CHAPITRE 6

Interface avec l'utilisateur

EAUCOUP du réalisme d'un simulateur provient de la qualité des images affichées. Le mouvement de l'objet décrit lors des précédents chapitres ne concerne que celui de particules représentant l'intérieur de l'objet. Nous allons dans ce chapitre décrire comment la *surface* de l'objet va rendre compte de ces déplacements. Nous détaillerons également la façon dont ont été implémentées la détection et la réponse aux collisions avec un objet virtuel, et en particulier le calcul de la force renvoyée quand on utilise un dispositif à retour d'effort.

C'est donc, plus généralement, de la gestion de l'interface avec l'utilisateur que nous allons maintenant discuter. Cette partie va permettre d'utiliser les résultats des chapitres précédents en décrivant la dernière couche du prototype de simulateur laparoscopique que nous avons créé.

1 Affichage de la surface

La surface sert non seulement à reproduire les déformations calculées de l'objet, mais aussi, dans notre cas, à cacher l'utilisation d'un processus multirésolution à l'utilisateur. Celui-ci n'a en effet pas à être troublé par la variation du nombre de particules utilisées.

1.1 Liaison avec les particules

La surface des objets sera constituée de triangles, primitive classique et optimisée pour l'affichage d'une surface. Le nombre de triangles composant la surface devra être un compromis entre qualité et rapidité, dépendant des capacités graphiques de la machine.

Ce choix n'affecte pas la résolution des maillages volumiques utilisés pour la simulation, la surface et le modèle physique interne étant *totalement* décorrélés. En pratique, notre maillage triangulaire sera pour la plupart des objets celui correspondant au maillage tétraédrique le plus fin, composé de quelques milliers de triangles.

La surface sera rattachée aux particules internes à l'aide de *liens*. Ceux-ci sont des vecteurs d'*offset* **o** fixes, qui définissent la position **n** d'un nœud de la surface directement à partir de celle **p** d'une particule interne (Fig. 6.1) :

$$\mathbf{n} = \mathbf{p} + \mathbf{o}$$

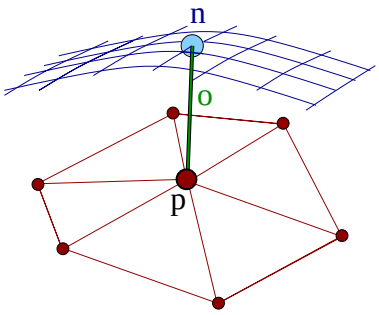


FIG. 6.1: Un nœud de la surface est lié à une particule du modèle interne grâce à un décalage **o**.

Gestion de la multirésolution

Un précalcul permet de déterminer quelle est la particule la plus proche de tout nœud de la surface ainsi que le décalage **o** associé. Cette recherche est en fait réalisée dans *tous* les maillages, de sorte qu'un nœud pourra déduire sa position de celle de n'importe quel maillage, et en pratique de celui qui sera actif dans cette zone.

La définition des fils d'une particule comme étant les particules du maillage inférieur situées à l'intérieur de sa région de Voronoï assure qu'un nœud de la surface sera lié à des particules faisant partie de la même sous-branche de l'arbre : si un nœud est lié à p dans un maillage, ce nœud sera lié à l'une des filles de p dans le maillage inférieur, et ainsi de suite.

Parmi les différents points (un par maillage) auxquels un nœud est lié, un seul sera donc actif à tout moment de la simulation. Lorsque la discrétisation interne évolue et entre deux affichages successifs, il est donc très facile à chaque nœud de déterminer quelle est le point actif parmi ceux auxquels il est lié. Si ce n'est celui qu'il a utilisé pour déterminer sa position à l'affichage précédent, c'est donc son fils (ou son père).

On recherche donc, en partant du niveau utilisé à l'affichage précédent, quel est le niveau réellement actif pour ce nouvel affichage. Ce sera très souvent le même, parfois celui directement supérieur ou inférieur et très rarement un niveau plus éloigné. Cette recherche est donc très rapide.

