Université de Montréal

# Animation interactive de mouvements secondaires par simulation de surfaces élastiques

par

Yannick Simard

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la faculté des études supérieures

en vue de l'obtention du grade de

Maître ès sciences (M.Sc.)

en informatique

décembre 2005

Université de Montréal
Faculté des études supérieures

Ce mémoire de maîtrise intitulé

# Animation interactive de mouvements secondaires par simulation de surfaces élastiques

présenté par
Yannick Simard

a été évalué par un jury composé des personnes suivantes :

Neil F. Stewart
président-rapporteur

Pierre Poulin
directeur de recherche

Balázs Kégl
membre du jury

# Sommaire

En synthèse d'animation, les mouvements secondaires sont des mouvements apparaissant en réponse aux actions d'un personnage ou d'un objet animé. Des exemples d'objets typiquement impliqués dans des mouvements secondaires sont les vêtements, la peau ou les cheveux d'un personnage. Nous proposons un modèle inspiré de la physique, qui est simple, rapide et stable, simulant des mouvements secondaires à l'aide de surfaces élastiques décrites à partir de maillages triangulaires animés quelconques. Les sommets et les arêtes des régions déformables d'un modèle géométrique sont respectivement convertis en masses ponctuelles et ressorts linéaires. Notre méthode se base sur un schéma d'intégration qui utilise des contraintes géométriques plutôt que des forces, ce qui nous permet d'atteindre le niveau d'efficacité des approches explicites en même temps qu'une stabilité semblable aux approches implicites. Les paramètres contrôlant la simulation sont intuitifs et simples à incorporer au modèle géométrique. Notre technique est adaptée aux contraintes particulières des jeux vidéo puisqu'elle est robuste aux actions arbitraires de l'usager. De plus, elle permet de simuler une vaste gamme de matériaux allant des surfaces élastiques aux surfaces très raides, grâce à un ordonnancement des contraintes en partant des masses fixes. Nous démontrons l'efficacité de notre méthode en l'utilisant dans des applications interactives, où l'usager contrôle un personnage ayant certaines régions élastiques et portant des accessoires simulés.

**Mots clés :**

Synthèse d'animation, animation basée sur la physique, modélisation basée sur la physique, simulation en temps réel, mouvements secondaires, animation de tissus, surfaces flexibles, corps mous.

# Abstract

In animation synthesis, secondary motion appears in response to the actions of an animated character or object. Examples of typical objects whose motion can be described as secondary motion are a character's clothing, skin, and hair. We propose a simple, fast, and stable physically-based model which simulates elastical surfaces, created from an arbitrary animated triangular mesh. The vertices and edges of the deformable regions of the geometrical model are converted respectively into point-masses and linear springs. We use an integration scheme based on geometrical constraints, rather than forces, to attain the level of performance of an explicit integration scheme coupled with stability approaching that of an implicit integration scheme. Simulation parameters are intuitive and easily added to the geometrical model. Since it is robust for arbitrary user input, our technique is adapted to the unique constraints of video games. The range of surfaces simulable by our method is broad, including elastical surfaces as well as stiff surfaces, due to an ordering of constraints ordering that starts from the fixed masses. We tested the efficiency of our method in interactive applications, where the user controls a character incorporating elastic geometrical regions and wearing simulated accessories.

**Keywords :**

Animation synthesis, physically-based animation, physically-based modeling, real-time simulation, cloth animation, secondary motions, flexible surfaces, soft bodies.

# Table des matières

**Remerciements** **xi**

**1 Introduction** **1**

**2 Systèmes d'animation** **4**

2.1 Squelette et os . . . . . 4

2.2 *Keyframes* . . . . . 6

2.3 Poses . . . . . 7

2.4 *Skinning* . . . . . 8

2.5 Contraintes pratiques . . . . . 8

2.5.1 *Pipeline* graphique . . . . . 9

2.5.2 Connectivité géométrique . . . . . 10

2.5.3 Couplage animation-simulation . . . . . 11

**3 Schémas d'intégration** **12**

3.1 Schémas explicites . . . . . 13

3.2 Schémas implicites . . . . . 15

3.3 Schémas semi-implicites . . . . . 17

3.4 Schémas par contraintes géométriques . . . . . 18

**4 Travaux antérieurs** **21**

4.1 Surfaces élastiques . . . . . 21

4.1.1 Modèles géométriques . . . . . 21

4.1.2 Forces élastiques . . . . . 23

4.1.3 Frottement . . . . . 27

4.2 Objets mous . . . . . 29

4.2.1 Déformation de l'espace . . . . . 29

4.2.2 Éléments finis . . . . . 30
4.2.3 Nuage de particules . . . . . 30
4.3 Détection de collisions pour surfaces élastiques . . . . . 31
4.3.1 Structures hiérarchiques et subdivision . . . . . 32
4.3.2 Validité géométrique . . . . . 33

**5 Notre modèle** **40**
5.1 Modèle physique . . . . . 41
5.2 Schéma d'intégration par contraintes géométriques . . . . . 43
5.2.1 Ressort linéaire . . . . . 44
5.2.2 Contraintes de pliage . . . . . 44
5.2.3 Frottement aérodynamique, contact et gravité . . . . . 45
5.3 Collisions . . . . . 45
5.3.1 Sphère . . . . . 48
5.3.2 Cylindre . . . . . 49
5.3.3 Triangle . . . . . 49
5.4 Étendue de l'applicabilité de notre modèle . . . . . 50
5.4.1 Ordonnancement des contraintes . . . . . 52
5.5 Modélisation artistique des paramètres physiques . . . . . 54

**6 Résultats** **56**

**7 Travaux futurs** **61**
7.1 Modèles à couches multiples . . . . . 61
7.2 Implémentation distribuée . . . . . 62
7.2.1 Calcul sur GPU . . . . . 62
7.2.2 Microprocesseur à noyaux multiples . . . . . 63

**8 Conclusion** **65**

**Bibliographie** **67**

# Table des figures

2.1 (a) Personnage articulé ; (b) Hiérarchie associée. . . . . 5

2.2 Illustration, en tons de gris, du poids de l'os $b2$ en fonction de la position sur le bras. Par exemple, le point $i$ indiqué possède la paire de poids $(w_i^{b1}, w_i^{b2}) = (0.5, 0.5)$. . . . . 6

3.1 Une itération du schéma implicite. . . . . 15

4.1 Gros plan sur un morceau de tissu de lin réel. . . . . 22

4.2 (a) Tissu suspendu par le coin supérieur ; (b) Particules simulées (sphères) et triangles du maillage (fils de fer en blanc). . . . . 23

4.3 Ressorts structurels (lignes grises) et de cisaillement (lignes noires) attachés à des particules (disques). . . . . 24

4.4 (a) Feuille flexible accrochée ; (b) chapeau entrant en collision avec le sol. Images tirées de l'article de Grinspun *et al.* [GHDS03]. . . . . 25

4.5 Élément de pliage de l'arête $\overline{34}$ d'angle diédral $(\pi - \theta)$. Figure tirée de l'article de Bridson *et al.* [BMF03]. . . . . 26

4.6 Interaction avec une tige flexible subdivisée respectivement en un, trois et cinq groupes de particules. Un usager tire sur le bout de la tige afin de la faire plier. Cette image provient de l'article de Müller *et al.* [MHTG05]. 31

4.7 Représentation de trois niveaux d'une hiérarchie de volumes englobant. Figure tirée de l'article de Teschner *et al.* [TKH$^+$05]. . . . . 32

4.8 Illustration de l'instant où a lieu chacun des deux types de collision possibles entre deux triangles. Pour faciliter l'interprétation, nous hachurons les arêtes cachées, nous numérotons les sommets définissant impliqués dans chaque collision, et nous ajoutons une flèche indiquant l'endroit où la collision a lieu. La figure (a) montre une collision entre le sommet d'un triangle et une face de l'autre triangle. La figure (b) montre la collision entre deux arêtes, chacune appartenant à un triangle différent. . . . . 35

4.9 (a) Personnage animé avec chandail simulé ; (b) Les bras du personnage intersectent son torse à la hauteur des aisselles. Ces deux illustrations sont tirés de l'article de Baraff *et al.* [BWK03]. . . . . 37

4.10 Courbe d'intersection. Figure tirée de l'article de Baraff *et al.* [BWK03]. 38

5.1 Ajout de régions simulées à des modèles géométriques. (a) Ajout d'une cape à un personnage. (b) Ajout d'une couche de peau sur un cube. . . 42

5.2 Mèche de cheveux simulée avec deux primitives de collisions servant à prévenir l'intersection géométrique de la mèche avec le personnage. La tête du personnage est simplifiée par une sphère et le cou, par un cylindre. Le rendu est fait avec polygones texturés en (a) et fils de fer (*wireframe*) en (b). Notons que la mèche suit la courbure de la sphère et s'appuie correctement sur le cylindre. . . . . 46

5.3 Sphère et objectif géométrique lors de l'application de la contrainte de non-pénétration. . . . . 48

5.4 Cylindre et objectif géométrique lors de l'application de la contrainte de non-pénétration. . . . . 49

5.5 Triangle et objectif géométrique lors de l'application de la contrainte de non-pénétration. L'extérieur d'un triangle est le côté normal à son plan de support. . . . . 50

5.6 Différences obtenues lors d'un pas de simulation d'un système de trois particules sur lequel deux contraintes $c_1$ et $c_2$ s'appliquent, selon que l'ordre d'application des contraintes soit $(c_1, c_2)$ ou $(c_2, c_1)$. . . . . 51

6.1 Illustration d'un bras dont le biceps est simulé avec notre méthode. (a) Modèle original. (b) Biceps simulé, s'étirant sous l'effet de la gravité. (c) Particules simulées pour créer un biceps mou. . . . . 57

6.2 Un personnage virtuel possédant des biceps mous. La figure (a) montre le personnage vu de haut, immobile. La figure (b) montre l'étirement des biceps du personnage lorsque ce dernier est en accélération angulaire dans le sens horaire. . . . . 57

6.3 Personnage virtuel possédant un nez mou. Trois situations sont ici montrées : le personnage est immobile en (a), le nez s'écrase, dans la figure (b), lorsque le personnage accélère brusquement vers la gauche et le nez s'allonge, dans la figure (c), lorsque le modèle accélère vers la droite. . . 58

6.4 Bande de tissu raide, suspendue par ses deux coins supérieurs, une fois immobilisée. Comparaison entre notre méthode avec ordonnancement des contraintes en (a) et sans ordonnancement en (b). Notons que la bande en (a) converge vers la solution théorique à un epsilon machine près ; en (b), la bande est environ 20% plus longue que sa longueur au repos. . . 58

6.5 Accélération vers la droite des points d'attache d'une bande de tissu raide accrochée par ses deux coins supérieurs. La figure (a) illuste le résultat obtenu en utilisant notre modèle avec ordonnancement des contraintes, alors que la figure (b) est obtenue sans ordonnancement. Nous avons indiqué par un échelon noir, dans la figure (b), la longueur de la bande au repos afin de bien faire remarquer que la bande voit sa longueur plus que doubler, alors qu'en (a), elle n'augmente que de 1 % lors d'une accélération semblable. . . . . 59

6.6 Collisions avec des primitives. La figure (a) montre une bande de tissu retenue par ses deux extrémités en contact avec un cylindre, alors que la figure (b) montre cinq bandes retenues par une seule de leurs extrémités lors du contact avec une sphère. . . . . 59

6.7 Bandoulière simulée attachée à une arme militaire. Remarquons la courbe caractéristique que prend naturellement la bandoulière sous l'effet de la force gravitationnelle, pour les deux inclinaisons illustrées. *Artwork ©2006 Electronic Arts Inc. All right reserved.* . . . . . 60

# Remerciements

Ce mémoire est le point final d'une aventure qui a commencé voilà un peu plus de deux ans. Plusieurs personnes ont influencé mon parcours et c'est pourquoi j'aimerais souligner ici leur importance.

Merci à Pierre Poulin, mon directeur de recherche, pour avoir partagé son temps et sa connaissance encyclopédique de l'infographie. Je le remercie également pour sa compréhension, son sens de l'humour et sa patience.

Merci à mes compagnons de route du LIGUM, avec lesquels j'ai eu de nombreuses discussions instructives et divertissantes. Ils sont : Philippe Beaudoin, Nicolas Bergeron, Simon Clavet, Jean-François Dufort, François Duranleau, Emric Epstein, Martin Granger-Piché, Luc Leblanc et Jean-François St-Amour.

Mille mercis à mes parents, Roberte et Gilles, qui m'ont transmis le plaisir d'apprendre. Je leur suis profondément reconnaissant de m'avoir soutenu inconditionnellement, autant du point de vue émotif que financier, pendant les nombreuses années qu'a duré mon parcours académique.

Je remercie tout particulièrement Marie-Isabelle Hodgson, pour son soutien, ses encouragements et sa précieuse présence dans ma vie.

Finalement, j'aimerais remercier l'entreprise *Electronic Arts* et le CRSNG pour leur généreux appui financier. De même, j'en profite pour remercier les talentueux membres du studio *EA Montréal*, m'ayant accueilli au sein de leur équipe.

# Remarques

## Terminologie

Lorsqu'un terme technique est employée en français dans le texte, nous indiquons, en italique entre parenthèses, son équivalent usuel en anglais. Certains termes, qui n'ont pas d'équivalent courant en français sont directement en anglais en italique.

## Droits d'auteur

Certaines images de ce document ont pu être tirées d'articles publiés dans des journaux ou des conférences scientifiques. La provenance des images est clairement indiquée dans les figures et la référence au travail est également fournie.

# Chapitre 1

# Introduction

*Pour atteindre la vérité, il faut une fois dans la vie se défaire de toutes les opinions qu'on a reçues, et reconstruire de nouveau tout le système de ses connaissances.*

René Descartes

L'augmentation fulgurante de la puissance des ordinateurs et des consoles de jeux vidéo au cours des 20 dernières années a rendu possible l'apparition d'applications interactives de plus en plus réalistes. Dans les années 1980, les jeux vidéo se déroulaient entièrement dans un univers bidimensionnel et l'animation assistée par ordinateur au cinéma commençait à peine, alors que de nos jours, le domaine du divertissement présente des personnages interagissant physiquement avec leur environnement, projetant des ombres crédibles et dont les détails vont souvent jusqu'à leurs pores de peau. De surcroît, l'acquisition d'animations automatisée (*motion tracking*) ainsi que le travail intuitif de talentueux artistes permettent maintenant aux personnages de se mouvoir de façon très convaincante.

