés lors de la facettisation doivent être traités plusieurs fois.

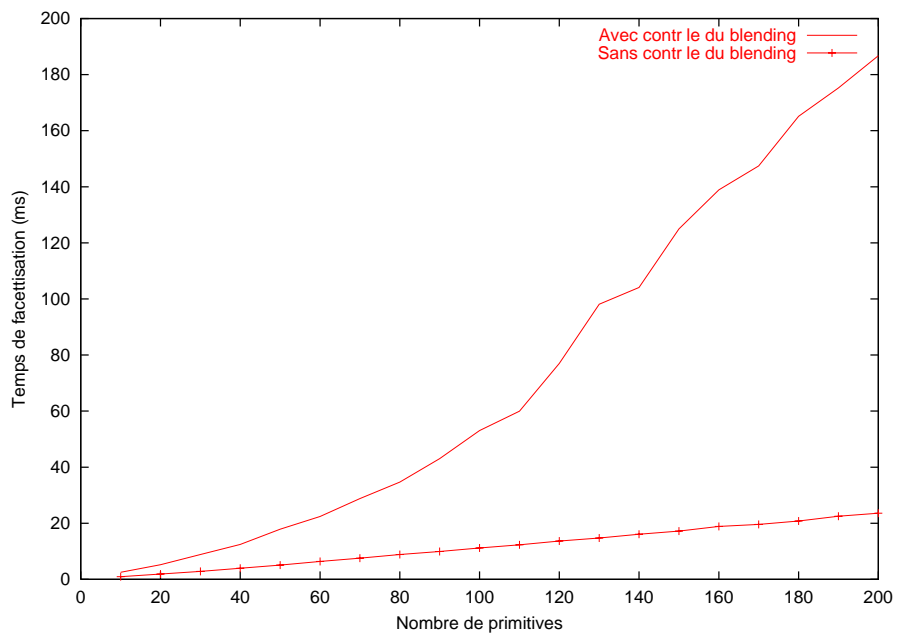


FIG. 5.11 – Temps de facettisation d'un objet avec et sans contrôle du blending

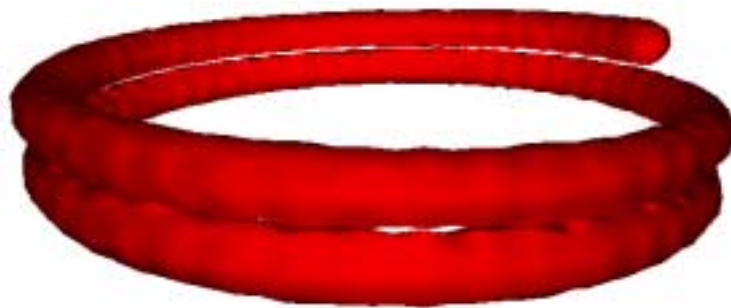


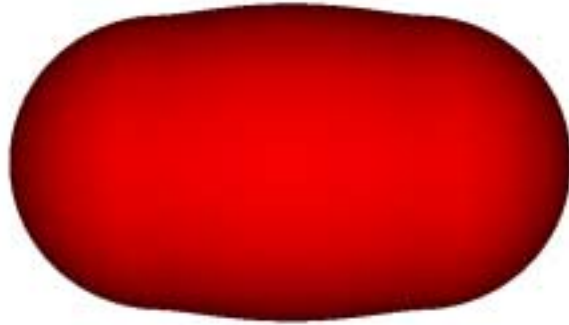
FIG. 5.12 – Surface avec contrôle du blending

5.4 Quelques captures d'écrans

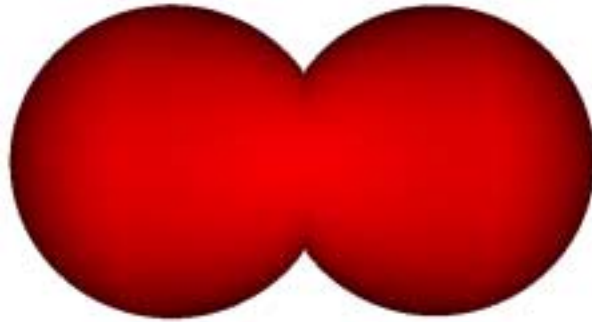
La figure 5.13 présente les 3 modes de blending appliqués à deux primitives : le blending peut être autorisé (figure 5.13a), simplement inhibé (figure 5.13b) ou inhibé avec en plus la propagation des déformations liées aux compressions des surfaces.

La figure 5.14 présente sensiblement les mêmes surfaces représentées cette fois en vue filaire, ce qui permet de bien se rendre compte des facettisations obtenues, en particulier au niveau des zones de contact.