Lissage de la position

La méthode précédemment décrite a l'inconvénient de déplacer trop grossièrement les nœuds de la surface lorsque la simulation utilise des maillages volumiques internes de faible résolution. Dans ces cas là, on déduit en effet rigidement la position de quelques milliers de nœuds de celle des quelques dizaines de particules simulées. La conséquence est que le mouvement de la surface se fait par *plaques*, chacune étant liée à l'une des particules internes actives.

Pour éviter ce problème et lisser les positions des nœuds de la surface, nous utilisons un filtrage de la position de *plusieurs* particules internes pour déterminer la position d'un nœud. Nous avons choisi encore une fois la simplicité et la rapidité en effectuant une interpolation linéaire de la position de plusieurs particules.

Un nœud va être lié à trois particules, celles-ci formant le triangle au dessus duquel il se trouve (voir Fig. 6.2). Sa position sera alors une somme pondérée des positions décalées des particules :

$$\mathbf{n} = \frac{\sum_{i \in 1..3} w^i(\mathbf{p}^i + \mathbf{o}^i)}{\sum_{i \in 1..3} w^i} \qquad w^i = \frac{1}{||\mathbf{o}^i||} \tag{6.1}$$

Les coefficients pondérateurs w^i sont pris comme étant inversement proportionnels aux distances entre le nœud et les trois points $\mathbf{p}^i$ (analogie avec coordonnées barycentriques de la projection du nœud sur le triangle). Ils sont renormalisés de façon à ce que leur somme fasse 1 et que l'on évite ainsi d'effectuer une division inutile. Toutes ces informations (w^i, $\mathbf{p}^i$ et $\mathbf{o}^i$ pour chaque niveau de la hiérarchie) sont précalculées et stockées dans les nœuds de la surface.

Un nœud est ainsi capable de déterminer, avec plus ou moins de précision, sa position à partir de celles des triangles du modèle physique. Il utilisera à un instant donné de la simulation le triangle mis-à-jour (i.e. composé de points actifs ou fantômes) offrant la meilleure précision, c'est-à-dire appartenant au niveau le plus fin (voir Fig. 6.2). On obtiendra ainsi un déplacement optimal du nœud en fonction de la précision des déformations disponible dans sa zone.

Il peut arriver avec cette méthode que la position d'un nœud change soudainement lors des subdivisions ou des regroupements de points, le nœud changeant alors de triangle de référence. Pour éviter cet effet de *popping*

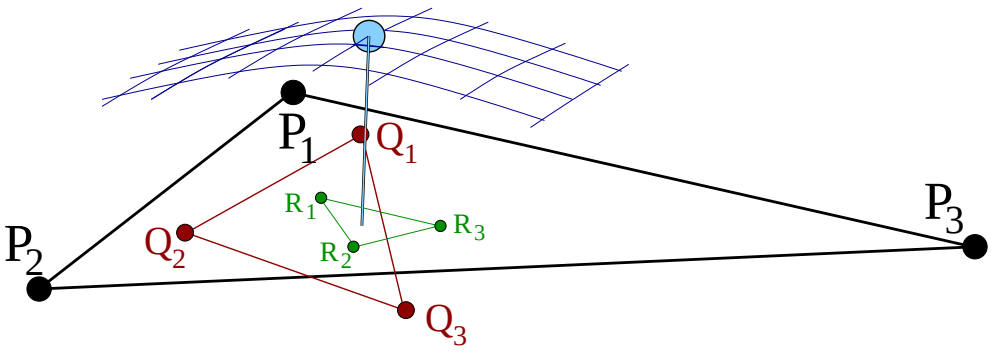


FIG. 6.2: Un nœud de la surface est lié aux différents triangles surfaciques des maillages tétraédriques.

de la surface, on pourra ajouter un léger filtrage temporel permettant de passer continuement, sur quelques images, d'un triangle à un autre.

On peut aussi limiter ce phénomène en jouant sur les seuils de subdivision et de regroupement pour contrôler à partir de quelle distance de la position d'équilibre (qui par construction ne produit aucun *popping*) on décide de subdiviser, limitant ainsi les déplacements brusques.