Malgré tout, étant donné que l'œil humain est habile à déceler ce qui semble invraisemblable, même un personnage ayant les mouvements les plus naturels ne confond personne lorsque sa chevelure demeure immobile ou que ses vêtements amples n'ondulent pas lorsqu'il marche. Ces détails souvent oubliés, appelés "mouvements secondaires", contribuent grandement à rendre physiquement plausible l'animation des objets et des personnages dans les jeux vidéo.

Lorsqu'il est question d'animer les vêtements ou les cheveux d'un personnage, des milliers de sommets de maillages polygonaux servent souvent à représenter leur géométrie. Compte tenu que chaque sommet est libre, le nombre de degrés de liberté est extrêmement élevé pour quiconque voudrait générer manuellement le mouvement d'un tissu ou de chaque mèche de cheveux. De plus, la dynamique de ces mécanismes physiques est difficile à reproduire intuitivement. Par exemple, le frottement entre les mèches d'une longue chevelure influence grandement son comportement, mais son impact sur l'animation des cheveux est difficile à prédire pour un artiste.

C'est entre autres pour cette raison que, pour les applications interactives, de plus en plus de techniques, inspirées de la physique, transfèrent le fardeau du dos des artistes vers le matériel et le logiciel, alors responsables de générer l'évolution des objets simulés en fonction des contraintes spécifiées par l'usager. En général, l'artiste identifie les objets à simuler physiquement et les contraintes à satisfaire, telles que l'élasticité des matériaux, la gravité, ou encore les régions interdites de la scène. Cette dernière contrainte est habituellement gérée lors de la phase de la simulation appelée la "détection de collisions".

Étant donné que plusieurs aspects d'un jeu vidéo compétitionnent pour la mémoire disponible et le temps de calcul, deux denrées limitées dans les consoles de jeux, le budget computationnel allouable pour la simulation physique est restreint à une fraction de seconde par image rendue.

Malgré tout, remarquons qu'il est possible d'obtenir des résultats réalistes, même si le calcul est contraint à être court, puisqu'il est difficile pour un observateur de discerner entre un mouvement synthétique plausible et la réalité. Il est donc possible d'utiliser des modèles physiques approximatifs sans affecter grandement la qualité visuelle de l'animation. La difficulté de l'observateur à discerner entre une animation plausible et une animation exacte est due au fait que les objets simulés sont des mécanismes physiques complexes qui, lorsqu'en mouvement, ont un comportement pratiquement impossible à prédire sans l'aide de calculs. Même si une certaine liberté est ainsi donné lors de la création d'un modèle de simulation, certaines propriétés sont intuitivement nécessaires, comme l'inertie des objets qui fait continuer sa course à la chevelure d'un personnage lors d'un arrêt brusque, et ce, quelle que soit la précision du modèle employé.

Un autre problème relié à l'interactivité de l'usager sur l'environnement virtuel est celui de la stabilité de la simulation. En effet, il est nécessaire pour l'algorithme de si-

mulation de ne jamais se retrouver dans des situations où la qualité du résultat pourrait se dégrader visuellement de façon permanente. Des mouvements secondaires invraisemblables attireraient facilement le regard au détriment de tout le reste de l'environnement.

Le chapitre 2 propose une introduction sur les concepts d'animation utiles à la compréhension de nos travaux. Nous y discutons également des contraintes qu'impose le fonctionnement typique des systèmes d'animation dans les jeux vidéo. Le chapitre 3, quant à lui, définit le concept de "schéma d'intégration", facette primordiale à toutes les méthodes de simulation numériques, et donne un aperçu de la stabilité de quelques grandes familles de schémas. Ensuite, le chapitre 4 jette un regard sur les résultats antérieurs servant de base à nos travaux.

Dans cette recherche, nous proposons une méthode simple et très rapide pour simuler les mouvements secondaires d'objets et de personnages animés, à l'aide de surfaces élastiques discrétisées en particules (masses ponctuelles), dont le comportement est régi par des contraintes d'élasticité et de non-pénétration. La description de notre méthode se retrouve au chapitre 5.

Notre modèle inspiré de la physique répond à plusieurs nécessités propres au domaine des jeux vidéo : stabilité, efficacité et robustesse. Notre modèle permet de simuler très rapidement et avec une stabilité numérique garantie des matériaux élastiques, comme de la peau et des vêtements, et dont la robustesse aux états géométriques invalides est suffisante pour résister aux actions arbitraires d'un usager.

Du côté de la modélisation des objets simulés, nous proposons un processus basé sur la peinture de sommets (*vertex painting*), facile à incorporer dans les logiciels existants. L'artiste y spécifie aisément les parties d'un objet à simuler, choisit les paramètres physiques de façon simple et intuitive, et incorpore de l'information géométrique pour la détection de collisions.

Finalement, nous proposons une façon simple d'adapter notre modèle de façon à simuler des matériaux ne s'étirant peu ou pas comme des mèches de cheveux, des sangles ou de la corde. Pour la simulation, nous traduisons les forces agissant sur le système sous la forme de contraintes géométriques bien définies et nous utilisons un ordonnancement des contraintes pour améliorer le traitement des matériaux raides.

Nous présentons nos résultats dans le chapitre 6, discutons des améliorations possibles à apporter à notre modèle dans le chapitre 7, et concluons dans le chapitre 8.

# Chapitre 2

# Systèmes d'animation

*Nous sommes automates dans les trois quarts de nos actions.*

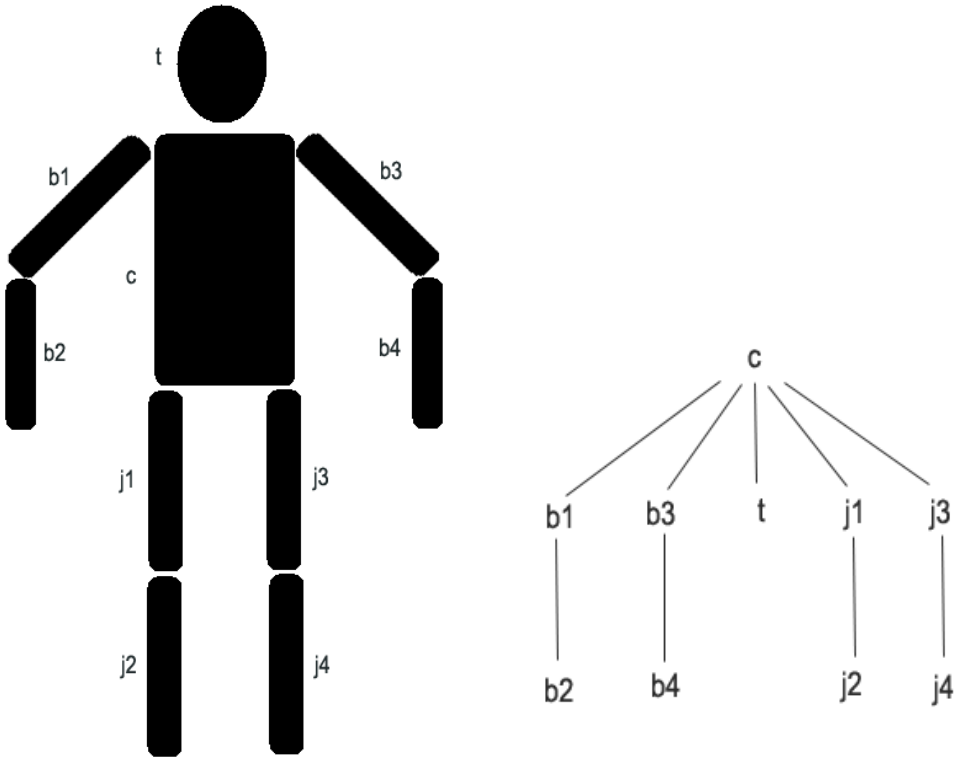
Leibniz

L'animation est une illusion du mouvement provoquée par la succession rapide d'images statiques. Comme le système visuel humain prend un certain temps à réagir aux variations de stimuli, phénomène appelé la "persistance visuelle", une séquence d'images cohérente dans le temps est interprétée par le cerveau comme un continuum. Si les images ne sont pas affichées en succession suffisamment rapide, ou encore si les paires d'images adjacentes dans la séquence varient trop brusquement, l'illusion est alors perdue au profit d'une impression de mouvement saccadé.

## 2.1 Squelette et os

Dans les jeux vidéo, la géométrie d'un personnage animé est d'abord définie dans un logiciel de modélisation comme *Maya* [Ali], *3D Studio Max* [Dis] ou *Softimage XSI* [Sof]. L'artiste lui associe ensuite des sections articulées logiques, organisées dans une arborescence représentant la dépendance qu'ont les référentiels des sections les uns par rapport aux autres, *i.e.* chaque transformation géométrique appliquée sur une section se propage à tous ses descendants dans la hiérarchie. La hiérarchie complète d'un objet animé est appelé le "squelette" (*skeleton*) de l'objet et ses sections sont les "os" (*bones*). Un squelette peut être réutilisé pour plusieurs personnages ayant la même physionomie,

ce qui est courant, compte tenu de la tendance, par souci d'économie, à recycler les données dans les jeux vidéo pour les personnages de moindre importance.




(a) (b)

FIG. 2.1 – (a) Personnage articulé ; (b) Hiérarchie associée.

La figure 2.1 (a) illustre un personnage avec des sections articulées typiques (tête, bras, jambes), alors que la figure figure 2.1 (b) illustre la hiérarchie associée à ce personnage. Ainsi, si un artiste décide d'appliquer une rotation sur la section $j3$, autour de l'articulation reliant $c$ et $j3$, non seulement l'articulation entre $j3$ et $j4$ sera préservée, mais $j4$ subira la même rotation que son parent. Notons que, dans cette figure, nous avons volontairement omis d'illustrer la géométrie aux points d'articulation, nous concentrant uniquement sur la portion de géométrie n'appartenant clairement qu'à un seul os du squelette.

Chaque sommet du maillage habillant le squelette se voit associer un poids pour chaque os influençant son mouvement. Pour définir correctement une pondération, la somme des poids d'un sommet se doit d'être l'unité. L'utilité de cette pondération est d'augmenter la qualité de l'animation des sommets au niveau des articulations, où les

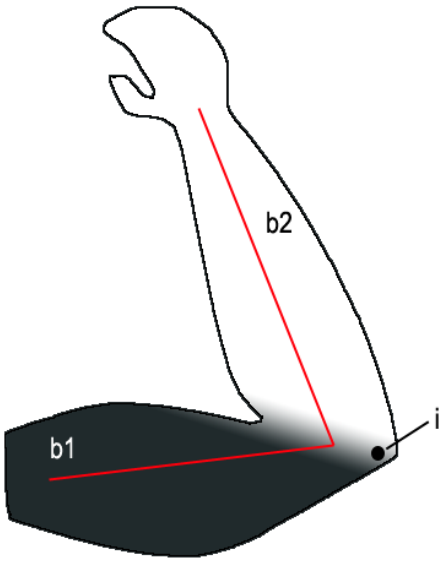



FIG. 2.2 – Illustration, en tons de gris, du poids de l'os $b2$ en fonction de la position sur le bras. Par exemple, le point $i$ indiqué possède la paire de poids $(w_i^{b1}, w_i^{b2}) = (0.5, 0.5)$.

polygones risquent de s'intersecter s'ils ne suivent qu'un seul os. Par exemple, dans la figure 2.2, soit un sommet $i$ se trouvant près de l'intersection entre les os $b1$ et $b2$. Si nous appelons les poids des os $b1$ et $b2$ respectivement $w_i^{b1}$ et $w_i^{b2}$, il est typique d'avoir $w_i^{b2}$ qui varie de 1 (blanc) à 0 (noir), à mesure que l'on s'approche de l'intersection des deux os en partant de la main, alors que $w_i^{b1} = (1 - w_i^{b2})$, permettant de réduire considérablement les risques d'intersection géométrique lors du *skinning*, processus expliqué plus loin.

## 2.2 *Keyframes*

L'appellation *keyframe* provient de l'animation traditionnelle, où les mouvements complexes sont décomposés en "images clés". Ainsi, lorsqu'un artiste décide du style de la démarche d'un personnage, il définit la position et l'orientation des différents os du squelette du personnage à certains moments clés d'une boucle de marche (normalement deux enjambées). Par exemple, si un personnage boite, l'un des moments clés sera celui où, plutôt que de mettre son poids entièrement sur sa jambe "sensible", le personnage met en mouvement sa "bonne jambe" en avance comparativement à un personnage dont les deux jambes sont en aussi bonne santé.

Noter que pour produire une séquence animée à 24 images par seconde, la fréquence

de rafraîchissement cinématographique, le nombre d'images clés requis est souvent de l'ordre de quatre à six, ce qui signifie qu'un effort supplémentaire considérable doit être déployé pour générer les images intermédiaires.

De l'âge d'or de l'animation traditionnelle dans les années 1930 jusqu'à tout récemment, les longs-métrages d'animation requéraient une armée de dessinateurs de raccords (*inbetweeners*) pour produire les images entre les *keyframes*. Leur travail consistait en somme à interpoler la configuration des objets animés.

Avec l'arrivée de l'animation tridimensionnelle assistée par ordinateur, la tâche des artistes s'est modifiée quelque peu. Lors du processus de génération des différentes séquences d'animation du personnage, où une séquence correspond à un mouvement particulier, l'artiste manipule un personnage par le biais de son squelette comme on manipule une figurine articulée, et enregistre les configurations clés (position et orientation de chaque os) servant à définir artistiquement le mouvement. Le temps de chaque *keyframe*, temps relatif au début de l'animation courante, est aussi enregistré.

## 2.3 Poses

La pose d'un personnage est définie comme la posture de son squelette à un instant donné. Il s'agit d'une combinaison particulière de positions et orientations de ses os. Dans une application interactive, comme un jeu vidéo, la pose du personnage contrôlé par l'usager est déterminée en interpolant les transformations de chaque os du squelette en fonction de l'action du personnage et du temps actuel. Les deux *keyframes* bornant le moment où la pose est évaluée sont utilisés pour l'interpolation en question.