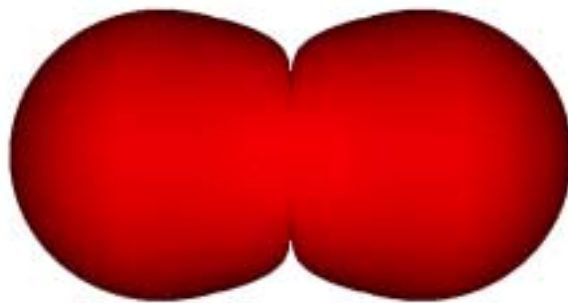
FIG. 5.13 – *Deux primitives*



(a) Avec blending

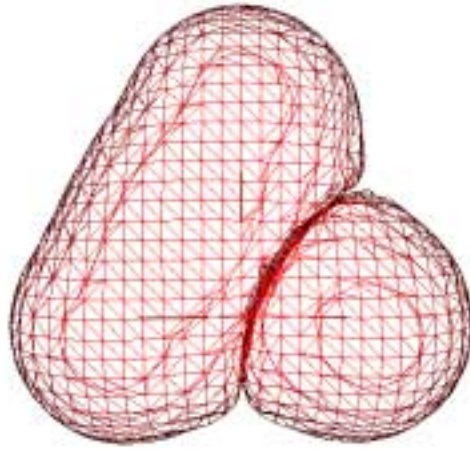


(b) Avec contrôle du blending

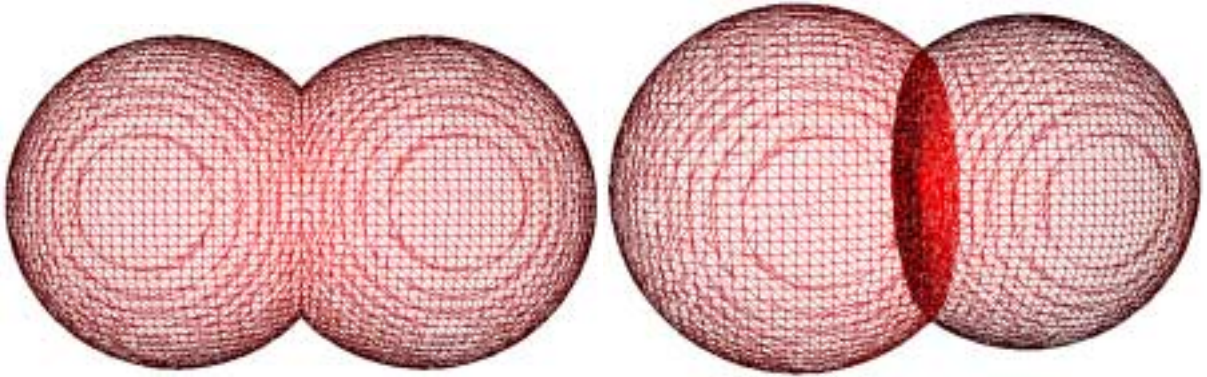


(c) Avec déformation des surfaces comprimées

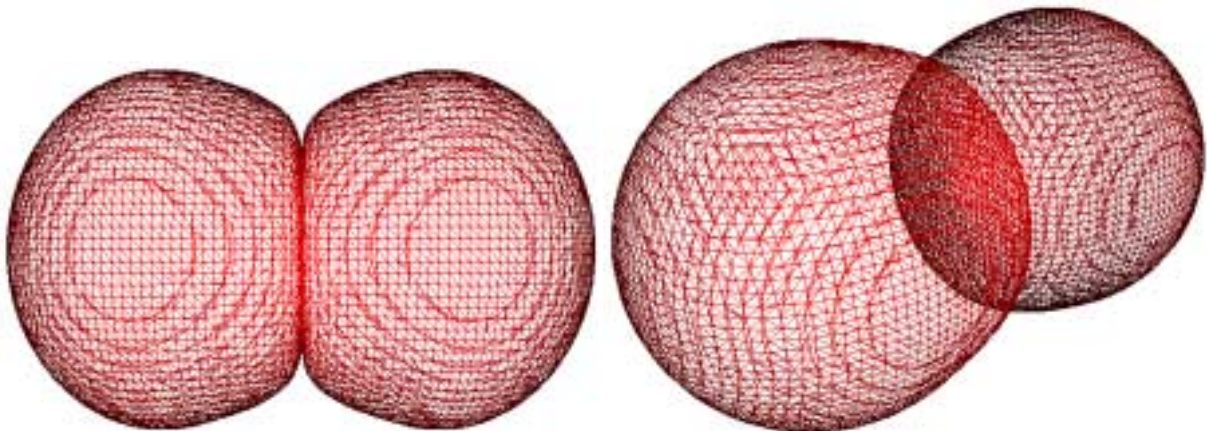
FIG. 5.14 – Quelques vues en fil de fer



(a) Trois primitives



(b) Voir la figure 5.13b (vue de face et légèrement de côté)



(c) Voir la figure 5.13c (vue de face et légèrement de côté)

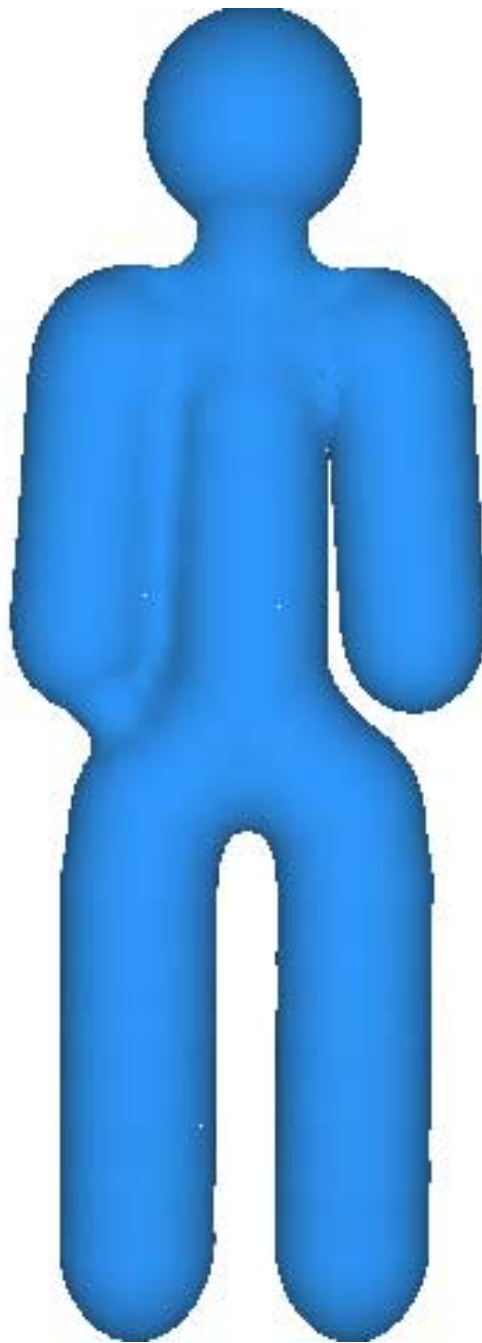


FIG. 5.15 – *Contrôle du blending entre le bras de droite et le corps mais pas au niveau du bras de gauche*

5.5 Utilisation dans SPIC

Notre bibliothèque a enrichi *SPORE* des fonctionnalités permettant la visualisation temps réel de surfaces implicites à squelette. Ces surfaces sont utilisées pour représenter d'une part des corps n'ayant pas de structure propre comme les écoulements sanguins et d'autre part des organes très déformables mais dont la structure (la topologie) ne varie pas dans le temps comme c'est le cas des intestins.

La figure 5.16 présente un écoulement sanguin que l'on peut «manipuler» très librement par exemple avec les outils de chirurgie utilisés lors de la simulation d'opération.