Ce modèle à base de décalage rigide entre la surface et le modèle interne est en théorie limité à de faibles déplacements, et devrait en particulier mal supporter les rotations de l'objet. En pratique, les vecteurs de décalage $\mathbf{o}^i$ sont suffisamment petits pour que des rotations raisonnables ne produisent pas d'effet visuel trop notable.

Si c'était le cas, il suffirait de définir les décalages $\mathbf{o}^i$ dans un repère *local* lié à chaque triangle. Cela demanderait simplement de mettre à jour la normale de ces triangles, ce que nous n'avons pas jugé nécessaire de faire.

2 Collisions avec l'outil

Le simulateur chirurgical va consister principalement en un objet déformable (un modèle de foie dans nos exemples) et un outil virtuel manipulé par le chirurgien. Nous allons voir comment ces deux objets interagissent.

2.1 Détection de la collision

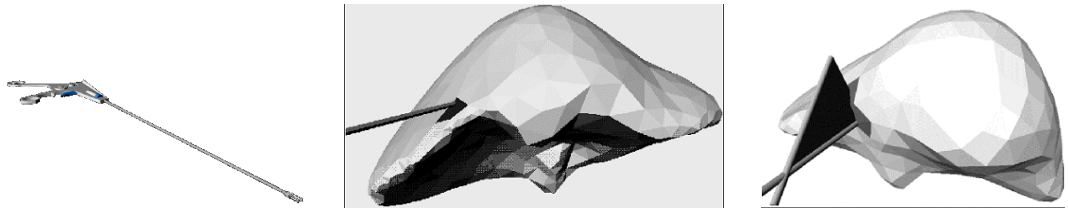
Les méthodes classiques de détection de collision sont bien adaptées aux objets rigides (partitionnement de l'espace et boîtes englobantes pour accélérer la recherche). Pour traiter la collision avec un objet déformable, il convient d'utiliser d'autres méthodes.

Dans le cadre de l'action incitative AISIM [INR] a été mis au point une méthode très rapide basée sur l'utilisation du matériel graphique [LCN99]. Profitant du fait qu'un seul des objets est déformable et que l'outil est de forme assez simple (voir Fig. 6.3a), l'idée est de placer une caméra orthographique à l'intérieur de l'outil pour obtenir rapidement les parties de la surface qui sont éventuellement intersectées. Un rendu hors-écran (*offscreen*) de la surface de l'objet, vue au travers de cette caméra, et tirant ainsi parti des performances graphiques de la machine, peut fournir la liste des triangles intersectés.

Si l'outil est déplacé rapidement, et en fonction de la fréquence des détections, il peut avoir parcouru une grande distance entre deux détections de collision successives. Ce cas peut être traité en utilisant une caméra projective classique et des plans de *clipping* qui définiront l'espace dans lequel s'est déplacé l'outil (voir Fig. 6.3c).

Même rapide (150 fois plus qu'avec des algorithmes de partitionnement de l'espace comme *Rapid*), cette détection reste assez chère. Aussi avons-nous choisi de ne la faire qu'avant chaque affichage de la scène et non pas à chaque pas de simulation. En faisant ainsi coïncider affichage et détection de collision, on s'assure que l'affichage prendra bien en compte la collision qui vient éventuellement d'avoir lieu.

Les mouvements d'un chirurgien étant assez lents et la détection se faisant à environ 25Hz, le déplacement de l'outil est au plus de quelques millimètres entre deux détections successives et il n'est donc pas nécessaire d'utiliser en pratique la version dynamique de la méthode(Fig. 6.3c) ou une détection plus fréquente.



(a) (b) (c)

FIG. 6.3: Une caméra placée à l'intérieur de l'outil (a) permet de détecter rapidement les collisions avec l'objet, que l'on considère l'outil comme statique (b) ou dynamique (c).