En pratique, les positions et orientations des os du personnage sont données relativement à une pose canonique appelée la "pose de référence" (*bind pose*). La pose de référence est la configuration du personnage tel que modélisé originalement par l'artiste, alors qu'aucune animation n'est appliquée. Lors de la création de la pose de référence, chaque os $k$ est transformé avec la matrice $\mathbf{M}_k^{Ref}$, la matrice de référence, amenant l'os $k$ dans sa configuration initiale dans le référentiel local de l'objet.

Mathématiquement, les transformations sont sous forme matricielle. Au moment où est évaluée la pose, une matrice de transformation $\mathbf{A}_k$ est associée à chaque os $k$. Alors, la matrice $\mathbf{M}_k$ permettant le passage du référentiel local de $k$ dans la pose de référence

au référentiel monde est donnée par la récurrence

$$\mathbf{M}_k = \mathbf{A}_k \mathbf{M}_{parent(k)} \tag{2.1}$$

dont $\mathbf{M}_{parent(racine)}$, le cas de base, est la matrice identité et *racine* est la racine de la hiérarchie. Soulignons que $\mathbf{A}_{racine}$ peut varier dans le temps et n'est pas nécessairement la matrice identité. Sa variation, appelée "mouvement de la racine" (*root motion*), peut être utilisée dans les animations jouées en boucles[1] pour déterminer la vitesse de déplacement de l'objet dans la scène.

## 2.4 *Skinning*

Pour pouvoir faire l'affichage du modèle dans sa pose actuelle, un processus appelé *skinning* est utilisé, ce dernier consistant à appliquer une transformation géométrique sur chaque sommet du maillage de l'objet animé pour l'amener dans le référentiel monde.

Comme nous l'avons vu dans la section 2.1, à chaque sommet est associée une pondération d'influence de différents os.

Soient $\mathbf{x}_i$ la position du sommet $i$ dans le référentiel local de l'objet dans la pose de référence et $\mathbf{x}_i^{monde}$ la position en espace monde du sommet une fois l'animation appliquée. Le *skinning* du sommet $i$ s'exprime alors de la façon suivante :

$$\mathbf{x}_i^{monde} = \sum_{k \in \Omega(i)} w_i^k \cdot \mathbf{M}_k \cdot (\mathbf{M}_k^{Ref})^{-1} \cdot \mathbf{x}_i \tag{2.2}$$

où $\Omega(i)$ est l'ensemble des os ayant une influence sur le sommet $i$, $w_i^k$ est le poids de l'os $k$ sur le sommet $i$, et $\mathbf{M}_k^{Ref}$ est la matrice de référence introduite dans la section 2.3.

## 2.5 Contraintes pratiques

La simulation de mouvements secondaires d'un personnage dépend naturellement des mouvements de ce dernier. Nous discutons maintenant des contraintes rencontrées en pratique lors de l'intégration à un système d'animation d'un simulateur de mouvements secondaires basé sur la physique.

[1] marche, course, etc.

### 2.5.1 *Pipeline* graphique

Le *pipeline* graphique est la séquence d'opérations s'effectuant sur les données géométriques qui sont transmises de l'application vers le matériel d'accélération graphique. Cette séquence comprend :

- la transmission des sommets, des polygones et des matrices de transformation représentant les objets animés de la scène, le point de vue de la caméra, la position des lumières et la pose actuelle des squelettes associés aux objets,
- les opérations par sommet : appliquent les transformations géométriques positionnant chaque sommet par rapport à la caméra, déplacent les sommets animés en appliquant les transformations requise par la pose actuelle du squelette associé et calculent l'illumination des sommets, et
- les opérations par pixel : reçoivent le résultat de l'étape précédente, projettent les sommets en espace image et remplissent les polygones en interpolant les coordonnées de texture, les normales, etc., et calculant ainsi la couleur finale de chaque pixel.

Les cartes graphiques ont jusqu'à récemment possédé un *pipeline* graphique fixe, n'offrant qu'une variété limitée d'opérations par sommet et par pixel, mais les nouvelles cartes permettent de remplacer les étapes d'opérations par sommet et par pixel du *pipeline* par des programmes, appelés *shaders*, spécialement conçus pour s'exécuter sur le processeur graphique (GPU ou *Graphics Processing Unit*). Ces programmes s'exécutant par sommet et par pixel sont respectivement identifiés en tant que *vertex shaders* et *pixel shaders*.

Dans le contexte du *pipeline* graphique programmable, chaque sommet est l'agrégation de toutes les propriétés servant à le définir, souvent un sous-ensemble de : position, normale, tangente, binormale, coordonnées de textures et indices et poids des os du squelette associé. Pour être transmis au matériel graphique, les sommets de chaque objet sont regroupés dans un *vertex buffer*, un espace mémoire contigu stocké dans la mémoire principale ou la mémoire graphique, dépendamment des besoins d'accès[2] spécifiés par l'usager ou parfois spécifique à la plateforme de développement.

[2] lecture et écriture

Avec la standardisation des *shaders* dans les cartes d'accélération graphiques, le *skinning* dans les jeux vidéo est maintenant couramment implémenté à l'aide de *vertex shaders*. Tout d'abord, les matrices du squelette de la pose courante de l'objet animé sont regroupées dans une liste, appelée la "palette de matrices". Ensuite, la palette de matrices est transmise à la carte graphique par le biais des paramètres des *shaders*. Finalement, le contenu du *vertex buffer* de l'objet est transmis pour le rendu. Nous supposons ici que les indices des matrices associées sont transmises au *vertex shader* en même temps que la position de chaque sommet, par le biais du *vertex buffer*.

Étant donnée la simplicité relative de faire des accès arbitraires à la mémoire et de réaliser des algorithmes numériques sur CPU comparativement au GPU, la plupart des systèmes d'animation basés sur la physique sont implémentés sur CPU. Soulignons toutefois que récemment, cette situation était sur le point de changer : *Aegia* [Age], avec son PPU (*Physics Processing Unit*) et *Havok* [Hav], supportant le calcul sur GPU pour les petits objets de type "débris", compétitionnent actuellement férocement sur le front du calcul parallèle sur matériel spécialisé. Comme nous nous préoccupons d'ajouter des détails simulés physiquement sur les objets animés et que notre simulation est faite sur CPU, le résultat du *skinning* doit parvenir à la simulation, ce qui fait que le *skinning* doit calculé sur CPU pour tous les objets pour lesquels des mouvements secondaires sont simulés.

### 2.5.2 Connectivité géométrique

Lorsque le maillage géométrique d'un objet est animé physiquement[3] et que seule l'évolution de la position des sommets du maillage nous intéresse, il est souvent raisonnable d'utiliser directement son complexe cellulaire comme représentation physique, où chaque sommet est interprété comme une particule physique attachée par des ressorts à ses sommets voisins.

Cependant, lors de la modélisation d'objets avec un processus appelé*normal-mapping*, l'artiste peut dupliquer certains sommets pour générer des arêtes dures (*hard edges*), permettant de représenter un changement discontinu des normales entre deux triangles adjacents, pour modéliser des crevasses par exemple. Simuler une telle topologie reviendrait à considérer que le matériau possède des fentes. Cependant, une discontinuité

[3]donc, sans utiliser le *skinning*

physique est rarement la signification des arêtes dures, car elles ne servent en général qu'à l'aspect visuel de l'objet.

Pour éviter de simuler les sommets superposés comme deux particules indépendantes, il peut être utile de maintenir différentes représentations de l'objet : la représentation physique pour la simulation et la vue graphique pour le rendu. Pour ce faire, il est nécessaire de générer la topologie physique de l'objet à simuler en considérant les sommets superposés comme une seule particule.

### 2.5.3 Couplage animation-simulation

Pour interagir avec l'environnement et les modèles animés, la simulation des objets physiques doit être couplée avec l'application afin de connaître la configuration de la scène. Pour l'animation de mouvements secondaires, comme l'indiquent O'Brien *et al.* [OZH00], ce couplage est soit unidirectionnel, bidirectionnel ou hybride.

Le couplage unidirectionnel, que nous utilisons dans nos travaux, fait en sorte que l'environnement et les objets animés provoquent des mouvements secondaires, mais ces derniers n'ont pas d'effets secondaires sur l'environnement et les objets. Dans le couplage bidirectionnel, la simulation des mouvements secondaires a un impact sur l'évolution de la scène. Par exemple, si un personnage saute dans une flaque d'eau, une simulation d'eau couplée unidirectionnellement calcule l'éclaboussement correspondant à la force de l'impact du personnage, sans affecter le mouvement de ce dernier, alors que si la simulation est couplée bidirectionnellement, la vitesse du personnage est affectée par sa collision avec l'eau et réciproquement, de manière à respecter les lois de conservations physiques. De son côté, le couplage hybride représente toute la gamme des méthodes *ad hoc*, où l'interaction entre les systèmes est approximée.

Compte tenu de la complexité et des contraintes artistiques[4] qu'impose le couplage bidirectionnel, la synthèse d'animation basée sur la physique dans les jeux vidéo se contente en général des types de couplages unidirectionnel ou hybride.

[4]Les décisions prisent par l'artiste sur l'évolution de la scène peuvent être affectées par des facteurs sur lesquels il n'a pas un contrôle absolu.

# Chapitre 3

# Schémas d'intégration

*L'histoire est un éternel recommencement.*

Thucydide

Les schémas d'intégration numérique servent à extrapoler l'état futur d'une fonction dont on connaît (ou dont on peut estimer) des dérivées dans le temps et les conditions initiales. En animation basée sur la physique, on tente de déterminer l'état dans lequel sera un système après un certain laps de temps, souvent appelé le "pas de temps" (*timestep*).

Dans les systèmes de particules, ensemble de masses ponctuelles organisé de façon quelconque, l'état observable de la particule $i$ à un temps $t$ est sa position $\mathbf{x}_i(t)$ et sa vitesse $\dot{\mathbf{x}}_i(t)$. En général, différentes contraintes et forces s'appliquent sur chaque particule, ce qui modifie sa vitesse. C'est la séquence comprenant la mise à jour de la vitesse à partir des forces et contraintes et la mise à jour de la position, que nous appelons le "schéma d'intégration".

Les applications interactives, comme les jeux vidéo, nécessitent une simulation rapide afin de pouvoir générer plusieurs images par seconde. Il n'est pas rare de voir des applications interactives dont la fréquence de rendu tend vers 60 Hz. Donc, si parmi les effets souhaités on retrouve des mécanismes physiques, ces derniers ne disposent que d'une fraction d'un soixantième de seconde pour faire leurs calculs, sans quoi la simulation provoquera une latence dans l'interactivité de l'application.

À l'opposé, le domaine de la physique computationnelle se préoccupe surtout d'ob-

tenir des résultats de grande précision, ce qui lui permet de (voire l'oblige à) consacrer plusieurs secondes, voire minutes ou heures, simplement pour extrapoler l'état du système quelques millièmes de seconde dans le futur.

Plus loin, nous parlons de la stabilité des schémas d'intégration. Il s'agit du problème où le schéma ne converge pas vers une solution unique ou diverge. En effet, les schémas ne se fiant qu'à l'état actuel, et ne s'assurant pas de la validité physique sur le "trajet" menant à l'état futur calculé, risquent de mettre le système dans une impasse numérique, où le système explose numériquement. Cette situation est prévalente dans les systèmes physiques dits "raides" (*stiff*), dans lequels les forces appliquées sont peu élastiques. Le problème est d'autant plus important à résoudre, car les matériaux que nous simulons dans nos travaux (vêtements, sangles, cheveux) sont justement raides.

Finalement, compte tenu du contrôle arbitraire que possède l'usager dans les jeux vidéo (les objets peuvent changer très subitement de position et d'orientation), la simulation doit être robuste, *i.e.* la qualité visuelle du phénomène simulé ne doit pas se dégrader sous l'influence des actions de l'usager.

## 3.1 Schémas explicites

Typiquement, la dynamique d'un système de particules est régie par l'équation du mouvement de Newton :

$$\ddot{\mathbf{x}}_i = \frac{\mathbf{F}_i}{m_i} \tag{3.1}$$

où $\ddot{\mathbf{x}}_i$ est l'accélération de la particule $i$, $\mathbf{F}_i$ est la somme des forces s'y appliquant, et $m_i$ est sa masse. Les forces sont soit internes (répulsion, attraction, torsion, frottement interne, etc.) ou externes (gravité, vent, etc.).

Un des schémas explicites les plus couramment utilisés pour résoudre cette équation différentielle du deuxième degré est appelé le schéma d'Euler explicite modifié. Sa formulation itérative est la suivante :

$$\dot{\mathbf{x}}_i^{n+1} = \dot{\mathbf{x}}_i^n + \frac{\Delta t}{m_i}\mathbf{F}_i^n \tag{3.2}$$

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \Delta t \dot{\mathbf{x}}_i^{n+1} \tag{3.3}$$

où l'exposant indique le temps, tel que $\mathbf{x}^n \equiv \mathbf{x}(n\Delta t)$, $n$ est un entier représentant le numéro du pas de temps et $\Delta t$ est le pas de temps. Notons que la version non-modifiée

du schéma d'Euler explicite serait celle dans laquelle la position au temps $n+1$ est mise à jour grâce avec la vitesse au temps $n$.

Pour bien comprendre le fonctionnement et les limitations de ce schéma, prenons l'exemple d'une seule particule de masse $m$ que nous voudrions contraindre à rester à l'origine. Pour ce faire, nous la relions à un ressort linéaire dont la longueur au repos est nulle, attaché à $\mathbf{x} = \mathbf{0}$, dont la force de rappel est exprimée ainsi :

$$\mathbf{F} = -k\mathbf{x} \tag{3.4}$$

où $k$ est appelée la constante de rappel ou constante de Hooke.