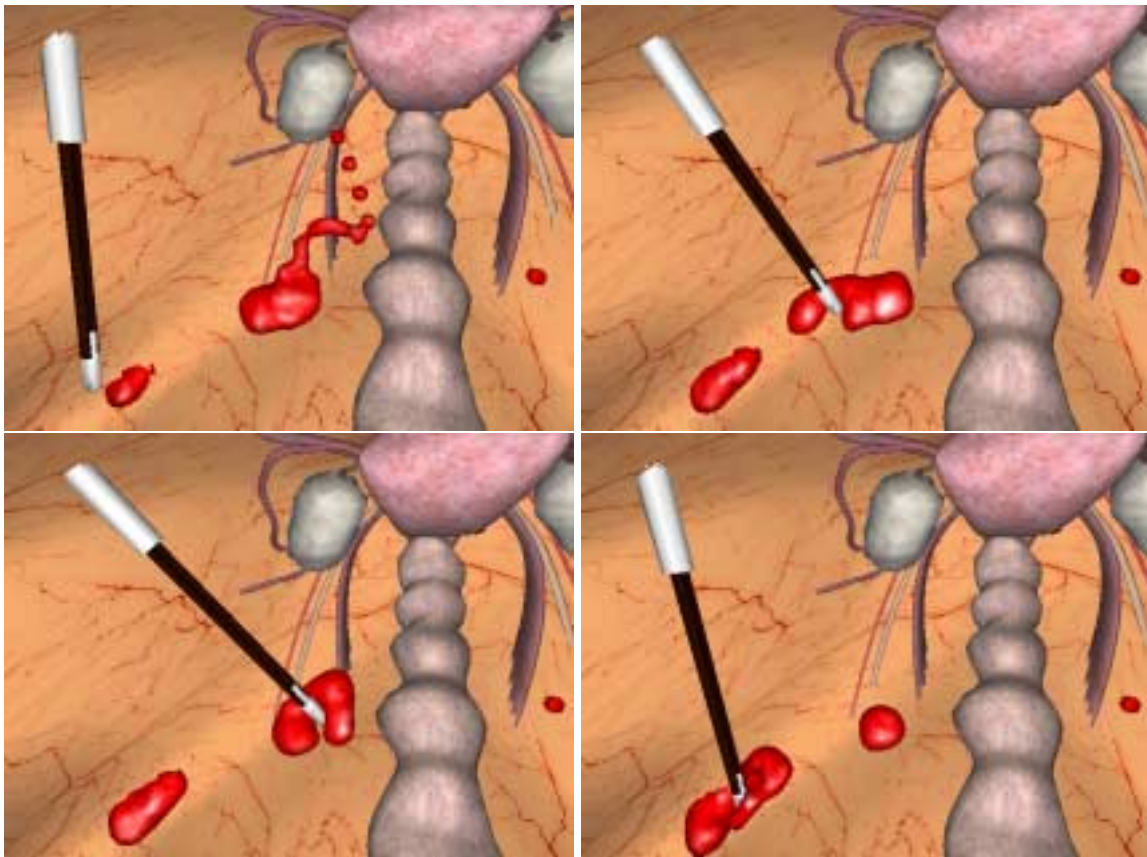


FIG. 5.16 – *Écoulements sanguins*

La représentation d'intestins sur la figure 5.17, utilise un contrôle du blending afin d'éviter la fusion aux endroits où il se replie sur lui-même. Le cas de l'intestin pose actuellement un problème imprévu. L'utilisation de notre outil de facettisation est actuellement restreinte à une

seule grille et donc une seule résolution de facettisation. Cette résolution doit être suffisamment fine pour que les écoulements sanguins ne soient pas anguleux. C'est donc cette résolution qui est utilisée pour facettiser l'intestin et il s'est avéré que le nombre de triangles construits pour le représenter (entre 30 000 et 50 000) dépasse les capacités des accélérateurs graphiques. Il sera donc utile de réduire le nombre de triangles de certains corps en prenant en considération d'une part la mise en place d'une facettisation adaptative et d'autre part des améliorations dans notre gestion de la mémoire afin de pouvoir utiliser en même temps des grilles de Marching Cubes à des résolutions différentes.

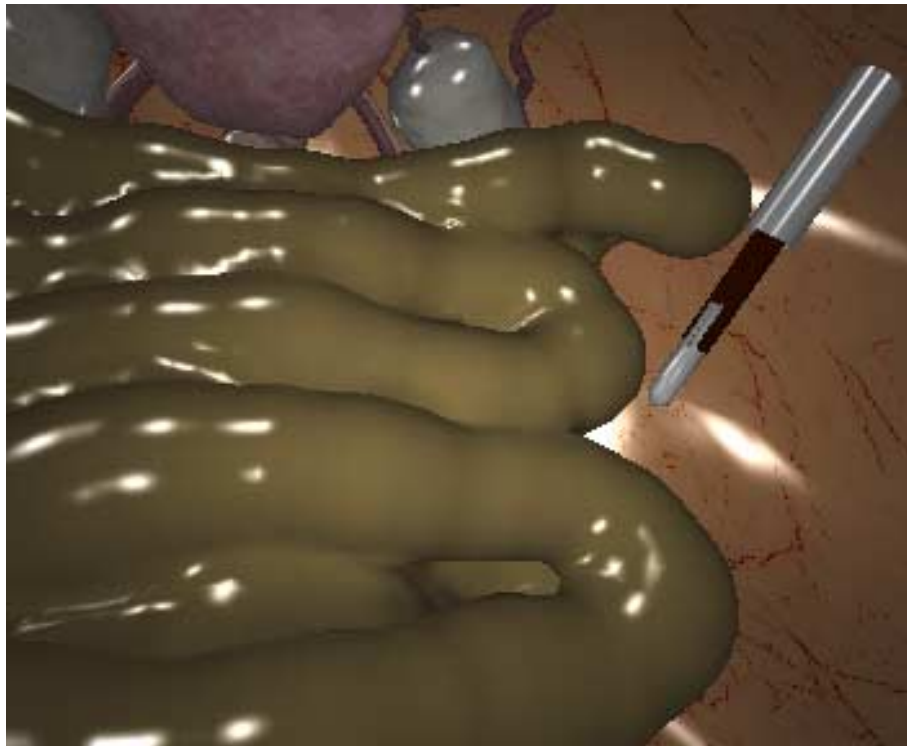


FIG. 5.17 – *Intestin avec contrôle du blending*



5.6 Librairie et collaborations

L'implantation réalisée a été mise sous forme de librairie afin de satisfaire le cahier des charges de la plateforme *ALSPeMe*. La librairie que nous avons conçue nous permet actuellement d'afficher des écoulements sanguins et un intestin dans le simulateur *SPIC*.

La simplicité d'utilisation de cette librairie lui a valu un certain succès auprès d'autres laboratoires de recherche français et étrangers.

Au LERI²⁵ de Reims, notre librairie est utilisée comme outil de prévisualisation d'objets complexes pouvant comporter plusieurs milliers de primitives. Les objets sont affichés très rapidement (une à deux secondes) grâce à notre librairie pour ensuite être traités dans un moteur de lancer de rayon qui fournit des images de bien meilleure qualité au prix de calculs très longs (plusieurs minutes).

Notre librairie était initialement prévue pour être utilisée dans un contexte radicalement différent (quelques centaines de primitives et des résultats en temps réel). Cependant le comportement très satisfaisant de notre librairie dans les expérimentation du LERI ont confirmé que son utilisation ne se limite pas qu'aux animations temps réel de surfaces de complexité moyenne.

5092 primitives 35152 facettes 4 secondes	4918 primitives 30498 facettes 6 secondes
	

On remarque certains défauts dans les surfaces obtenues : des facettes manquent dans les

25. Laboratoire d'Étude et de Recherche en Informatique

maillages composants les objets. Ceci est dû à l'utilisation d'une ancienne version de notre librairie pour réaliser ces images. Cette version ne traitait pas les problèmes de facettisations incorrectes. Ces problèmes sont à présent résolus.

Le GMI²⁶ de l'université de Valence (Espagne) travaille sur la reconstruction à partir d'informations de type $2.5D$. Le but est de reconstruire les surfaces de différents tissus à partir de plans de coupe de type scanner ou issus du projet Visible Human par exemple. Ma librairie ne permettait pas initialement de traiter ce type de problème et il a fallu en adapter certaines parties. Le logiciel de recherche dans lequel ma librairie a été intégrée est visible en annexe B et un résultat de reconstruction est présenté en figure 5.18.

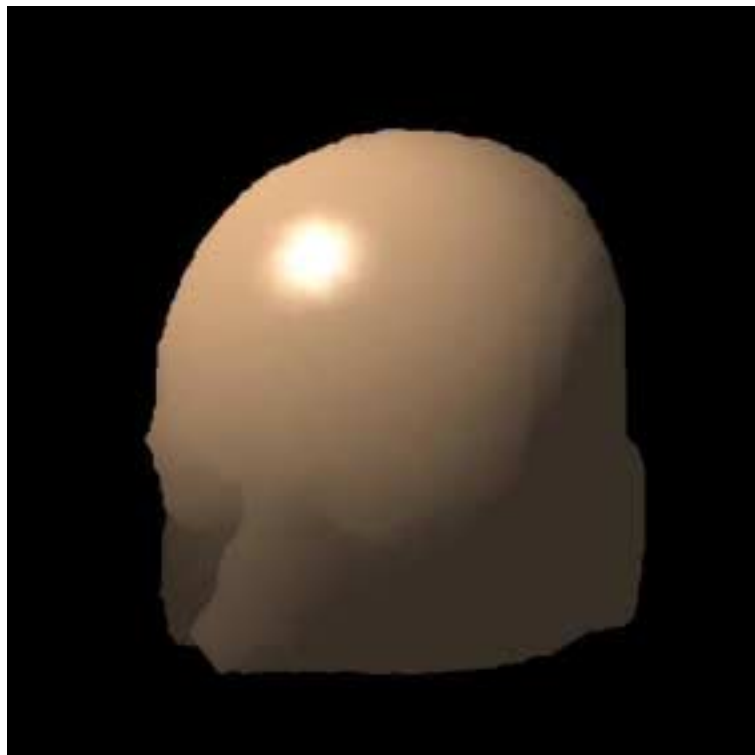


FIG. 5.18 – *Reconstruction d'un crane*

Notre librairie est également utilisée à l'ACROE de Grenoble [GL01] et au CRAS de Guadalajara.