2.2 Réponse à la collision

Une fois la collision détectée, reste à la propager au modèle physique interne. Cette partie est loin d'être évidente et la solution proposée, bien que rapide, est parfois mise en défaut en particulier quand la taille de l'objet est importante (ce qui n'est pas le cas avec les outils laparoscopiques).

La détection de collisions a renvoyé une liste de polygones, qui sont les intersections des triangles formant la surface avec le volume de vue de la caméra (*frustum*). Le traitement décrit ici va être appliqué à chacun de ces triangles.

On calcule tout d'abord le barycentre B du polygone d'intersection entre le triangle et la caméra (voir Fig. 6.4). Les trois sommets du triangle seront ensuite déplacés au *prorata* des coefficients barycentriques de B dans le triangle.

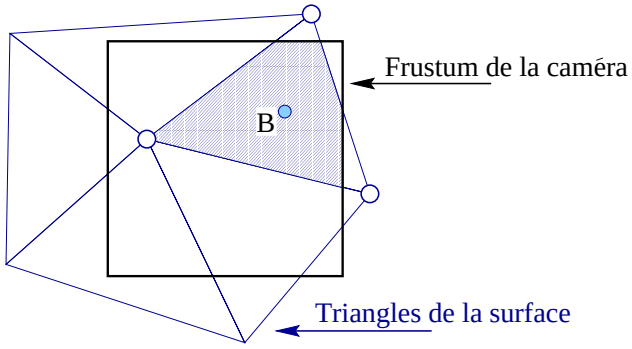


FIG. 6.4: Pour chacun des triangles intersectés, on calcule le centre de gravité B de la zone d'intersection. Les trois sommets du triangle seront déplacés en fonction des coefficients barycentriques de ce point.

Déterminer la meilleure direction de déplacement des nœuds du triangle est assez complexe. Nous avons choisi de la diriger selon l'opposée de la normale du triangle, de sorte que la surface sera déplacée le plus efficacement possible. Sa norme est telle que le centre de gravité B sorte de l'outil. On ne garantit donc pas que le triangle sera totalement repoussé hors de l'outil, ce qui serait difficile à calculer, mais on fait en sorte qu'il sorte suffisamment. En quelques itérations de l'algorithme, le triangle sera presque totalement hors de l'outil.

Un coefficient permet de régler à quel point on désire pousser B hors de l'outil. Il peut être utile de le régler en fonction de la rigidité de l'objet, mais un coefficient de 1 (le point B est repoussé exactement à la limite de l'outil) donne de bons résultats, la surface ne paraissant pas rester en collision.

Une fois les points de la surface déplacés, il reste à communiquer ce mouvement aux particules internes. Ceci est fait en utilisant le même processus que celui qui avait permis, dans l'autre sens, de déterminer la position de la surface à partir de celle des particules. Les particules internes sont en effet déplacées en utilisant les poids w^i et les décalages $\mathbf{o}^i$ (voir Eq. 6.1) qui les lient aux nœuds de la surface.

Chaque particule peut recevoir plusieurs déplacements, pondérés par les w^i, de la part des différents nœuds de la surface qui lui sont liés. Elle moyennera donc à la fin du processus de réponse aux collisions les déplacements qu'elle a reçus pour savoir quelle est la réelle influence de l'outil sur sa position.

3 Retour d'effort

En plus d'un affichage et d'un mouvement de qualité, le retour haptique est une composante importante d'un simulateur réaliste. Différents systèmes permettent de transmettre à l'utilisateur une force dépendant de ses gestes et de la simulation. On pourra se référer à [Cot97, MPT99] pour un inventaire des techniques existantes.

Nous avons utilisé un dispositif de retour haptique de type *PHANToM*, produit par *Sensable Technologies* [Tec]. Donnant la position de ses 6 degrés de liberté, il permet en retour de transmettre à l'utilisateur une force de direction et d'intensité données.

FIG. 6.5: Le Phantom desktop.