Le schéma d'Euler explicite (modifié) pour ce système s'écrit donc :

$$\dot{\mathbf{x}}^{n+1} = \dot{\mathbf{x}}^n - k\frac{\Delta t}{m}\mathbf{x}^n \tag{3.5}$$

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t\dot{\mathbf{x}}^{n+1}. \tag{3.6}$$

Ce schéma est très simple à calculer et à implémenter, mais il comporte un handicap considérable : avec $\Delta t \geq \sqrt{\frac{2m}{k}}$, le système est instable. Pour observer ce comportement, injectons $\Delta t = \sqrt{\frac{2m}{k}}$ dans les équations 3.5 et 3.6, avec comme conditions initiales $\dot{\mathbf{x}}^0 = \mathbf{0}$ et $\mathbf{x}^0 \neq \mathbf{0}$, et faisons une itération du schéma. Nous obtenons alors :

$$\begin{aligned} \dot{\mathbf{x}}^1 &= -\sqrt{\frac{2k}{m}}\mathbf{x}^0 \\ \mathbf{x}^1 &= -\mathbf{x}^0. \end{aligned}$$

Nous remarquons que non seulement la particule ne s'arrête pas à l'endroit où la force devait l'attirer, mais que pour tous les prochains pas de simulation, la position de la particule oscillera entre $-\mathbf{x}^0$ et $\mathbf{x}^0$. Si nous augmentons encore le pas de temps, nous verrons que l'itération du schéma fait osciller la particule avec une amplitude de plus en plus grande à chaque itération ; le schéma d'intégration crée de l'énergie ! Le point de non-retour, où une énergie artificielle commence à apparaître, est régi par la condition $\Delta t \geq \sqrt{\frac{2m}{k}}$, qui est souvent appelée la "condition d'instabilité" du schéma. Elle relie la raideur d'un système au pas de temps et à la masse de ses composantes, et elle existe pour tous les systèmes simulés explicitement.

Une force d'amortissement (*damping*) explicite est souvent ajoutée dans les systèmes de particules pour leur faire perdre de la vitesse (*i.e.* dissiper de l'énergie). Cependant, cette dernière n'est souvent pas suffisante lorsque le pas de temps est plus grand que

ce que la condition de stabilité ne permet. D'ailleurs, comme plusieurs phénomènes physiques sont peu amortis et très raides, leur ajouter une force d'amortissement supplémentaire ne fait qu'enlever du réalisme à la simulation.

## 3.2 Schémas implicites

Le but des schémas implicites est de permettre un pas de temps de grandeur arbitraire. Dans la section précédente, une des raisons qui font échouer le schéma explicite est que ce schéma ne tient pas compte du fait que des forces devraient s'appliquer pour freiner la particule durant le trajet la menant vers une position invalide, où l'énergie du système est artificiellement augmentée, l'empêchant d'atteindre la position en question.

Pour sa part, la formulation du schéma d'Euler implicite, pour la particule $i$, avec $n$ dénotant le temps, est la suivante :

$$\dot{\mathbf{x}}_i^{n+1} = \dot{\mathbf{x}}_i^n + \frac{\Delta t}{m_i}\mathbf{F}_i^{n+1} \tag{3.7}$$

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \Delta t \dot{\mathbf{x}}_i^{n+1}. \tag{3.8}$$

Il est intéressant de noter dans l'équation 3.7 que la nouvelle vitesse $\dot{\mathbf{x}}_i^{n+1}$ dépend de forces $\mathbf{F}_i^{n+1}$ mesurées dans le futur, à la position inconnue $\mathbf{x}_i^{n+1}$. C'est cette dépendance entre la position inconnue et les forces s'appliquant à cette position qui vaut au schéma son qualificatif "implicite". Pour chaque itération du schéma implicite, nous devons donc résoudre le système d'équations 3.7 et 3.8.

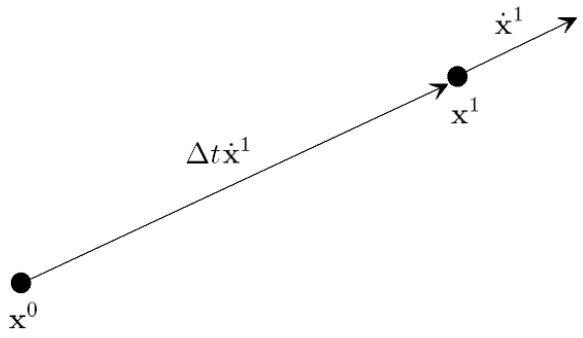



FIG. 3.1 – Une itération du schéma implicite.

La stabilité du schéma implicite est garantie par le fait que $\mathbf{x}_i^{n+1}$, la position finale

de la particule $i$, est choisie de telle sorte que les forces s'y appliquant génèrent une vitesse avec laquelle, si reculée d'un pas de temps, la particule retourne à sa position de départ (voir la figure 3.1).

Reprenons l'exemple de la particule de masse $m$ que nous voulions attirer vers l'origine (voir section 3.1) dont les conditions initiales sont $\dot{\mathbf{x}}^0 = \mathbf{0}$ et $\mathbf{x}^0 \neq \mathbf{0}$. Le système d'équations pour trouver $\dot{\mathbf{x}}^1$ et $\mathbf{x}^1$ est alors :

$$\begin{aligned} \dot{\mathbf{x}}^1 &= -\Delta t \frac{k}{m} \mathbf{x}^1 \\ \mathbf{x}^1 &= \mathbf{x}^0 + \Delta t \dot{\mathbf{x}}^1. \end{aligned}$$

La résolution du système mène à

$$\begin{aligned} \dot{\mathbf{x}}^1 &= -\frac{k\Delta t}{m + k(\Delta t)^2} \mathbf{x}^0 \\ \mathbf{x}^1 &= \frac{m}{m + k(\Delta t)^2} \mathbf{x}^0 \end{aligned}$$

et nous observons finalement cette propriété remarquable :

$$\begin{aligned} \lim_{\Delta t \to \infty} \dot{\mathbf{x}}^1 &= \mathbf{0} \\ \lim_{\Delta t \to \infty} \mathbf{x}^1 &= \mathbf{0}, \end{aligned}$$

signifiant que le pas de temps peut être de grandeur arbitraire sans que la solution numérique diverge et qu'il est théoriquement possible de satisfaire la contrainte en une seule itération. Cependant, en pratique, comme l'ont indiqué Baraff et Witkin [BW97], le pas de temps doit être petit (quelques millièmes à quelques centièmes de seconde) pour permettre une synchronisation avec les autres composantes du système interactif, comme la détection de collisions et l'animation par *keyframes*, qui avancent elles aussi avec un pas de temps qui leur est propre.

Pour bien comprendre ce qui donne une stabilité inconditionnelle aux schémas implicites, il faut savoir que l'instabilité des systèmes raides simulés explicitement est reliée à la limite de représentation des hautes fréquences du champ de forces sur un domaine (le temps) discrétisé. Or, comme Baraff et Witkin [BW98] ont souligné, les schémas implicites sont équivalents à filtrer les forces appliquées sur les particules, de façon à tenir compte de la variation spatiale de ces forces le long du trajet parcouru durant un pas de temps. Notons que ce filtrage se réduit à ajouter de la viscosité numérique,

amortissant critiquement[1] le système de particules et ressorts.

Les schémas d'intégration implicites ont été abondamment utilisés en calcul hors-ligne pour l'animation de tissus dans les films d'animation. Cependant, notons que pour un système de $p$ particules, la taille du système d'équations à résoudre à chaque itération du schéma d'Euler implicite est dans $O(p)$. Le temps de calcul et la quantité de mémoire nécessaires à ce schéma rendent donc impraticable son utilisation pour des systèmes de particules de grande taille dans les applications interactives sur plateformes contraintes (consoles de jeux vidéo, ordinateurs personnels, etc.).

## 3.3 Schémas semi-implicites

Les difficultés rencontrées lors de la simulation de systèmes raides ont poussé le développement d'alternatives qui tentent d'allier la rapidité des schémas explicites à la stabilité des schémas implicites.

Les forces raides sont souvent associées aux forces internes du matériau, comme les forces élastiques et le frottement interne. Desbrun *et al.* [DSB99] ont donc proposé un schéma prenant avantage de cette particularité en appliquant un pas implicite spécialement conçu pour les forces internes et un pas explicite pour les forces externes. Le fait de n'utiliser l'approche implicite que pour une partie des forces vaut à ce schéma l'appellation "semi-implicite".

Dans l'équation 3.7, le développement de premier ordre de la série de Taylor pour le terme $\mathbf{F}_i^{n+1}$ s'écrit :

$$\mathbf{F}_i^{n+1} = \mathbf{F}_i^n + \frac{\partial \mathbf{F}_i}{\partial \mathbf{x}} \Delta^{n+1} \mathbf{x}_i \tag{3.9}$$

où $\Delta^{n+1}\mathbf{x}_i = \mathbf{x}_i^{n+1} - \mathbf{x}_i^n$. Cette solution est exacte pour les ressorts linéaires dont la longueur au repos est nulle et elle est une approximation pour les ressorts de longueur au repos non-nulle.

Avec un système de $p$ particules, soit l'état généralisé, $\boldsymbol{\psi}$, vecteur à $3p$ dimensions représentant l'agglomération de l'état $\boldsymbol{\psi}_i$ de chaque particule $i$ du système, tel que $\boldsymbol{\psi} \in \{\mathbf{x}, \dot{\mathbf{x}}, \mathbf{F}\}$. Alors, si nous utilisons l'équation 3.9 avec les équations 3.7 et 3.8 afin

[1] En physique, un système oscillateur harmonique est amorti critiquement lorsque l'énergie qu'il dissipe est tout juste suffisante pour prévenir toute oscillation, le laissant atteindre sa configuration de repos sans la dépasser.

de résoudre pour la vitesse généralisée $\dot{\mathbf{x}}^{n+1}$ (pour les positions, il suffit d'injecter cette solution dans l'équation 3.8), dans un système dont toutes les particules possèdent la même la masse $m$, nous obtenons :

$$\dot{\mathbf{x}}^{n+1} = \left(\mathbf{I} - \frac{\Delta t^2}{m}\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right)^{-1}\left(\frac{\Delta t}{m}\mathbf{F}^n + \dot{\mathbf{x}}^n\right) \tag{3.10}$$

dont l'élément placé à la ligne $i$ et à la colonne $j$ de la matrice $-\frac{\partial \mathbf{F}}{\partial \mathbf{x}}$ s'exprime de la façon suivante :

$$\left(-\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right)_{ij} = \begin{cases} k_{ij} & : \text{ un ressort de constante } k_{ij} \text{ relie les particules } i \text{ et } j \\ 0 & : \text{ sinon.} \end{cases} \tag{3.11}$$

Desbrun *et al.* [DSB99] appellent la matrice $-\frac{\partial \mathbf{F}}{\partial \mathbf{x}}$ l'Hessienne du système. Soulignons qu'en physique, $\mathbf{F}$ est déjà l'inverse additive du gradient du champ scalaire représentant l'énergie élastique du système. Donc, $-\frac{\partial \mathbf{F}}{\partial \mathbf{x}}$, la matrice Jacobienne de $-\mathbf{F}$, est la matrice Hessienne de l'énergie du système.

Remarquons que le terme $\left(\mathbf{I} - \frac{\Delta t^2}{m}\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right)^{-1}$ est constant durant la simulation et peut donc être calculé une seule fois, avant le démarrage de la simulation. Par conséquent, le schéma semi-implicite fait en sorte qu'il ne soit pas nécessaire de résoudre un système d'équations à chaque itération.

Étant donné que ce schéma est une solution approximative (comme indiqué plus tôt, l'équation 3.9 n'est en général qu'une approximation) du pas d'Euler implicite pour les forces qui agissent comme des ressorts (les forces raides), sa stabilité est proche de celle d'une méthode totalement implicite. Cependant, la formulation matricielle étant généralement plus lourde en calcul que les schémas explicites et étant donné que tout changement de constantes de rappel nécessite à nouveau l'inversion de la matrice, ce schéma est généralement évité dans les jeux vidéo.

## 3.4 Schémas par contraintes géométriques

Jusqu'à présent, nous avons considéré que le processus de simulation contient une étape où des champs de forces sont échantillonnés dans l'espace à un instant donné afin d'en déduire l'accélération des particules. Or, nous avons souligné que la faiblesse de l'approche explicite réside dans la possible invalidité de l'état futur des champs de force.

Il existe une famille de schéma d'intégration se basant sur des contraintes géométriques, que nous décrivons plus loin, où chaque contrainte est représentée comme un

objectif géométrique pour chaque particule du système plutôt que sous la forme d'une force. Par construction, cette simple modification garantit l'état futur du système, car un état géométrique connu *a priori* est utilisé comme objectif lors de l'itération courante du schéma. Il est alors possible d'éviter de dépasser (*overshoot*) cet objectif.

Nous présentons ici l'un de ces schémas, proposé par Müller *et al.* [MHTG05].

Pour une particule $i$, dont la position au temps $n$ est $\mathbf{x}_i^n$ et l'objectif géométrique est $\mathbf{g}_i^n$, nous pouvons construire le schéma d'intégration suivant :

$$\dot{\mathbf{x}}_i^{n+1} = \dot{\mathbf{x}}_i^n + k\frac{\mathbf{g}_i^n - \mathbf{x}_i^n}{\Delta t} + \frac{\Delta t}{m_i}\mathbf{F}_i^n \tag{3.12}$$

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \Delta t\dot{\mathbf{x}}_i^{n+1} \tag{3.13}$$

où $k \in [0, 1]$ est le facteur de raideur (*stiffness*) de la contrainte et $\mathbf{F}_i^n$ est la somme des forces externes agissant sur la particule.

Notons qu'une contrainte de constante de raideur $k$ s'appliquera dans une proportion $k$ en un pas de temps. Il peut être désirable de calibrer le délai d'atteinte d'un niveau d'application d'une contrainte en fonction du temps plutôt que du nombre de pas de simulation. Cette calibration se fait alors en exprimant $k$ en fonction de la grandeur du pas de temps et de la masse des particules sur lesquelles s'applique la contrainte associée.