26. Group of Medical Informatics

Conclusion

Notre bibliothèque atteint des performances suffisantes pour être utilisable au sein du moteur *SPORE* et par exemple dans le simulateur d'intervention cœlioscopique nommé *SPIC*.

L'habillage que l'on fournit est détaillé, et ce malgré la simplicité des modèles mécaniques sous-jacents utilisés.

La baisse de performances causée lorsque l'on contrôle le blending reste un problème important. Néanmoins, les temps de calculs actuels et certaines améliorations possibles nous laissent supposer que des résultats bien meilleurs sont envisageables.

Conclusion et perspectives

Bilan des travaux

Notre but était d'afficher des surfaces implicites à squelette dans le contexte temps réel des simulateurs chirurgicaux. La facettisation de ces surfaces, indispensable pour tirer profit des accélérateurs graphiques actuels est une tâche réputée pour être gourmande en temps de calculs. Bien souvent, l'utilisation des surfaces implicites est reléguée à des applications où les exigences sont moins focalisées sur la rapidité d'exécution. Par ailleurs, l'algorithme des Marching Cubes a également une assez mauvaise réputation en termes de performances.

Néanmoins, grâce à des structures de données évoluées et des optimisations avancées, nous sommes parvenus à tirer profit de la simplicité de l'algorithme initial qui fournit directement les triangles tout en l'adaptant à nos contraintes fortes de temps réel.

Les surfaces obtenues fournissent un habillage à des modèles mécaniques animant des corps au sein de simulateurs chirurgicaux.

Initialement prévue pour la représentation fluides qui sont des corps non structurés et difficilement modélisables avec d'autres outils géométriques que les surfaces implicites à squelette, notre méthode d'habillage a été étendue à la représentation d'autres types de corps présentant une structure et une topologie donnée, principalement les organes de type viscéral comme les intestins. Nous nous sommes intéressés au contrôle du blinding dans le contexte particulier des Marching Cubes où aucune solution n'avait été proposée jusqu'à présent. Notre méthode, bien qu'imparfaite, permet de facettiser des surfaces dont la topologie est définie par l'utilisateur et qui présente l'avantage de ne pas être figée et de pouvoir évoluer facilement au cours du temps.

Le contrôle du blinding impliquant certaines déformations aux surfaces, nous avons été amenés à adapter une méthode de propagation des déformations à notre système de facettisation.

Le travail réalisé nous fournit à présent une librairie dont l'utilisation est des plus simples, et dont les performances nous permettent de représenter aisément des fluides (écoulements sanguins

ou autres) et des corps pour lesquels un contrôle du blending est nécessaire.

Enfin, la bibliothèque mise au point pour être intégrée dans la plateforme *ALSPeMe* est utilisée dans d'autres laboratoires de recherche et dans des cadres bien différents du nôtre. Ces conditions d'utilisation bien différentes de celles pour lesquelles elle était initialement prévue marquent la robustesse et la souplesse des solutions que nous fournissons à différents problèmes.

Mes travaux s'orientent à présent vers les points présentés au chapitre précédent. Améliorer notre technique de contrôle de blending et économiser les ressources en mémoire sont deux points importants à régler pour l'utilisabilité de notre librairie. Viendront ensuite l'utilisation de squelettes continus et des investigations dans la facettisation par graines.

Perspectives

Le sujet reste encore bien ouvert et les perspectives d'évolution sont nombreuses.

1 Améliorations du contrôle du blending

La méthode que nous proposons pour contrôler le blending des surfaces est une avancée importante dans la mesure où l'algorithme des Marching Cubes initial n'est véritablement pas prévu pour traiter ce type de problème et où aucune méthode n'avait été proposée jusqu'alors dans ce cadre particulier. Néanmoins, certains défauts subsistent dont le surcoût non négligeable en temps de traitement et certains artefacts dans la facettisation.

Nous sommes actuellement obligés d'invalider de nombreux résultats de calculs (potentiels, points d'intersection surface/grille, vecteurs normaux obtenus grâce au gradient de la fonction de potentiel) pour pouvoir traiter plusieurs fois un même cube. Cependant, ces données doivent être recalculées par la suite (lorsque l'on traite les cubes voisins). Plutôt que d'invalider les données contenues dans un cube lorsque l'on doit le traiter plusieurs fois, peut-être serait-il intéressant de stocker les différentes configurations sous lesquelles il apparaît dans le traitement.

2 Les squelettes continus

Le modèle mécanique sur lequel notre travail s'est appuyé met en scène des systèmes mécaniques composés exclusivement de points. De ce fait, nous n'utilisons que des squelettes composés de primitives ponctuelles. Une évolution intéressante de notre bibliothèque serait l'incorporation de surfaces basées sur des squelettes continus ([BS91]) et plus particulièrement sur des courbes à subdivision ([CH01]).

Notre bibliothèque peut faire l'objet de légères modifications permettant de traiter également des squelettes continus. Cependant, les méthodes à base de graines telles que celle développée par A. Angelidis [Ang01] me semblent mieux appropriées car elles prennent en compte la topologie des squelettes de manière native et ne sont donc pas sujettes aux problèmes de contrôle de blending.

3 Économies en mémoire

Les ordinateurs utilisés pour la simulation chirurgicale présentent les caractéristiques matérielles pour utiliser sans trop de limitations notre librairie de facettisation dynamique de surfaces implicites. Cependant, un obstacle à une utilisation plus grand «grand public» réside dans la quantité de mémoire parfois importante dont notre librairie peut avoir besoin. Ceci pourra être amélioré en adaptant le traitement pour qu'il ne nécessite pas le stockage en mémoire de l'intégralité de la grille de cubes. La méthode consistant à facettiser un plan de cubes à la fois et présentée au paragraphe 3.5.1.3 devrait être programmable sans trop de problèmes et permettra des gains de mémoire importants.

4 Évolution de la bibliothèque

Il pourra être intéressant d'adapter notre bibliothèque de facettisation à des surfaces qui se déforment peu dans le temps et qui, de ce fait, permettent de construire la facettisation en s'appuyant sur le maillage obtenu à l'étape précédente.

Par ailleurs, il pourrait être utile de déterminer de manière automatique quelles doivent être les propriétés de la grille de voxels (voir paragraphe 3.2.2 en page 69) pour que les performances soient les meilleures possibles en fonction des caractéristiques du squelette à facettiser.

La librairie que nous avons développée pourra être complétée par l'implantation de méthodes de facettisation à base de graines, principalement dans le but de tirer profit de la structure linéaire de certains corps viscéraux comme les intestins.