3.1 Principe

Toutes les particules déplacées lors de la détection de collision seront stockées dans une liste. Lors des pas de simulation qui auront lieu entre deux détections de collision, ces particules calculeront bien la force qui leur est appliquée, mais ne l'*intégreront* pas. À la place, elle transmettront cette force à l'utilisateur, qui sentira donc l'effet de son geste et devra contrecarrer cette force pour maintenir l'objet dans sa position déformée.

On simule entre 16 et 128 pas de simulation entre chaque détection de collision ce qui avec un affichage à 25-30Hz correspond à un retour haptique d'une fréquence allant entre 400 et 3700 Hz, compatible avec les sensations fines du toucher.

3.2 Lissage de la force

Si l'on se contente de sommer les forces issues de toutes les particules déplacées pour les renvoyer à l'outil, on risque d'obtenir des discontinuités dans la force. Un très léger déplacement de l'outil peut en effet le faire entrer en collision avec un nouveau triangle de la surface, de nouvelles particules internes venant ajouter leur force. On aura alors une soudaine et artificielle augmentation de la force reçue (et inversement une diminution si on quitte un triangle).

Pour éviter ce problème, il suffit de quitter cette version discrétisée de la surface pour repasser à une version continue. La force ne sera pas alors directement liée au nombre de particules déplacées, mais plutôt à la *surface* de la zone de collision. On moyenne donc les forces issues des particules déplacées et on multiplie cette force moyenne par la surface de contact.

Il apparaît un autre problème avec la méthode précédemment décrite. Lorsque l'on déplace les particules internes après la détection de collision, celles-ci bougent soudainement puis restent immobiles pendant tous les pas de temps suivants, se contentant de transmettre la force calculée. Ce déplacement brusque crée une soudaine variation de la force, qui évolue ensuite lentement durant les pas de simulation suivants.

Pour éviter ce problème, il suffit d'appliquer *progressivement* le déplacement lié à la collision. On divise donc le déplacement imposé par les 16, 32, 64 ou 128 pas qui vont avoir lieu et on applique cette petite partie de déplacement à chaque pas, la collision s'appliquant ainsi continuement entre deux détections.

Le dernier problème rencontré avec le retour d'effort vient de la réponse aux collisions qui repousse chaque triangle de plus en plus hors de l'outil. Arrive un moment où le triangle est totalement sorti de l'outil et n'entre donc plus en collision. Les particules précédemment immobiles qui y sont attachées se retrouvent libres et se déplacent donc d'un coup vers leur position de repos. Elles entrent alors forcément en contact avec l'outil qui n'a pas bougé énormément. Celui-ci les repousse fortement, créant un brusque à-coup dans la force transmise.

Cet inconvénient peut être supprimé en empêchant que le triangle puisse être totalement sorti hors de l'outil. Au delà d'un certain seuil (0.1mm par exemple), on considère la collision comme suffisamment gérée et l'on arrête de repousser le triangle. On évite ainsi ce brusque retour dans l'outil du triangle libéré.

Il faut par contre faire en sorte d'éviter que le triangle ne "colle" à l'outil, le suivant dans ses déplacements puisqu'on l'empêche de totalement sortir. On pourra alors comparer la direction de mouvement de l'outil, la normale à la surface et la force subie par la particule pour repérer les cas où l'on cherche à sortir de l'objet et où il faut laisser sortir le triangle.

4 Conclusion

Nous avons décrit dans ce chapitre la façon dont sont reliées les particules internes et la surface affichée de l'objet. Assez simples, ces méthodes permettent d'obtenir rapidement un affichage correct dissimulant à l'utilisateur les changements de discrétisation interne. Le retour haptique apporte un grand réalisme à la simulation et il convient de bien le régler pour que des discontinuités ne viennent pas perturber les sensations obtenues.

Le travail décrit dans ce chapitre a été réalisé en collaboration avec différents stagiaires intervenant dans le cadre des actions incitatives INRIA *AISIM* et *CAESARE*. Ils ont principalement contribué à l'intégration dans le simulateur des travaux réalisés dans le cadre de l'action incitative sur l'aspect rendu de l'organe simulé.