Revenons à l'exemple utilisé dans les sections précédentes, d'une particule attirée vers l'origine, dont les conditions initiales sont $\dot{\mathbf{x}}^0 = \mathbf{0}$ et $\mathbf{x}^0 \neq \mathbf{0}$. Si $k = 1$, rendant le système le plus raide possible, le premier pas de l'itération du schéma donnera :

$$\begin{aligned}\dot{\mathbf{x}}_i^1 &= -\frac{\mathbf{x}_i^0}{\Delta t} \\ \mathbf{x}_i^1 &= \mathbf{x}_i^0 + \Delta t\dot{\mathbf{x}}_i^1.\end{aligned}$$

Notons que, comme avec la méthode explicite :

$$\begin{aligned}\lim_{\Delta t\to\infty} \dot{\mathbf{x}}^1 &= \mathbf{0} \\ \lim_{\Delta t\to\infty} \mathbf{x}^1 &= \mathbf{0},\end{aligned}$$

ce qui signifie que ce schéma est stable quelque soit le pas de temps.

Cette nouvelle famille de schémas, très prometteuse pour les systèmes informatiques contraints et les applications interactives, est inconditionnellement stable et est potentiellement aussi rapide que les schémas explicites. Notons toutefois que sa difficulté principale est dans la traduction des forces agissant sur un système en objectifs géométriques.

Comme nous le verrons dans le chapire 5, notre modèle de simulation, à l'instar de celui de Müller *et al.* [MHTG05], arrive toutefois à surmonter cette difficulté pour certains matériaux élastiques.

# Chapitre 4

# Travaux antérieurs

*Conservons par la sagesse ce que nous avons acquis par l'enthousiasme.*

Condorcet (Marie Jean Antoine Nicolas de Caritat)

Depuis le début des années 1990, la recherche sur la synthèse d'animation basée sur la physique a produit d'excellents résultats concernant la simulation de surfaces élastiques et d'objets mous. Cette section donne une vue d'ensemble sur certaines avancées importantes et sur des résultats sur lesquels nos travaux se basent.

## 4.1 Surfaces élastiques

Comme la peau, les vêtements et les accessoires flexibles sont omniprésents lorsqu'il est question de personnages animés, la simulation de ces objets constitue un pas important à franchir dans la quête du réalisme des mouvements secondaires.

### 4.1.1 Modèles géométriques

Pour pouvoir être simulé, un matériau doit posséder une description théorique bien définie et cette dernière doit être suffisamment proche de la physique réelle pour produire des résultats convaincants.

House et Breen [HB00] soulignent qu'un tissu n'est pas un matériau continu. *A contrario*, les éléments internes procurant son comportement au tissu, *e.g.* le tricot ou le tissage de fils, sont macroscopiques. De plus, comme l'illustre la figure 4.1, l'arrangement

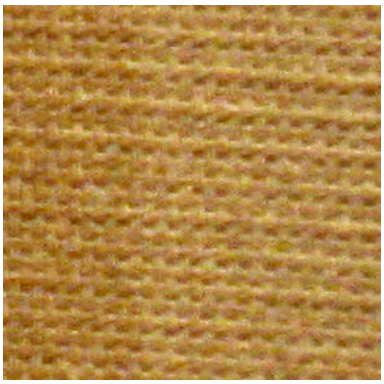


FIG. 4.1 – Gros plan sur un morceau de tissu de lin réel.

géométrique de ces matériaux internes est très ordonné, prenant souvent la forme d'un quadrillage uniforme. Il est donc naturel de modéliser un tissu comme une grille de particules reliées entre elles par des forces élastiques, tel qu'illustré à la figure 4.2 (b).

L'utilisation de systèmes de particules pour la simulation de matériaux flexibles quelconques est cependant aussi justifiée entre autres par Desbruns *et al.* [DSB99], Grinspun *et al.* [GHDS03], Meyer *et al.* [MDDB01] et Müller *et al.* [MHTG05]. En effet, comme les pas de temps sont une discrétisation du domaine temporel, les particules en tant que représentation d'un matériau sont une discrétisation de la matière sur le domaine spatial. Dans la mesure où les forces entre les particules sont modélisées de façon réaliste[1] du point de vue physique et que le nombre de particules est suffisamment grand, le matériau simulé peut suivre un comportement plausible.

Même si les surfaces élastiques sont souvent décrites en termes de forces entre les particules, notons qu'il existe d'autres approches équivalentes pour formuler la dynamique interne du système. Baraff et Witkin [BW98], par exemple, mesurent l'énergie d'étirement et le cisaillement pour chaque triangle et le pliage pour chaque arête partagée par deux triangles. Dans le cas où ce sont des énergies et non des forces qui sont mesurées, plutôt qu'exposée par l'équation 3.1, l'équation différentielle du mouvement à résoudre devient :

$$\ddot{\mathbf{x}}_i = m_i^{-1} \left( -\frac{\partial E}{\partial \mathbf{x}_i} + \mathbf{F}_i \right) \tag{4.1}$$

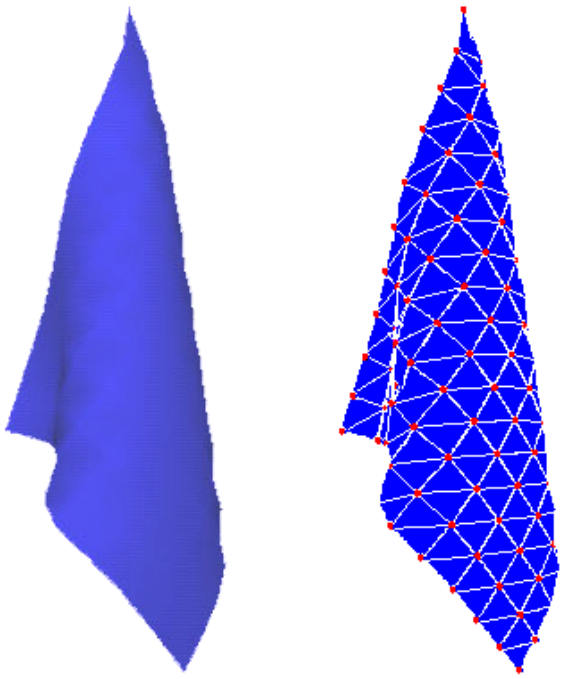
où $\ddot{\mathbf{x}}_i$ est l'accélération d'une particule $i$ de masse $m_i$, $\mathbf{F}_i$ est la somme des forces externes, et $E$, fonction de la position des particules influençant la particule $i$, représente

[1] qui s'approche des modèles théoriques acceptés

l'énergie due aux contraintes internes de la particule. Notons que toute force physique agissant sur une particule peut être exprimée comme l'inverse additif du gradient d'une fonction scalaire d'énergie, impliquant que les approches à base de forces et d'énergie sont équivalentes. Quelle que soit la formulation envisagée, le réalisme des résultats dépend surtout de la qualité du modèle physique sous-jacent qui est employé pour décrire la dynamique du système de particules..

### 4.1.2 Forces élastiques

La peau, les vêtements, le cuir et la corde sont des exemples de matériaux ayant tendance à résister plus ou moins fortement à la modification de leur forme et à retourner à leur configuration de repos initiale. Cette propriété d'un matériau est ce qu'on appelle son "élasticité". Ce sont les forces élastiques qui permettent à un tissu de maintenir des plis lorsque drapé sur un objet ou suspendu (voir la figure 4.2 (a)).



(a) (b)

FIG. 4.2 – (a) Tissu suspendu par le coin supérieur ; (b) Particules simulées (sphères) et triangles du maillage (fils de fer en blanc).

En se basant sur le système de mesures de l'industrie du textile de Kawabata [Kaw80], Breen et House [BH92] ont introduit la simulation de tissus par systèmes

de particules soumises à trois types de forces provoquées par le stress dans les tissus : l'étirement, le cisaillement et le pliage. Comme cette catégorisation des forces correspond aux trois modes de mouvement interne d'une surface élastique, elle fait généralement consensus et est souvent utilisée pour la simulation d'autres types de matériaux minces, comme l'ont fait Grinspun *et al.* [GHDS03] pour la simulation de feuilles de papier ou de canettes d'aluminium.

### Étirement

La force d'étirement est une force de rappel tentant de conserver à distance constante les paires de particules immédiatement voisines. Souvent, les tissus sont construits selon un maillage géométrique quadrillé orthogonalement. Dans le contexte de la force d'étirement, deux particules sont alors immédiatement voisines si elles sont adjacentes sur un des axes orthogonaux du tissu.

La force d'étirement agit comme un ressort radial, le long de l'axe reliant les deux particules impliquées. Pour la force $\mathbf{F}_{ij}$ appliquée sur la particule $i$, issue de l'étirement entre les particules voisines $i$ et $j$, Desbrun *et al.* [DSB99] proposent cette expression :

$$\mathbf{F}_{ij} = -k\left(\|\mathbf{x}_j - \mathbf{x}_i\| - l_0\right)\frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|} \tag{4.2}$$

où $k$ est la constante de Hooke du ressort et $l_0$ est sa longueur au repos. Notons que ce type de ressort est aussi appelé "ressort structurel". Voir la figure 4.3 pour une illustration des types de ressorts.

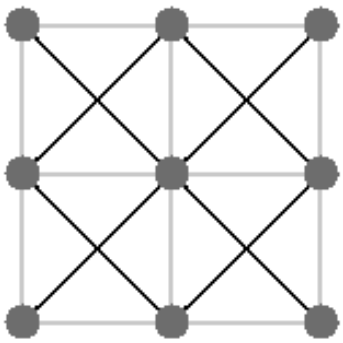


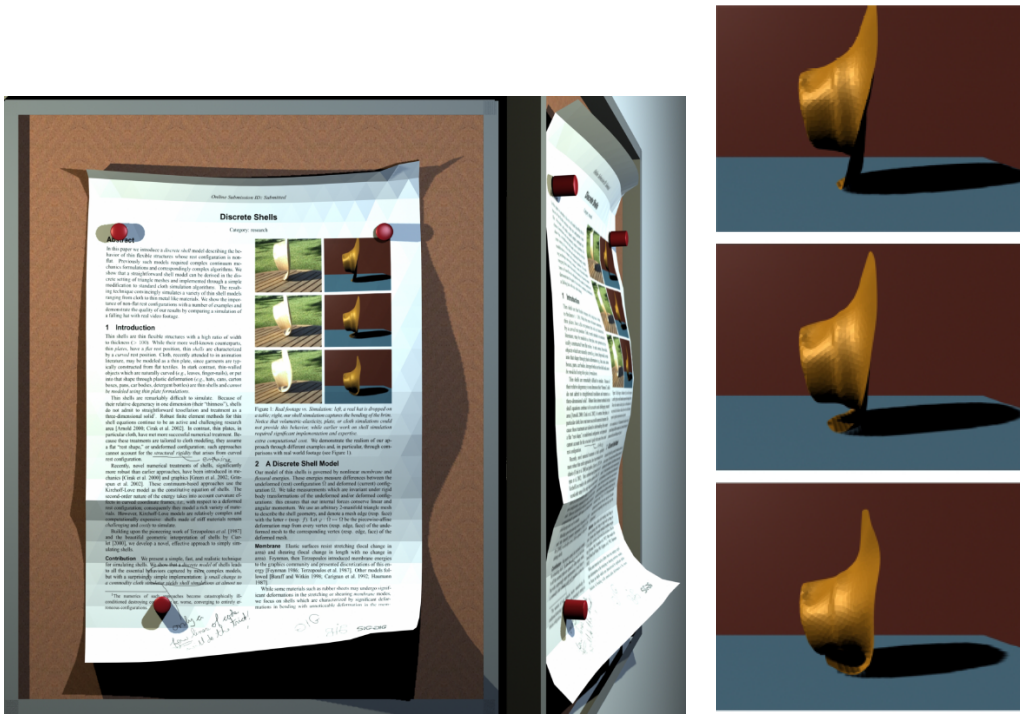
FIG. 4.3 – Ressorts structurels (lignes grises) et de cisaillement (lignes noires) attachés à des particules (disques).

### Cisaillement

La force de cisaillement agit contre les variations de l'aire d'une surface élastique. L'une des méthodes utilisées pour introduire ce type de force dans la simulation de tissus, proposée par Provot [Pro95], est d'ajouter, sur une grille rectangulaire de masses et de ressorts structurels, un ressort entre chaque paire de particules immédiatement voisines diagonalement sur la grille (voir la figure 4.3). La formulation d'une telle force est la même que celle de l'équation 4.2.

### Pliage

La force de pliage (*bending*) résiste à la variation de courbure de la surface élastique simulée. Elle détermine la difficulté à plier le matériau et affecte le rayon de courbure des plis lorsqu'un tissu est froissé ou est drapé sur un objet.




(a) (b)

FIG. 4.4 – (a) Feuille flexible accrochée ; (b) chapeau entrant en collision avec le sol. Images tirées de l'article de Grinspun *et al.* [GHDS03].

Grinspun *et al.* [GHDS03] ont soulevé l'importance d'un bon modèle de pliage pour une vaste gamme d'objets utilisés dans les environnements virtuels, comme les feuilles

de papier (figure 4.4 (a)), et ont montré comment cette force affecte le réalisme d'objets flexibles entrant en collision avec le sol (voir le chapeau à la figure 4.4 (b)). Nous ne décrivons pas leur formulation ici, car elle ressemble à celle de Bridson *et al.* [BMF03], que nous expliquons dans le prochain paragraphe.

L'une des préoccupations dans l'ajout d'une nouvelle force à un système de simulation est le risque de ne plus conserver la quantité de mouvement ou le moment cinétique. En physique, ces deux quantités sont régies par ce qui est appelé le "principe de conservation", qui stipule que pour un système fermé quelconque, elles demeurent constantes. Étant donné que la quantité de mouvement et le moment cinétique d'un système déterminent de façon unique son énergie cinétique, et qu'il est connu que l'énergie totale d'un système fermé est constante, il s'ensuit que la quantité de mouvement et le moment cinétique sont conservés. De plus, afin de pouvoir contrôler l'intensité relative des trois types de forces, il est certainement préférable d'avoir une force de pliage qui agit sur un mode de mouvement interne indépendant de l'étirement et du cisaillement.