Un autre point sur lequel notre librairie peut évoluer serait de fournir différents niveaux d'intervention à l'utilisateur. Il serait intéressant de pouvoir fournir d'autres fonctions de potentiel, ou même de facettiser des surfaces implicites de type plus général qui ne seraient pas nécessairement basées sur un squelette. Un dernier point a déjà été entamé dans notre collaboration

avec le GMI : l'utilisateur fournit une grille de potentiels qui peut alors être facettisée pour de la reconstruction par exemple.

Bibliographie

- [AG01] Samir Akkouche and Éric Galin. Adaptive implicit surface polygonization using marching triangles. *Computer Graphics Forum*, 20(2):67–80, jun 2001. ISSN 0167-7055.
- [Akl98] Ergun Akleman. Implicit surface painting. *Proceedings of Implicit Surfaces'98*, 1998.
- [Ang01] Alexis Angelidis. Surfaces implicites multi-résolution. Mémoire de DEA, Institut National Polytechnique de Grenoble, Université Joseph Fourier, 2001.
- [ART] Simulateur d'arthroscopie de l'épaule. <http://www.lifl.fr/GRAPHIX/rechapli/simulateur/arthro-epaule/>.
- [Bar84] Alan H. Barr. Global and local deformations of solid primitives. *Computer Graphics*, 18(3):21–29, 1984.
- [Bat82] Klaus-Jürgen Bathe. *Finite Element Procedures in Engineering Analysis*. Prentice-Hall, 1982.
- [Bli78] Jim Blinn. Simulation of wrinkled surfaces. In *Siggraph'78*, 1978.
- [Bli82] James F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, July 1982.
- [Blo88] Jules Bloomenthal. Polygonisation of implicit surfaces. *Computer Aided Geometric Design*, 5:341–355, 1988.
- [Blo90] Jules Bloomenthal. Techniques for implicit modeling. In *Modeling and Animating with Implicit Surfaces*, pages 13.1–13.18, 1990. SIGGRAPH Course Notes 23.
- [Blo94] Jules Bloomenthal. An implicit surface polygonizer. *Graphics Gems*, IV, 1994.
- [Blo95] Jules Bloomenthal. Bulge elimination in implicit surface blends. In *Implicit Surfaces'95*, pages 7–20, Grenoble, France, April 1995. Proceedings of the first international workshop on Implicit Surfaces.
- [BS91] Jules Bloomenthal and Ken Shoemake. Convolution surfaces. *Computer Graphics*, 25(4):251–256, July 1991. Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 1991).

- [BS95] Carole Blanc and Christophe Schlick. Extended field functions for soft objects. In *Implicit Surfaces'95*, pages 21–32, Grenoble, France, April 1995. Proceedings of the first international workshop on Implicit Surfaces.
- [Can98] Marie-Paule Cani. Layered models with implicit surfaces. In *Graphics Interface (GI'98) Proceedings*, Vancouver, Canada, Jun 1998. Invited paper, published under the name Marie-Paule Cani-Gascuel.
- [CD97] Marie-Paule Cani and Mathieu Desbrun. Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 3(1), Mar 1997. Published under the name Marie-Paule Cani-Gascuel.
- [CDC⁺96] S. Cotin, H. Delingette, J.M. Clément, M. Bro-Nielsen, N. Ayache, and J. Marescaux. Geometrical and physical representations for a simulator of hepatic surgery. *Proceedings of Medicine Meets Virtual Reality 4 (MMVR-4), San Diego, California*, pages 139–151, jan 1996.
- [CGMS00] P. Cignoni, F. Ganovelli, C. Montani, and R. Scopigno. Reconstruction of topologically correct and adaptive trilinear isosurfaces. *Computers and Graphics*, 24(3):399–418, 2000.
- [CGS98] Benoît Crespin, Pascal Guitton, and Christophe Schlick. Efficient and accurate tessellation of implicit sweeps. *CSG'98 (Bath, UK)*, 1998. <http://dept-info.labri.u-bordeaux.fr/~crespin/cv/#publis>.
- [CH01] Marie-Paule Cani and Samuel Hornus. Subdivision curve primitives: a new solution for interactive implicit modeling. In *Shape Modelling International*, Italy, May 2001.
- [CK00] H.K. Cakmak and U. Kühnapfel. Animation and simulation techniques for vr-training systems in endoscopic surgery. *Workshop on Animation and Simulation*, 2000.
- [Coo84] R. Coons. Shade trees. In *Siggraph'84*, 1984.
- [Cre98] Benoît Crespin. *Modélisation et déformation de forme libre à base de surfaces implicites équipotentielles*. PhD thesis, Université de Bordeaux, dec 1998.
- [CS98] Benoît Crespin and Christophe Schlick. Generating implicit fields from in/out images. *Implicit Surfaces'98*, pages 91–98, jun 1998. is98.
- [DDBC99] Gilles Debunne, Mathieu Desbrun, Alan H. Barr, and Marie-Paule Cani. Interactive multiresolution animation of deformable models. In *Eurographics Workshop on Computer Animation and Simulation*, Sep 1999.
- [DDCB01] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic real-time deformations using space and time adaptive sampling. In *Computer Graphics Proceedings*, Annual Conference Series. ACM Press / ACM SIGGRAPH, Aug 2001. Proceedings of SIGGRAPH'01.

-
- [Des97] Mathieu Desbrun. *Modélisation et animation de matériaux hautement déformables en synthèse d'images*. PhD thesis, Institut National Polytechnique de Grenoble, Université Joseph Fourier, dec 1997.
- [dFdMGTV92] Luiz Henrique de Figueiredo, Jonas de Miranda Gomez, Demetri Terzopoulos, and Luiz Velho. Physically-based methods for polygonization of implicit surfaces. In *Graphics Interface'92*, pages 250–257, Vancouver, Canada, May 1992.
- [DG95] Mathieu Desbrun and Marie-Paule Gascuel. Animating soft substances with implicit surfaces. *Computer Graphics*, 1995. Proceedings SIGGRAPH'95.
- [DTG95] Mathieu Desbrun, Nicolas Tsingos, and Marie-Paule Gascuel. Adaptive sampling of implicit surfaces for interactive modelling and animation. In *Implicit Surfaces'95*, pages 171–186, Grenoble, France, April 1995. Proceedings of the first international workshop on Implicit Surfaces.
- [FCA96] Eric Ferley, Marie-Paule Cani, and Dominique Attali. Skeletal reconstruction of branching shapes. In John C. Hart and Kees van Overveld, editors, *Implicit Surfaces'96, The Second International Workshop on Implicit Surfaces*, pages 127–142, Eindhoven, The Netherlands, Oct 1996. Published under the name Marie-Paule Cani-Gascuel.
- [FCG00] Eric Ferley, Marie-Paule Cani, and Jean-Dominique Gascuel. Practical volumetric sculpting. *the Visual Computer*, 16(8):469–480, dec 2000. A preliminary version of this paper appeared in *Implicit Surfaces'99*, Bordeaux, France, sept 1999.
- [FDFH90] James D. Foley, Andries Van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics Principles and Practice. The Systems Programming Series*. Addison-Wesley Publishing Company, Inc., 1990.
- [For92] S. Fortune. Voronoi diagrams and delaunay triangulations, 1992.
- [FS95] Luiz Figueiredo and Jorge Solfi. Adaptive enumeration of implicit surfaces with affine arithmetic. In *Implicit Surfaces'95*, pages 161–170, Grenoble, France, April 1995.
- [FV97] Max Froumentin and Éric Varlet. Dynamic implicit surface tessellation. *Proceedings of ACM Symposium on Virtual Reality Software and Technologie*, pages 79–86, 1997.
- [GA00] Éric Galin and Samir Akkouche. Incremental polygonization of implicit surfaces. *Graphic Models and Image Processing*, 62(2):19–39, 2000.
- [Gas93] Marie-Paule Gascuel. An implicit formulation for precise contact modelling between flexible solids. *Computer Graphics*, 27:313–320, 1993.
- [Gas95] Jean-Dominique Gascuel. Implicit patches: An optimised and powerful ray in-