Pierre-Olivier Agliati a créé l'ossature du simulateur et y a intégré l'affichage des reflets d'une lampe virtuelle sur l'objet. Antoine Leroy et Sylvain Trimoreau ont durant leur stage ajouté la gestion des effets dynamiques (gouttes de sang, blanchiment, brûlures) qui peuvent apparaître sur la surface. Ils ont également modifié la structure de données de la surface (utilisation de *triangles fans* et simplification de maillage) et ont parallélisé le code.

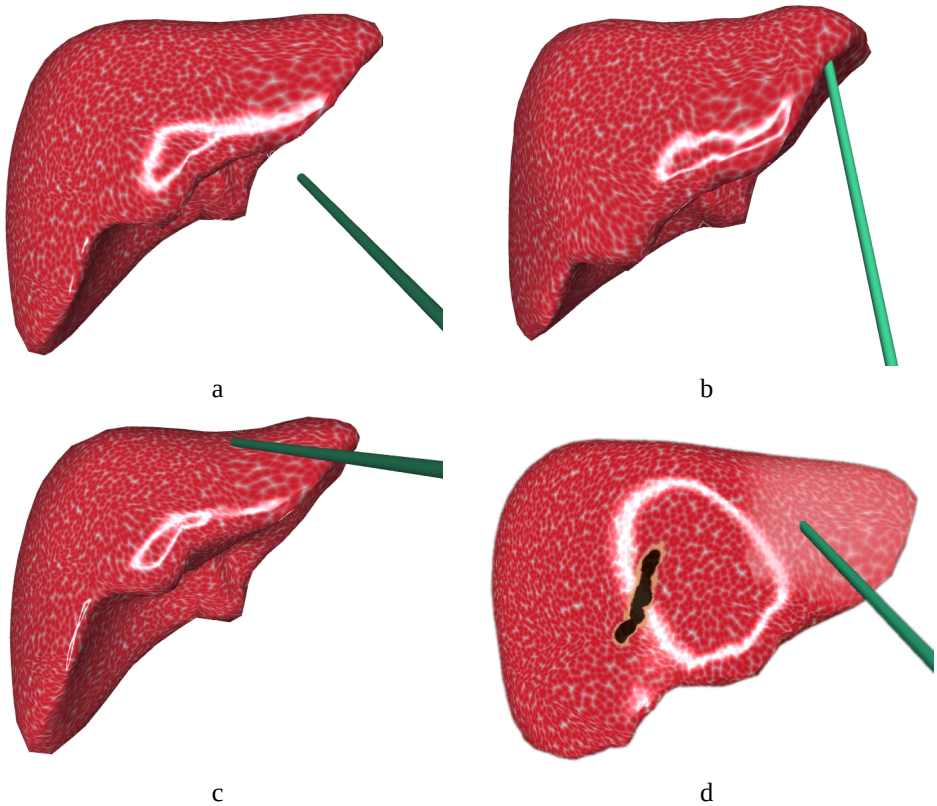
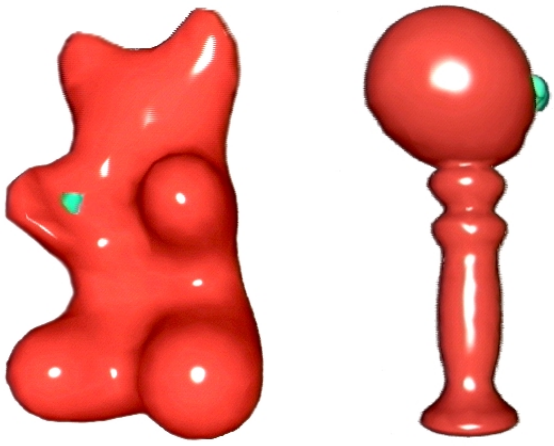
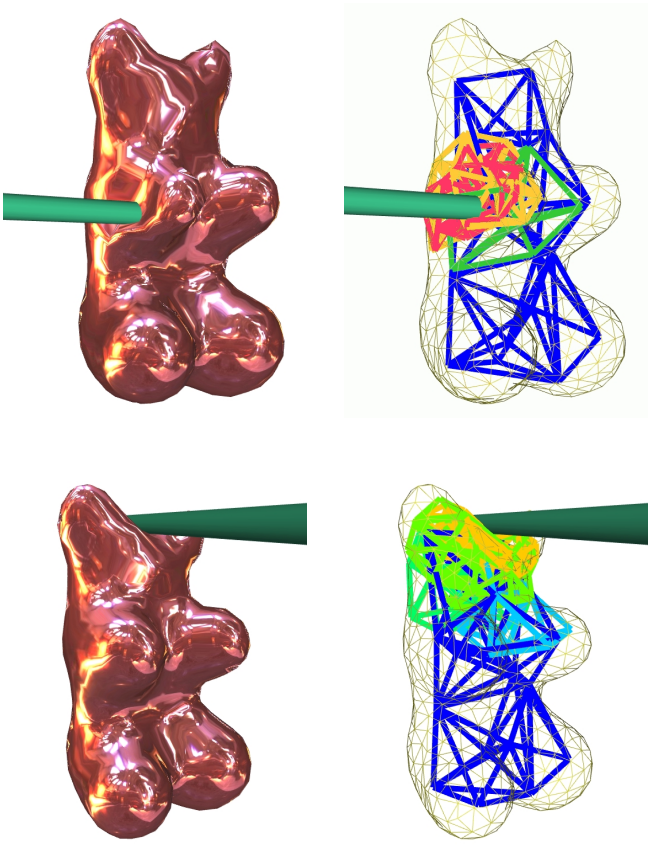