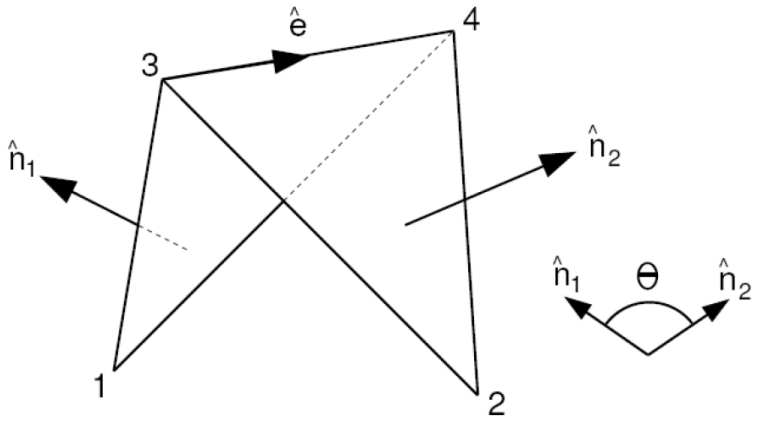



FIG. 4.5 – Élément de pliage de l'arête $\overline{34}$ d'angle diédral $(\pi - \theta)$. Figure tirée de l'article de Bridson *et al.* [BMF03].

Bridson *et al.* [BMF03] ont proposé une force de pliage respectant le principe de conservation et agissant sur un degré de liberté indépendant des autres types de forces. Cette force est calculée pour chaque arête partagée par deux triangles et la figure 4.5 illustre la configuration de quatre particules numérotées impliquées dans le pliage d'une arête.

Soit $\mathbf{x}_i$ la position de la particule numérotée $i$, et soient $\mathbf{n}_1 = (\mathbf{x}_1 - \mathbf{x}_3) \times (\mathbf{x}_1 - \mathbf{x}_4)$, $\mathbf{n}_2 = (\mathbf{x}_2 - \mathbf{x}_4) \times (\mathbf{x}_2 - \mathbf{x}_3)$ et $\mathbf{e} = \mathbf{x}_4 - \mathbf{x}_3$. La position permise $\mathbf{u}_i$ pour chaque particule $i$ dans le mode de mouvement du pliage est :

$$\begin{aligned}
\mathbf{u}_1 &= \|\mathbf{e}\| \frac{\mathbf{n}_1}{\|\mathbf{n}_1\|^2} \\
\mathbf{u}_2 &= \|\mathbf{e}\| \frac{\mathbf{n}_2}{\|\mathbf{n}_2\|^2} \\
\mathbf{u}_3 &= \frac{(\mathbf{x}_1 - \mathbf{x}_4) \cdot \mathbf{e}}{\|\mathbf{e}\|} \frac{\mathbf{n}_1}{\|\mathbf{n}_1\|^2} + \frac{(\mathbf{x}_2 - \mathbf{x}_4) \cdot \mathbf{e}}{\|\mathbf{e}\|} \frac{\mathbf{n}_2}{\|\mathbf{n}_2\|^2} \\
\mathbf{u}_4 &= \frac{(\mathbf{x}_1 - \mathbf{x}_3) \cdot \mathbf{e}}{\|\mathbf{e}\|} \frac{\mathbf{n}_1}{\|\mathbf{n}_1\|^2} - \frac{(\mathbf{x}_2 - \mathbf{x}_3) \cdot \mathbf{e}}{\|\mathbf{e}\|} \frac{\mathbf{n}_2}{\|\mathbf{n}_2\|^2}
\end{aligned}$$

et la force de pliage $\mathbf{F}_i$ agissant sur la particule $i$ proposée par Bridson *et al.* est alors :

$$\mathbf{F}_i = k \frac{\|\mathbf{e}\|^2}{\|\mathbf{n}_1\| + \|\mathbf{n}_2\|} \sin(\theta/2) \mathbf{u}_i \tag{4.3}$$

avec $k$ la constante de raideur de pliage, pour $i \in \{1, 2, 3, 4\}$.

### 4.1.3 Frottement

La force tenant un foulard autour d'un cou est le frottement. C'est aussi elle qui fait qu'une voiture adhère sur la route et qui nous permet de soulever un verre sans qu'il ne nous glisse entre les doigts. Compte tenu que le frottement affecte grandement le comportement d'un tissu interagissant avec son environnement, il augmente le réalisme de la simulation lorsque correctement modélisé.

#### Frottement dynamique et statique

La force de frottement apparaît lorsque deux surfaces en contact accélèrent l'une par rapport à l'autre. Le frottement existe entre autre parce qu'aucune surface n'est parfaitement lisse ; même les surfaces les plus lisses possèdent de minuscules aspérités. Deux surfaces en contact et au repos relatif s'emboîtent donc légèrement l'une dans l'autre ; il faut dépenser une énergie supérieure à l'énergie potentielle de cet "emboîtement", proportionnelle à la force normale entre les surfaces, pour accélérer l'une des surfaces par rapport à l'autre. Cette force-seuil est le frottement "statique". Une autre forme de frottement entre les surfaces, appelé frottement "cinétique", se manifeste lorsque les deux surfaces en contact possèdent une vitesse relative. Il y a alors collisions entre les aspérités des deux surfaces, dissipant de l'énergie sous forme de chaleur, décélérant les

surfaces dans leur référentiel relatif, et agissant sur le système sous la forme d'une force opposée au mouvement, proportionnelle à la vitesse relative entre les deux surfaces et à la force normale entre les surfaces.

Pour simuler le frottement statique, on ajoute parfois un ressort temporaire sur les particules lentes par rapport à la surface sur laquelle elles sont en contact, dont l'effet est retiré lorsqu'une certaine tension est atteinte. L'une des façons employées pour ajouter du frottement cinétique à une particule est de lui appliquer une impulsion appropriée, modifiant directement la partie tangentielle de sa vitesse $\dot{\mathbf{x}}$. Bridson *et al.* [BFA02] ont proposé la formulation suivante :

$$\dot{\mathbf{x}} = \max\left(1 - \mu\frac{\Delta\dot{\mathbf{x}}_N}{\|\dot{\mathbf{x}}_T^{pre}\|}, 0\right)\dot{\mathbf{x}}_T^{pre} + \dot{\mathbf{x}}_N^{pre} \tag{4.4}$$

où $\mu$ est la constante de frottement de la surface impliquée, l'indication "$pre$" en exposant signifie la valeur prise au pas précédent, $\dot{\mathbf{x}}_N$ est la composante normale à la surface de la vitesse $\dot{\mathbf{x}}$ de la particule, $\dot{\mathbf{x}}_T$ est la composante tangentielle à la surface de cette vitesse et $\Delta\dot{\mathbf{x}}_N \equiv \dot{\mathbf{x}}_N - \dot{\mathbf{x}}_N^{pre}$. Notons que par définition, $\dot{\mathbf{x}} \equiv \dot{\mathbf{x}}_T + \dot{\mathbf{x}}_N$.

Dans l'équation 4.4, la fonction "max" est utilisée pour ne tenir compte de la friction que pour les cas où la particule touche à la surface durant le pas de simulation, ce qui n'arrive que lorsque sa vitesse normale à la surface pointe vers la surface, et exclure les cas où elle s'en éloigne. Aussi, l'utilisation du terme $\Delta\dot{\mathbf{x}}_N$ est justifiée par le fait que la friction est proportionnelle à la force normale $F_N = m\frac{\Delta\dot{\mathbf{x}}_N}{\Delta t}$ et que l'impulsion à appliquer, pour conserver la validité géométrique après le pas de temps, est justement égale à $\mu F_N \Delta t$, où $\Delta t$ est le pas de temps.

### Frottement interne

Une autre manifestation du frottement est celui qualifié "d'interne". Le frottement interne est la force qui fait en sorte que tout mécanisme physique isolé[2] perd de l'énergie sous forme de chaleur. Lorsqu'un tissu est déformé, cette énergie est perdue par les fibres glissant les unes sur les autres. Dans la simulation, ce frottement interne peut être représenté par une force radiale couplée à chaque ressort et proportionnelle à la projection de la vitesse relative entre les deux particules sur le vecteur radial les reliant. Ce type de force est aussi appelé "viscosité". Voici une équation courante, utilisée par

[2] sans apport énergétique extérieur

Baraff et Witkin [BW97], décrivant le frottement interne agissant sur la particule $i$ :

$$\mathbf{F}_i = -k_v \frac{\Delta\dot{\mathbf{x}}_{ij} \cdot \Delta\mathbf{x}_{ij}}{\|\Delta\mathbf{x}_{ij}\|^2} \Delta\mathbf{x}_{ij} \tag{4.5}$$

où $k_v$ est la constante de viscosité, $\Delta\dot{\mathbf{x}}_{ij}$ est la vitesse relative des particules $i$ et $j$, et $\Delta\mathbf{x}_{ij}$ est leur position relative.

#### Viscosité externe

Il est possible de simuler la viscosité du milieu dans lequel se déplacent les particules à l'aide d'une force de frottement externe. Cette force $\mathbf{F}_i$, s'appliquant sur la particule $i$, peut s'exprimer simplement de cette façon :

$$\mathbf{F}_i = -k_v \dot{\mathbf{x}}_i \tag{4.6}$$

où $k_v$ est la constante de viscosité et $\dot{\mathbf{x}}_i$ est la vitesse de la particule $i$.

## 4.2 Objets mous

Nous discutons dans cette section des travaux antérieurs importants traitant de la simulation d'objets mous.

### 4.2.1 Déformation de l'espace

Sederberg et Parry [SP86] ont réalisé un travail pionnier en modélisation d'objets déformables lorsqu'ils ont proposé de décrire les déformations d'un objet en terme de la déformation de son espace environnant, ce qu'ils appellent les *free-form deformations*.

Terzopoulos *et al.* [TPBF87] ont appliqué cette idéalisation à la dynamique des objets élastiques, formulant l'énergie interne d'un objet déformé comme une fonction de la variation entre son état déformé et son état au repos. De plus, Faloutsos *et al.* [FvdPT97] ont étendu l'approche à la synthèse d'animation en général, où la dynamique élastique du matériau ainsi que l'interaction avec son environnement (collisions et contraintes physiques) sont décrites en terme de l'énergie impliquée dans la déformation selon la méthode des *free-form deformations*.

### 4.2.2 Éléments finis

Malgré la popularité du modèle particulaire, notons qu'une autre méthode existe pour la simulation d'objet élastiques : la méthode des éléments finis [SF89]. Cette méthode sert souvent, en ingénierie, à résoudre des problèmes aux conditions limites. Le matériau est subdivisé en sous-régions continues, les éléments finis, qui sont attachées les unes aux autres. Les conditions d'équilibre et de continuité du matériau sont alors posées pour chaque élément, et un système d'équations simultanées est résolu pour les différentes inconnues du système, par exemple les forces sur le système. Il est alors possible d'en déduire l'évolution du maillage d'éléments, qui peut aussi bien représenter le volume que la surface de l'objet simulé en fonction du temps.

Notons que les contraintes des consoles de jeu, où la simulation basée sur la physique partage les ressources du système avec plusieurs autres activités computationnelles, rendent la méthode des éléments finis peu praticable. En effet, les systèmes d'équations nécessitent souvent des techniques de résolution numériques itératives, relativement lentes et gourmandes côté mémoire.

Néanmoins, Capell *et al.* [CGC+02a, CGC+02b] ont su développer un formalisme multi-résolution pour permettre à la simulation d'être précise là où les déformations ou les détails sont importants, simulant les modèles proposés avec des performances interactives lorsque toutes les ressources du système y sont consacrées.

### 4.2.3 Nuage de particules

Müller *et al.* [MHTG05] proposent une méthode pour simuler des objets mous ne dépendant pas de la connectivité du maillage géométrique les représentant. Comme dans plusieurs autres méthodes, les sommets du maillage sont interprétés comme des particules, mais toutes les forces internes sont exprimées en terme de la distance de chaque particule à sa position de repos dans l'objet original, non-déformé.

Ils utilisent le schéma d'intégration par contraintes géométriques dont nous avons discuté dans la section 3.4, ce qui leur confère une grande rapidité et une stabilité garantie.

À chaque pas de simulation, afin de générer l'objectif géométrique des particules, leur technique fait une correspondance de formes (*shape matching*) entre le nuage de

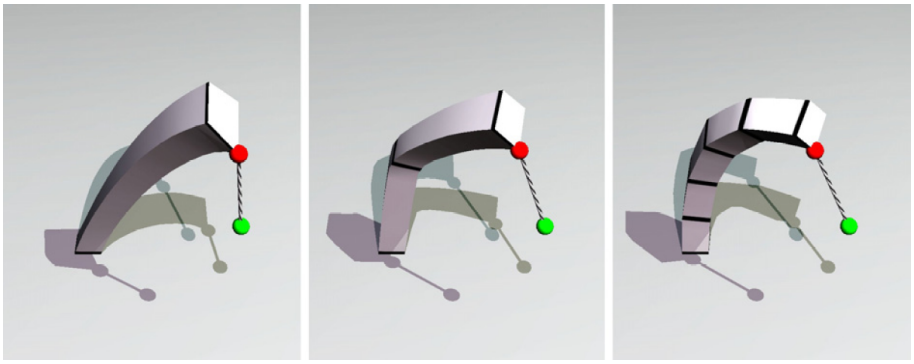


FIG. 4.6 – Interaction avec une tige flexible subdivisée respectivement en un, trois et cinq groupes de particules. Un usager tire sur le bout de la tige afin de la faire plier. Cette image provient de l'article de Müller *et al.* [MHTG05].

particules animées et l'état de repos de l'objet. Elle trouve donc l'orientation et la translation minimisant la somme des carrés des distances euclidiennes entre la position de chaque particule déplacée et de sa position au repos.

Pour permettre des effets globaux de pliage, les objets sont subdivisés en groupes de particules se chevauchant, où chaque groupe représente un morceau de matériau continu. Comme la correspondance de formes est faite pour chaque groupe de particules de façon indépendante, le chevauchement des groupes maintient la continuité de l'objet. La figure 4.6 illustre l'interaction d'un usager avec un objet mou ainsi subdivisé.