- tersection algorithm. In *Implicit Surfaces'95*, pages 143–160, Grenoble, France, April 1995. Proceedings of the first international workshop on Implicit Surfaces.
- [GH95] Cindy Grimm and John F. Hughes. Smooth isosurface approximation. In *Implicit Surfaces'95*, pages 57–76, Grenoble, France, April 1995. Proceedings of the first international workshop on Implicit Surfaces.
- [GL01] Claire Guilbaud and Annie Luciani. Visualisation de modèles physiques particuliers - cas des sables et des pâtes. *Huitièmes journées du Groupe de Travail Animation et Simulation de l'AFIG et du GDR ALP, Rennes*, jul 2001.
- [Gou94] Jean-Paul Gourret. *Modélisation d'Images Fixes et animées, Chapitre VI "Assemblage et Animation d'Objets Déformables : Simulation par Éléments finis"*. Masson, 1994.
- [GTS] GNU Triangulated Surface. <http://gts.sourceforge.net>.
- [GVP91] Marie-Paule Cani Gascuel, Anne Verroust, and Claude Puech. A modelling system for complex deformable bodies suited to animation and collision. *Journal of Visualization and Computer Animation*, aug 1991.
- [GW87] D.F. Gillies and C.B. Williams. An interactive graphic simulator for teaching of fiberoscopic techniques. *Proceedings of Eurographics'87 Conference, Amsterdam*, pages 127–138, aug 1987.
- [GW95] Andrew Guy and Brian Wyvill. Controlled blending for implicit surfaces. In *Implicit Surfaces'95*, pages 107–112, Grenoble, France, April 1995. Proceedings of the first international workshop on Implicit Surfaces.
- [Hab97] Arash Habibi. *Modèles Physiques Supports de la Relation Mouvement-Forme-Image*. PhD thesis, Institut National Polytechnique de Grenoble, janvier 1997. Thèse de doctorat en informatique.
- [Har93] John Hart. Ray tracing implicit surfaces. In *Modeling, Visualizing and Animating Implicit Surfaces*, pages 13.1–13.15, 1993. SIGGRAPH Course Notes 25.
- [HDH98] J. Hart, A. Durr, and D. Harsh. Critical points of polynomial metaballs. *Implicit Surfaces'98*, pages 69–76, June 1998. is98.
- [Hec98] P. Heckbert. Fast surface particle repulsion. *Report CMU-CS-97-130, April 1997. Appeared in SIGGRAPH 98 course notes*, 1998.
- [HGW92] A. Haritsis, D.F. Gillies, and C.B. Williams. Realistic generation and real time animation of images of the human colon. *Proceedings of Eurographics'92 Conference, Computer Graphics Forum, Cambridge*, pages 367–379, sep 1992.
- [HPF] Hyperfun project, language and software tools for f-rep geometric modeling. <http://www.hyperfun.org/>.
- [HSIW96] A. Hilton, A. J. Stoddart, J. Illingworth, and T. Windeatt. Marching triangles:

-
- range image fusion for complex object modelling. *International Conference on Image Processing*, 1996.
- [IMM] Immersion. <http://www.immersion.com/products/medical/products.shtml>.
- [Ito89] H. Ito. Ray tracing meta-balls. *Pixel*, unknown(77):76–80, 1989. In Japanese.
- [JDK97] A.C. Jambon, P. Dubois, and S. Karpf. A low-cost training simulator for initial formation in gynecologic laparoscopy. *Proceedings of CVRMed'97, Grenoble, France*, pages 347–356, mar 1997.
- [Kau00] Arie E. Kaufman. Introduction to volume graphics. *Course Notes of International Springschool on Visualization, Bonn-Röttgen*, mar 2000.
- [KB89] Devendra Karla and Alan H. Barr. Guaranteed ray intersections with implicit surfaces. *Computer Graphics*, 23(3):297–306, 1989.
- [KIS] Kismet (kinematic simulation, monitoring and off-line programming environment for telerobotics). <http://www-kismet.iai.fzk.de>.
- [LC86] Annie Luciani and Claude Cadoz. Utilisation de modèles mécaniques et géométriques pour la synthèse et le contrôle d'images animées. *2ème colloque CESTA, Nice*, pages 704–711, 1986.
- [LC87] William Lorensen and Harvey Cline. Marching cubes: a high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987. Proceedings of SIGGRAPH'87 (Anaheim, California, July 1987).
- [LKM01] Erik Lindholm, Mark J. Kilgard, and Henry Moreton. User-programmable vertex engine. *Siggraph'2001*, 2001. VIDIA Corporation.
- [MER] Mitsubishi electric research laboratories. <http://www.merl.com/>.
- [Mes97] Philippe Meseure. *Modélisation de corps déformables pour la simulation d'actes chirurgicaux*. PhD thesis, Université des Sciences et Techniques de Lille, janvier 1997.
- [MI87] S. Murakami and H. Ichihara. On a 3D display method by metaball technique. *Journal of papers given at the Electronic Communication*, J70-I(8):1607–1615, 1987. In Japanese.
- [Mit90] Don P. Mitchell. Robust ray intersection with interval arithmetic. In *Graphics Interface'90*, pages 68–74, 1990.
- [MJC00] Benjamin Mora, Jean-Pierre Jessel, and René Caubet. Accelerating volume rendering with quantized voxels. *IEEE/ACM SIGGRAPH Symposium on Volume Visualization and graphics 2000, Salt Lake City, Ut, USA*, pages 63–70, oct 2000.
- [MKC⁺95a] P. Meseure, S. Karpf, C. Chaillou, P. Dubois, and J.F. Rouland. Low-cost medical simulation: a retinal laser photocoagulation simulator. *Proceedings of the Graphics'Interface 95, Québec City, Québec*, pages 155–162, may 1995.