FIG. 6.6: Images du simulateur montrant quelques déformations du foie. La Figure (d) montre quelques effets de surface (brûlures, blanchiment, reflets).



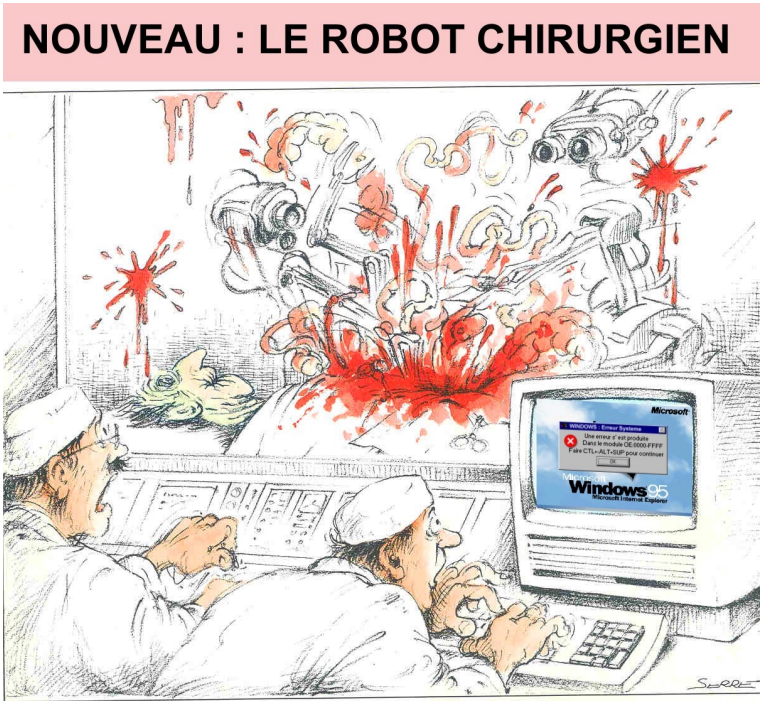
a b

FIG. 6.7: D'autres exemples d'animation, aux comportements plus ou moins rigides en fonction des paramètres choisis.



a b

FIG. 6.8: Le rendu final utilise des textures d'environnement qui explicitent la déformation subie (a). Le maillage interne correspondant (b).



"Intel products are not intended for use in medical, life saving, or life sustaining applications."

Intel Copyright Notice