## 4.3 Détection de collisions pour surfaces élastiques

La détection de collisions est une phase importante de la simulation. Lorsque des objets entrent en collision, une étape appelée la "réponse aux collisions" se charge de donner une réaction adéquate aux collisions détectées, afin de préserver la validité géométrique. En plus d'être importantes, les détections et réponses aux collisions peuvent s'avérer extrêmement coûteuses car, dans une scène de $p$ polygones, en utilisant l'approche par force brute, il peut y avoir potentiellement $O(p^2)$ collisions à tester dans une seule itération de la phase de détection, *i.e.* entre chaque paire de polygones. Notons que d'autres itérations du processus de détection et réponse peuvent être nécessaires pour tenir compte des nouvelles collisions engendrées par la réponse à celles détectées précédemment.

Dans les jeux vidéo, les algorithmes par force brute sont généralement à proscrire car le temps utilisé par ceux-ci est souvent trop grand pour atteindre un niveau de performance interactif dans des scènes animées non-triviales.

Dans la section 4.3.1, nous présentons certaines structures accélératrices courantes qui ont été éprouvées spécialement dans la détection de collisions pour les objets déformables. Ensuite, dans la section 4.3.2, nous discutons des approches possibles une fois que des primitives géométriques susceptibles d'entrer en collision sont identifiées.

### 4.3.1 Structures hiérarchiques et subdivision

Remarquons que ce qui ralentit considérablement les algorithmes par force brute est le fait qu'ils exécutent des tests de collisions entre des primitives se trouvant trop éloignées les unes des autres pour être entrées en collision durant le pas de simulation courant. Ainsi, dans les stratégies servant à accélérer la détection de collisions, l'effort principal est mis à éviter ces tests inutiles.

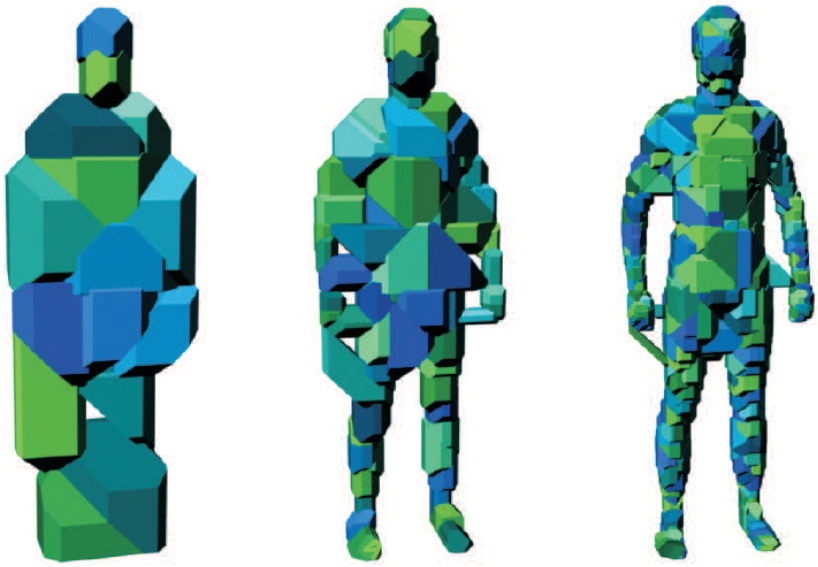


FIG. 4.7 – Représentation de trois niveaux d'une hiérarchie de volumes englobant. Figure tirée de l'article de Teschner *et al.* [TKH+05].

Les hiérarchies de volumes englobants (*bounding volume hierarchies*) servent justement à cette fin. Comme l'expliquent Teschner *et al.* [TKH+05], ces hiérarchies sont

parmi les structures les plus efficaces pour réduire le temps de détection de collisions. Elles ont été utilisées abondamment dans le cas où tous les objets animés sont rigides, mais servent de plus en plus pour les objets déformables.

Une telle hiérarchie est sous forme d'arborescence. Chaque niveau hiérarchique est un volume englobant dont les enfants sont soit d'autres volumes englobants, soit des primitives géométriques, les feuilles de la hiérarchie.

La construction de l'arborescence se fait en général en partant des enfants vers les parents, en commençant par les primitives géométriques, qui sont individuellement englobées d'un volume. Par la suite les volumes voisins sont regroupés selon différents critères géométriques, qui peuvent dépendre de la courbure locale de la surface, de la proximité géométrique etc., jusqu'au moment où il ne reste plus qu'un volume, la racine de la hiérarchie, englobant toute la géométrie de l'objet. Un algorithme de regroupement hiérarchique de faces (*hierarchical face clustering*) peut être utilisé pour obtenir l'arborescence à partir du maillage de l'objet.

Lorsque les objets englobés sont déformables, il est nécessaire, après chaque pas de temps de l'animation, mais avant la phase de détection des collisions, de mettre à jour la hiérarchie de volumes englobants en entier. Pour les objets simulés ne subissant pas de cassures, la hiérarchie possède une topologie constante et n'a à être calculée qu'une seule fois, et seulement la taille et la position des volumes englobants sont à mettre à jour lors d'un pas de la simulation. En remarquant que le nombre de noeuds d'une telle hiérarchie est dans $O(p)$, avec $p$ le nombre de polygones de la scène, le temps de mise à jour de la hiérarchie est aussi dans $O(p)$.

La figure 4.7 illustre trois niveaux de la hiérarchie de volumes englobants d'un personnage virtuel. Pour la détection de collisions proprement dite entre deux hiérarchies, nous référons le lecteur à l'algorithme 4.1, décrivant le parcours des hiérarchies de deux objets.

### 4.3.2 Validité géométrique

Un élément simulé est considéré "géométriquement valide" si son maillage géométrique ne contient pas d'auto-intersection et qu'il n'intersecte pas non plus la scène dans laquelle il se trouve.

ALGORITHME 4.1 : Détection de collisions hiérarchique

**Fonction :** Parcourir(a,b)

**Entrée :** Hiérarchie a,b ;

**Type de retour :** Intersection

```
1.  si a.racine.intersecte( b.racine ) = faux alors
2.      retourner nil ;
3.  si a.estFeuille() et b.estFeuille() alors
4.      Intersection résultat ← intersecterPrimitives( a.primitive(), b.primitive() ) ;
5.      si résultat ≠ nil alors
6.          retourner résultat ;
7.  sinon
8.      pour chaque paire ( a.enfant(i), b.enfant(j) )
9.          Intersection résultat ← Parcourir( a.enfant(i), b.enfant(j) ) ;
10.         si résultat ≠ nil alors
11.             retourner résultat ;
12. retourner nil ;
```

### Détection exacte

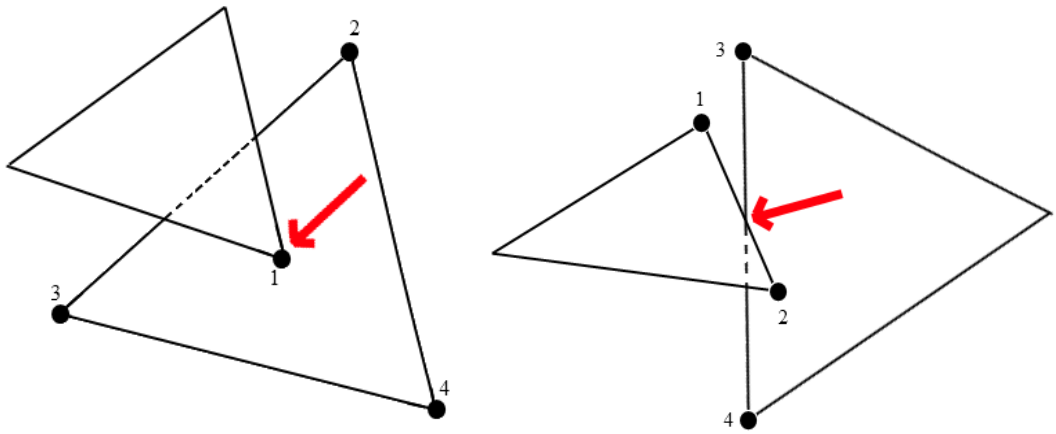
Si la configuration géométrique initiale des objets de la scène est géométriquement valide et que la résolution des collisions de chaque pas de simulation n'introduit pas d'intersection, il s'ensuit que la validité géométrique est conservée.

C'est cette approche que Bridson *et al.* [BFA02] préconisent. Leur phase de détection de collisions calcule le moment précis où chaque collision a eu lieu et peut ainsi appliquer une réponse exacte conservant la validité géométrique.

Comme dans la plupart des systèmes de simulation, la géométrie que Bridson *et al.* utilisent pour chaque tissu est représentée par un maillage triangulaire placé dans une hiérarchie de volumes englobants. Ainsi, à chaque pas de temps, une fois que la

hiérarchie est parcourue et qu'est trouvée une paire de triangles susceptibles d'être entrés en collision, un test de collision exact est effectué pour calculer le moment où la collision se serait produite.

Deux types de collisions peuvent se produire entre deux triangles : soit un sommet d'un des triangles frappe la surface de l'autre triangle, soit une paire d'arêtes appartenant chacune à un des triangles sont entrées en collision. Intentionnellement, les situations dégénérées, comme les collisions sommet-sommet et face-face, ne sont pas traitées. Dans la réalité, la probabilité que de telles collisions se produisent est pratiquement nulle, ce qui fait que décréter au hasard un endroit où la collision s'est d'abord produite donnera des résultats aussi plausibles visuellement. Concrètement, les situations dégénérées sont ainsi traitées comme l'un des deux types décrits précédemment. La figure 4.8 illustre les deux différents types de collisions. Il s'agit donc, pour deux triangles suspectés d'être entrés en collision, de faire le test de collision sur chaque paire sommet-triangle et chaque paire arête-arête. Quatre sommets, que nous numérotons $i \in \{1, 2, 3, 4\}$, sont impliqués dans les deux cas.




(a) (b)

FIG. 4.8 – Illustration de l'instant où a lieu chacun des deux types de collision possibles entre deux triangles. Pour faciliter l'interprétation, nous hachurons les arêtes cachées, nous numérotons les sommets définissant impliqués dans chaque collision, et nous ajoutons une flèche indiquant l'endroit où la collision a lieu. La figure (a) montre une collision entre le sommet d'un triangle et une face de l'autre triangle. La figure (b) montre la collision entre deux arêtes, chacune appartenant à un triangle différent.

Il est possible d'exprimer analytiquement l'instant de collision de deux triangles d'un maillage animé, où chaque sommet peut se mouvoir dans des directions indépendantes. Soient les positions respectives des sommets en question $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ ainsi que leurs vitesses $\{\dot{\mathbf{x}}_1, \dot{\mathbf{x}}_2, \dot{\mathbf{x}}_3, \dot{\mathbf{x}}_4\}$. Doghri *et al.* [DMT98] ont montré que calculer l'instant où la collision a lieu se réduit à résoudre pour $t$ l'équation cubique suivante :

$$((\mathbf{x}_{1,2} + t\dot{\mathbf{x}}_{1,2}) \times (\mathbf{x}_{1,3} + t\dot{\mathbf{x}}_{1,3})) \cdot (\mathbf{x}_{1,4} + t\dot{\mathbf{x}}_{1,4}) = 0 \tag{4.7}$$

où $\dot{\mathbf{x}}_{i,j} \equiv \dot{\mathbf{x}}_j - \dot{\mathbf{x}}_i$ et $\mathbf{x}_{i,j} \equiv \mathbf{x}_j - \mathbf{x}_i$. Il y a collision si $t$ est réel, positif et plus petit ou égal au pas de temps de la simulation. Notons que cette équation revient à chercher l'instant où les quatre sommets sont coplanaires.

### Fonction de distance

Une autre façon de détecter et de résoudre les collisions, proposée par Hoff *et al.* [ICK+99], est d'utiliser un champ scalaire de distance à la surface pour chaque objet rigide, servant à sortir les primitives simulées se retrouvant à l'intérieur d'autres objets. Dans la méthode de Hoff *et al.*, la fonction de distance est construite en espace image grâce au matériel graphique, permettant d'atteindre des performances interactives. Une des difficultés de cette méthode, comme avec la plupart des méthodes convertissant un maillage polygonal en une fonction de distance discrétisée est dans le choix de la finesse de la grille sur laquelle la fonction est évaluée. La finesse de la grille doit être suffisante pour représenter correctement tous les détails géométriques de l'objet, sans quoi la détection du passage des régions extérieures à intérieures ne se fera pas avec une précision adéquate, *i.e.* des objets passeront au travers les uns des autres.

Une autre façon d'implémenter la fonction de distance, est de générer, pour tous les objets statiques, une grille de voxels dans laquelle le point de l'objet le plus près du centre de chaque voxel est stocké, comme proposé par Meyer *et al.* [MDDB01].

En général, la fonction de distance utilisée est signée, ayant une valeur négative à l'intérieur et positive à l'extérieur de l'objet. Son gradient pointe alors à l'opposé de la normale à la surface. Pour résoudre la collision, la fonction de distance $\phi(\mathbf{x})$, dépendant de la position $\mathbf{x}$, permet de calculer simplement la nouvelle position $\mathbf{x}_i^{nouveau}$ à donner à une particule $i$ dont la position actuelle est $\mathbf{x}_i$, à l'aide de l'équation suivante :

$$\mathbf{x}_i^{nouveau} = \mathbf{x}_i - \phi(\mathbf{x}_i)\frac{\nabla\phi(\mathbf{x}_i)}{\|\nabla\phi(\mathbf{x}_i)\|}. \tag{4.8}$$

Par définition de la fonction de distance, $\mathbf{x}_i^{nouveau}$ est la position sur la surface de l'objet la plus près de $\mathbf{x}_i$. En pratique, la résolution avec laquelle $\phi$ est discrétisée influence la précision du résultat obtenu par l'équation précédente.

### Gestion de l'invalidité géométrique

Il arrive parfois qu'il ne soit pas possible de garantir la validité géométrique. C'est le cas lorsqu'un usager intervient et que les mouvements qu'il peut induire sont arbitraires. Les personnages animés dans les films et les jeux vidéo exhibent régulièrement des états géométriques "pathologiques", comme illustré par la figure 4.9 (b).

(a) (b)

FIG. 4.9 – (a) Personnage animé avec chandail simulé ; (b) Les bras du personnage intersectent son torse à la hauteur des aisselles. Ces deux illustrations sont tirés de l'article de Baraff *et al.* [BWK03].