- [MKC⁺95b] P. Meseure, S. Karpf, C. Chaillou, P. Dubois, and J.F. Rouland. Sophocle: a retinal laser photocoagulation simulator - overview. *Proceedings of CVRMed'95, Nice, France*, pages 105–114, apr 1995.
- [NB93] Paul Ning and Jules Bloomenthal. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, 13(6):33–41, November 1993.
- [NHK⁺85] Hitoshi Nishimura, Makoto Hirai, Toshiyuki Kawai, Toru Kawata, Isao Shirakawa, and Koichi Omura. Object modeling by distribution function and a method of image generation. *The Trans. of the Inst. of Elec. and Comm. Eng. of Japan*, J68-D(4):718–725, 1985. In Japanese (transl. into English by T. Fujiwara).
- [OC97] Agata Opalach and Marie-Paule Cani. Local deformations for animation of implicit surfaces. *SCCG'97 (Bratislava, Slovakia)*, jun 1997. <http://w3imagis.imag.fr/Membres/Marie-Paule.Cani/mouImplicite.html>.
- [OM93] Agata Opalach and Steve Maddock. Implicit surfaces: Appearance, Blending and Consistency. In *Fourth Eurographics Workshop on Animation and Simulation*, pages 233–245, Barcelona, Spain, September 1993.
- [PASS95] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 11(8):429–446, 1995.
- [PDR97] F. Peugnet, P. Dubois, and J.F. Rouland. Clinical assessment of a training simulator for retinal photocoagulation. *Proceedings of CVRMed'97, Grenoble, France*, mar 1997.
- [Ped95] Hans K ohling Pedersen. Decorating implicit surfaces. *Siggraph Proceedings*, 1995.
- [Ped96] Hans K ohling Pedersen. A framework for interactive texturing on curved surfaces. *Siggraph Proceedings*, 1996.
- [Pfi99] Hanspeter Pfister. Architectures for real-time volume rendering. *Journal of Future Generation Computer Systems (FGCS)*, 15(1):1–9, feb 1999.
- [PPP88] Alexander Pasko, V. Pilyugin, and V. Pokrovskij. Geometric modeling in the analysis of trivariate functions. *Computers and Graphics*, 12(3/4):457–465, 1988.
- [PW89] A. Pentland and J. Williams. Good vibrations: Modal dynamics for graphics and animation. *Siggraph Proceedings*, pages 215–222, aug 1989.
- [RNN00] Y. R emion, J.M. Nourrit, and O. Nocent. Dynamic animation of n-dimensional deformable objects. *WSCG'2000 (Plzen, Czech Republic)*, feb 2000.
- [SBM⁺94] M.A. Sagar, D. Bullivant, G.D. Mallison, P.J. Hunter, and J.W. Hunter. A virtual environment and model of the eye for surgical simulation. *Siggraph'94, Conference Proceedings, Computer Graphics annual conference series, Orlando*, pages 205–212, jul 1994.
- [SBX] Symbionix. <http://www.symbionix.com/>.

-
- [SDS96] E. Stollnitz, T. Derose, and D. Salesin. *Wavelets for Computer Graphics, Chapitre VI "Subdivision Curves"*. Morgan Kaufman, San Francisco, California, 1996.
- [Sen] Sensable. <http://www.sensable.com/>.
- [SIM] Simedge. <http://www.simedge.com/>.
- [SP86] T. Sederberg and S. Parry. Free form deformations of solid geometric models. *Siggraph'86 Conference Proceedings, Computer Graphics*, pages 151–160, aug 1986.
- [SP98] V. Savchenko and A. Pasko. Transformation of functionally defined shapes by extended space mappings. *The Visual Computer, No. 5/6*, 14:257–270, 1998.
- [SPA] Simulateur pédagogique d'anniocentèse et de cordocentèse. <http://www.lifl.fr/GRAPHIX/rechappli/simulateur/spac/>.
- [TMC99] Frédéric Triquet, Philippe Meseure, and Christophe Chaillou. Affichage interactif de surfaces implicites. *AFIG'99 (Reims, France)*, 1:69–77, nov 1999.
- [TMC01] Frédéric Triquet, Philippe Meseure, and Christophe Chaillou. Fast polygonization of implicit surfaces. *WSCG'2001 (Plzen, Czech Republic)*, 2:283–290, feb 2001. <http://wscg.zcu.cz>.
- [Ton89] David Tonnesen. Ray-tracing implicit surfaces resulting from the summation of bounded polynomial functions. Technical Report TR-89003, Rensselaer Design Research Center, Rensselaer Polytechnic Institute, Troy, New York, 1989.
- [VC97] E. Varlet and C. Chaillou. Modèle géométrique et modéleur pour un simulateur pédagogique d'écho-endoscopie digestive (s.p.e.e.d.). *Journée 'Modeleurs géométriques*, sep 1997.
- [VCDM97] E. Varlet, C. Chaillou, P. Dubois, and V. Maunoury. A simulator for initial training in ultrasound endoscopy. *World Congress on Medical Physics and Biomedical Engineering, Nice*, sep 1997.
- [VdFG99] Luiz Velho, Luiz Henrique de Figueiredo, and Jonas Gomes. A unified approach for hierarchical adaptive tessellation of surfaces. *ACM Transactions on Graphics*, 18(4):329–360, 1999.
- [Vel90] Luiz Velho. Adaptive polygonisation of implicit surfaces using simplicial decomposition and boundary constraints. In C. E. Vendoni and D. A. Duce, editors, *Eurographics'90*, pages 125–136, 1990.
- [Vel95] Luiz Velho. Adaptive polygonization made simple, 1995.
- [Vel96] Luiz Velho. Simple and efficient polygonization of implicit surfaces. *Journal of Graphics Tools: JGT*, 1(2):5–24, 1996.
- [VTK] Visual ToolKit. <http://www.kitware.com/>.
- [WH94] Andrew Witkin and Paul Heckbert. Using particles to sample and control implicit surfaces. *Computer Graphics*, pages 269–278, July 1994. Proceedings of SIGGRAPH'94.

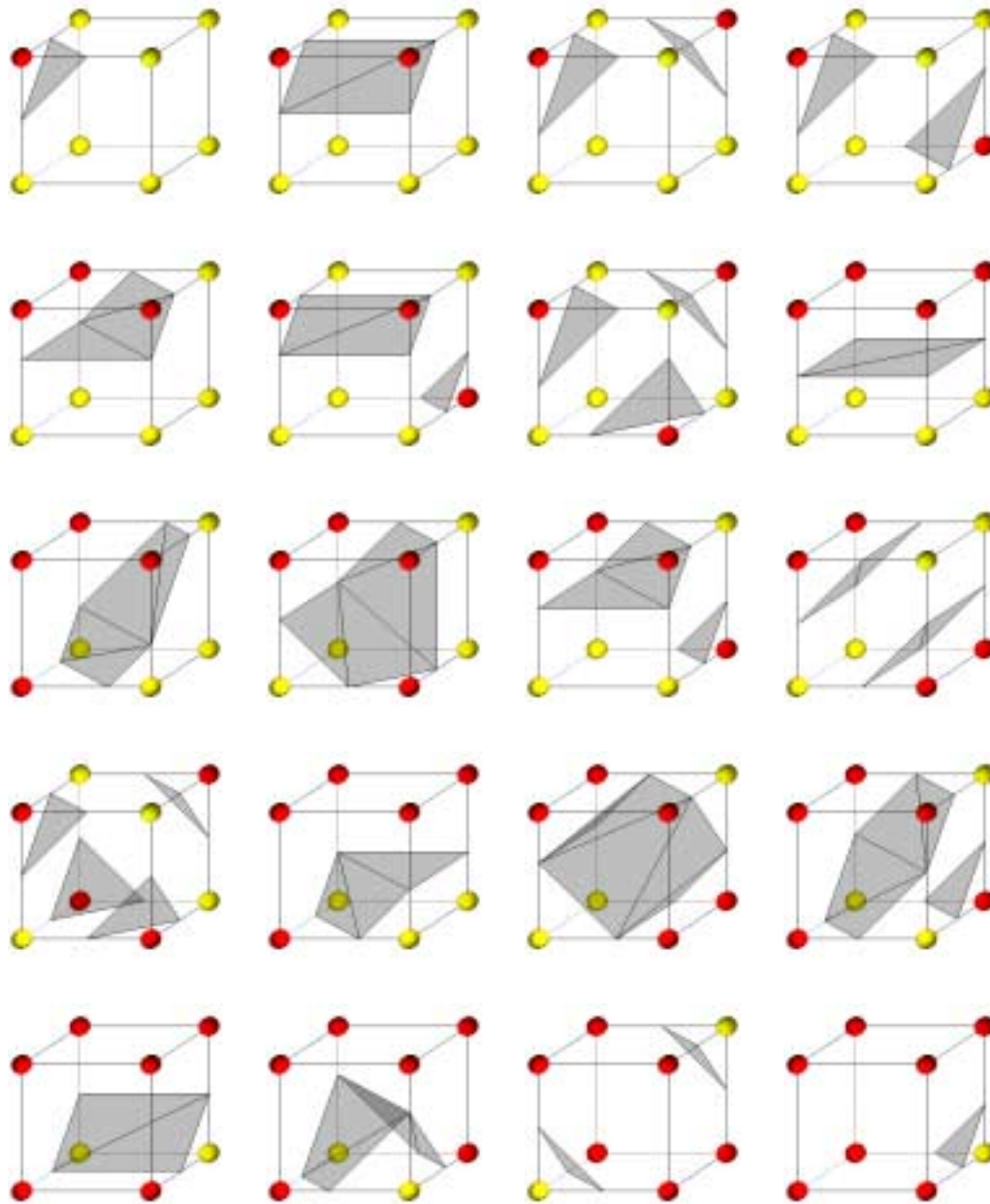
- [Wib00] Luc Wibaux. *Une méthode de synthèse d'images échographiques : application à des simulateurs médicaux d'entraînement*. PhD thesis, Université des Sciences et Techniques de Lille, juillet 2000.
- [WMW86a] Brian Wyvill, Craig McPheeters, and Geoff Wyvill. Animating soft objects. *The Visual Computer*, pages 235–242, August 1986.
- [WMW86b] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structure for soft objects. *The Visual Computer*, pages 227–234, August 1986.
- [WMW87] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Solid texturing of soft objects. *IEEE Computer Graphics and Applications*, 7(12):20–26, December 1987.
- [WT90] Geoff Wyvill and Andrew Trotman. Ray-tracing soft objects. In *Computer Graphics International'90*, pages 469–475, 1990.
- [WW89] Brian Wyvill and Geoff Wyvill. Field functions for implicit surfaces. *The Visual Computer*, 5:75–82, December 1989.
- [WW90] A. Witkin and W. Welch. Fast animation and control of nonrigid structures. *Siggraph'90 Conference Proceedings, Computer Graphics*, pages 243–252, aug 1990.
- [Wyv90] Geoff Wyvill. Texturing implicit surfaces. In *Modeling and Animating with Implicit Surfaces*, pages 15.1–15.5, 1990. SIGGRAPH Course Notes 23.
- [XIT] Xitact, virtual patient. <http://www.xitact.com/>.
- [ZJ91] Cornelia Zuhlten and Hartmut Jürgens. Continuation methods for approximating isovalued complex surfaces. In *Eurographics'91*, pages 5–19, Vienne, Autriche, September 1991.

Annexes

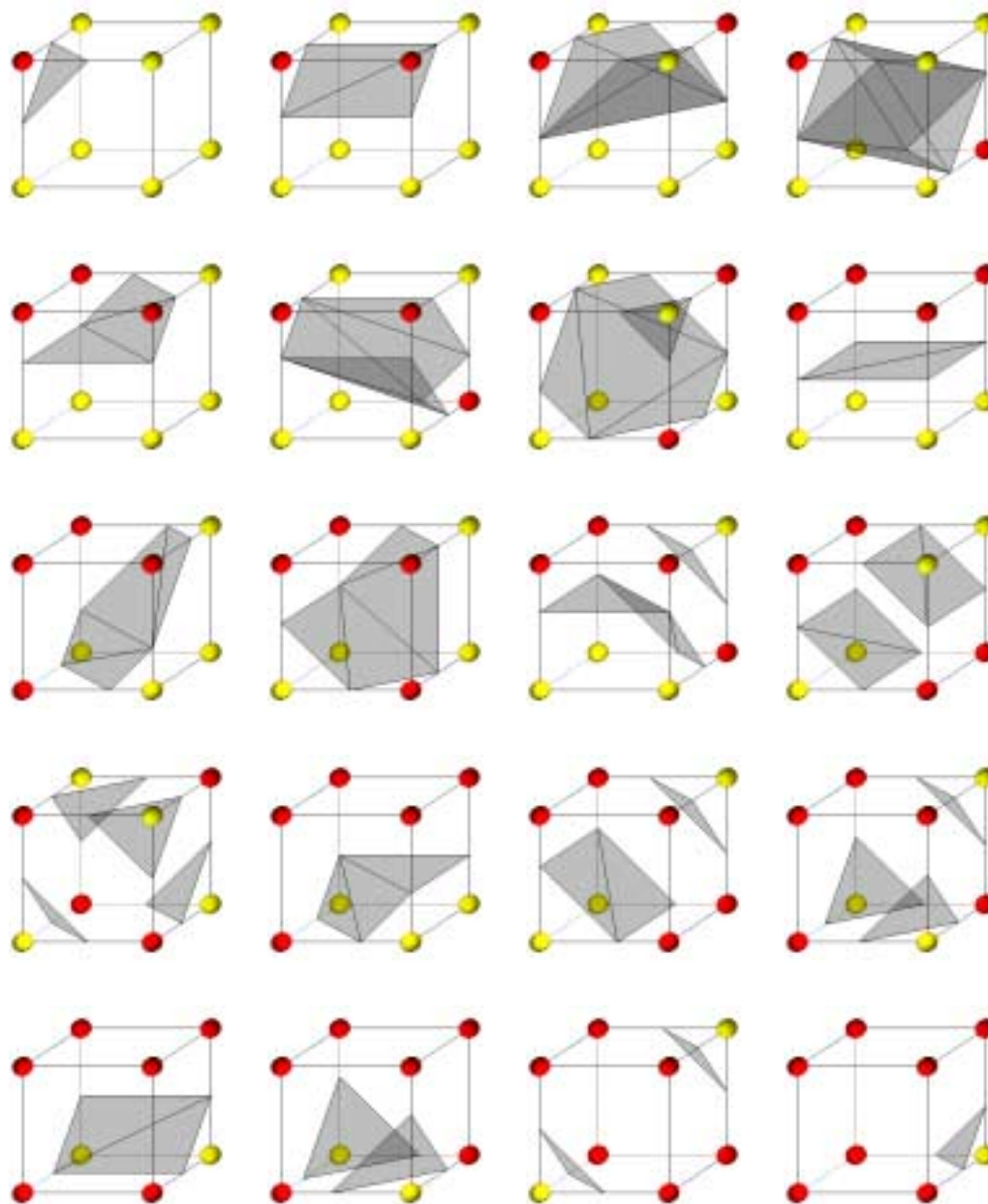
Annexe A

Tables de facettisation

A.1 Table “séparation”



A.2 Table “fusion”



Annexe B

Vue d'ensemble du logiciel développé au CRAS de Valence (Espagne)