Comme les logiciels de modélisation laissent souvent une liberté totale à l'artiste, ils lui permettent de créer des *keyframes* où des parties simulées du personnage peuvent entrer dans un état géométriquement invalide. Une phase de détection de collisions qui prend alors pour acquis qu'à la fin du pas de simulation les états géométriquement invalides sont inaccessibles peut alors produire des résultats incorrects. Des aberrations géométriques, comme un vêtement s'auto-intersectant, peuvent alors devenir permanentes.

Baraff *et al.* [BWK03] proposent une méthode permettant à la simulation d'être

robuste à l'invalidité géométrique, pouvant d'elle-même revenir graduellement à un état valide lorsque la situation le permet, *i.e.* lorsque la configuration du personnage perd ses pathologies. Leur méthode gère les auto-intersections du tissu[3] et les situations où le tissu est "coincé" dans une région pathologique de l'espace, comme le vêtement sous les bras du personnage de la figure 4.9 (a).

Lorsque les tests exactes de détection de collisions ne sont pas fiables, dû à l'impact de l'usager sur la simulation, il est possible qu'il faille détecter et résoudre les collisions une fois le fait accompli. Les difficultés principales, lorsque deux surfaces entrant alors en collision sont des surfaces élastiques, est de dire, pour chaque particule, si elle est ou non impliquée dans la collision, et si oui, de choisir la stratégie à adopter localement pour résoudre globalement la collision.

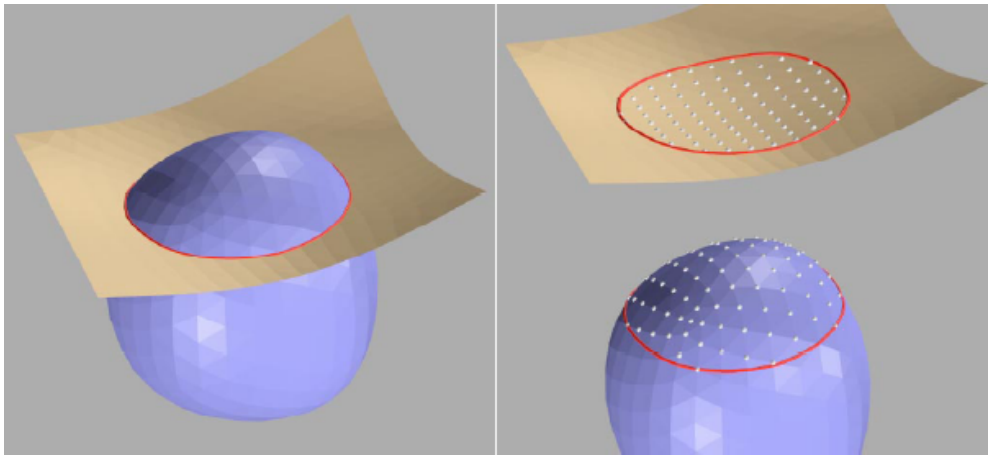


FIG. 4.10 – Courbe d'intersection. Figure tirée de l'article de Baraff *et al.* [BWK03].

Pour détecter et régler l'intersection entre deux maillages triangulaires de tissu, Baraff *et al.* [BWK03] proposent donc de faire ce qu'ils appellent une "analyse d'intersection globale" (*global intersection analysis*). À chaque pas de la simulation, une courbe d'intersection est construite pour chaque objet impliqué dans une collision.[4] Une telle courbe est illustrée à la figure 4.10. Lorsque deux intersections trouvées sont adjacentes dans un même maillage, elles font partie de la courbe d'intersection. Une fois cette courbe en main, un algorithme de remplissage par diffusion (*flood fill*) est employé

[3]Leur algorithme ne cherche pas des instants de collisions, mais gère plutôt les défaillances géométriques une fois le fait accompli.

[4]L'ensemble des triangles s'intersectant potentiellement peut être trouvé grâce à une structure accélératrice décrite dans la section précédente.

pour distinguer l'extérieur et l'extérieur de la portion de surface bornée par la courbe d'intersection. Il est alors possible de désambiguïser la situation de chaque particule, à savoir où se trouve la région pour laquelle il est géométriquement valide de placer la particule lors de la résolution de l'intersection des surfaces.

D'autre part, pour rendre la simulation plus robuste aux animations spécifiées par un artiste, Baraff *et al.* proposent une solution pour les particules se retrouvant dans une position physiquement impossible, comme l'état de coincement pathologique décrit plus tôt. Baraff *et al.* ont souligné qu'en général, lorsqu'un tissu est coincé dans une articulation pliée, comme une aisselle, les particules ont tendance à demeurer immobiles par rapport aux surfaces les coinçant, étant donné que le frottement tenant le tissu dans l'articulation est fortement susceptible de compenser pour toutes les autres forces. Pour des surfaces[5] coinçant la particule et se déplaçant les unes par rapport aux autres, Baraff *et al.* proposent de déplacer la particule suivant une pondération du déplacement de chaque surface la coinçant. Ils expriment d'abord la position de chaque particule coincée dans le référentiel local des surfaces impliquées. Ces particules cessent d'être traitées par le simulateur physique, et obtiennent leurs nouvelles positions, durant le temps où elles sont coincées, par le biais de la pondération du déplacement des surfaces coinçantes.

[5] pouvant être plus nombreuses que deux

# Chapitre 5

# Notre modèle

*On fait la science avec des faits, comme on fait une maison avec des pierres : mais une accumulation de faits n'est pas plus une science qu'un tas de pierres n'est une maison.*

Henri Poincaré

Comme nous l'avons admis d'entrée de jeu, lorsqu'utilisée dans des applications interactives, la synthèse d'animation inspirée de la physique se soucie surtout d'obtenir des résultats visuellement plausibles. En troquant exactitude pour performance, il est alors possible de simuler interactivement des mécanismes complexes. Le défi est donc de trouver quelles sont les particularités d'un phénomène physique à conserver dans un modèle simplifié pour que la simulation du phénomène demeure plausible.

Comme les artistes et les joueurs ont un impact important dans ce genre d'application interactive, les objets et les personnages animés ne sont pas toujours régis par les lois physiques réelles. Par exemple, les objets et personnages animés sont souvent accélérés violemment, voire téléportés. Si une simulation physique exacte et rigoureuse était utilisée, beaucoup d'objets finiraient en miettes sous l'intensité des accélérations subies.

Finalement, un modèle idéal d'animation basé sur la physique doit également tenir compte des besoins de l'artiste qui configure généralement la simulation lors de la phase de modélisation.

## 5.1 Modèle physique

Notre modèle de simulation est une solution à plusieurs problèmes rencontrés lors de la simulation de mouvements secondaires dans les jeux vidéo. Ses avantages sont sa grande rapidité, sa robustesse aux situations invalides et aux accélérations arbitraires que subissent les personnages animés, et le fait que les paramètres physiques qu'il utilise sont intuitifs à contrôler pour l'artiste.

Le modèle que nous proposons est couplé unidirectionnellement avec le système d'animation rigide des objets de la scène : il reçoit l'information sur la pose de chaque objet, lui permettant de simuler les mouvements secondaires, mais n'affecte pas en retour la pose de l'objet animé.

Nous utilisons un schéma d'intégration par contraintes géométriques. Dans un tel schéma, les constantes de raideur ne sont plus en unités physiques, mais en "degré d'application" de la contrainte. Ce type de constantes facilite le contrôle artistique, car il semble plus intuitif d'exprimer le degré d'élasticité d'un matériau avec un paramètre valant de 0 (mou, sans élasticité) à 1 (totalement raide) que de l'exprimer dans des unités comme les newtons par mètre ($N/m$).

Dans notre méthode, les surfaces élastiques simulées sont représentées par un maillage triangulaire où chaque sommet est une masse ponctuelle (particule) et chaque arête représente un ressort reliant deux particules.

Lorsque nous voulons qu'une partie d'un objet animé soit simulée par notre méthode, il suffit de convertir un sous-ensemble des sommets du modèle géométrique animé en particules attachées à leurs voisines. Notons que ce processus de conversion des sommets en particules est une correspondance bijective entre un sous-ensemble des sommets géométriques du modèle animé et un nombre égal de particules.

Lors de la simulation d'un vêtement ou d'un accessoire élastique, il est important de le contraindre à suivre les mouvements du personnage sur lequel il se trouve, si brusques soient-ils. C'est pourquoi, au besoin[1], certaines particules simulées sont attachées, à l'aide d'un ressort, à une particule fixée à l'emplacement du sommet correspondant

[1]Certains accessoires, comme les bandoulières, ne sont attachés, dans la simulation, que par les endroits où ils le sont normalement dans le monde réel, *i.e.* leurs extrémités. Selon notre expérience, lorsqu'un vêtement comme un chandail ou un pantalon est simulé, nous avons obtenu de meilleurs résultats lorsque chaque particule du vêtement était attachée.

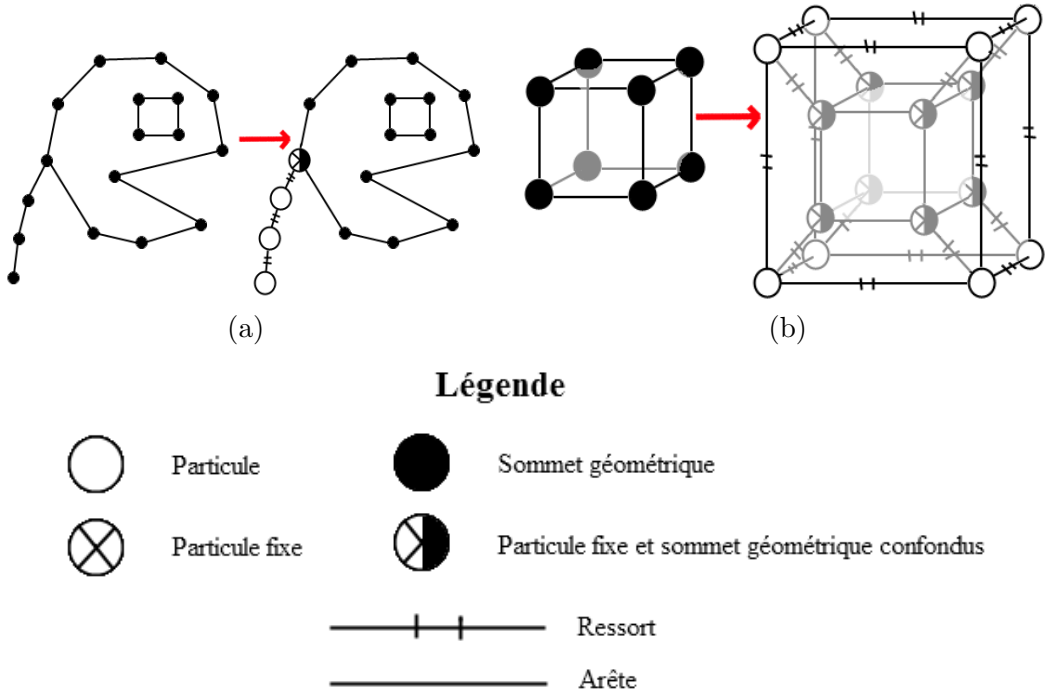



FIG. 5.1 – Ajout de régions simulées à des modèles géométriques. (a) Ajout d'une cape à un personnage. (b) Ajout d'une couche de peau sur un cube.

dans le maillage géométrique du personnage. Ces ressorts travaillent en partie comme un frottement statique qui garde les particules du vêtement près de l'endroit où elles ont été originalement placées sur le corps du personnage. La longueur au repos de chaque ressort est choisie de façon à permettre l'amplitude de mouvement que désire l'usager et chaque ressort est implémenté comme une contrainte géométrique, garantissant que la surface suive en permanence le personnage, même lorsque ce dernier est accéléré violemment. Comme des particules sont attachées à des sommets animés (par le *skinning*) du personnage, elles entraînent à leur tour la surface simulée durant l'animation.

Notons que l'utilisation de ressorts entre chaque particule mobile et une particule suivant le modèle animé est semblable à la méthode de Müller *et al.* [MHTG05], mais notre méthode utilise l'animation de l'objet pour déterminer l'évolution des particules simulées plutôt qu'une correspondance de formes. De plus, notre méthode permet de détacher certaines particules ou de spécifier une longueur de repos non-nulle pour les ressorts, rendant possible la simulation de surfaces libres comme des tissus ou des mem-

branes élastiques.

En somme, chaque particule d'un matériau simulé se retrouve dans l'une des trois catégories suivantes :

- **fixe** : particule dont la position est constante dans le référentiel de l'objet animé et sur laquelle les contraintes géométriques n'ont pas d'influence, ce type de particule servant en général de point d'attache pour les surfaces élastiques simulées ;
- **attachée** : particule qui est reliée à une particule fixe ;
- **mobile** : particule qui n'est pas attachée au modèle géométrique par une particule fixe, mais qui est contrainte par les ressorts la reliant à ses voisines.

La figure 5.1 (a) illustre, en deux dimensions, une configuration d'un personnage avec une cape, alors que la figure 5.1 (b) illustre une couche de "peau" ajoutée autour d'un cube.

## 5.2 Schéma d'intégration par contraintes géométriques

Comme nous l'avons mentionné précédemment, nous utilisons un schéma d'intégration par contraintes géométriques, ce qui facilite la compréhension intuitive des paramètres de simulation. Tel que mentionné dans la section 3.4, les contraintes géométriques utilisées sont créées à partir de forces physiques. Nous formulons ces contraintes dans les sous-sections suivantes.

Compte tenu que plusieurs contraintes géométriques peuvent s'appliquer en même temps sur une même particule, notre schéma d'intégration est une modification du schéma donné par les équations 3.12 et 3.13. Nous appliquons les contraintes les unes après les autres, ce qui se formule de la façon suivante :

$$\begin{aligned}
\mathbf{x}_{0,i}^{n+1} &= \mathbf{x}_i^n + \Delta t \dot{\mathbf{x}}_i^n + \frac{\Delta t^2}{m_i}\mathbf{F}_i^n \\
\mathbf{x}_{1,i}^{n+1} &= \mathbf{x}_{0,i}^{n+1} + k_1(\mathbf{g}_{1,i}^n - \mathbf{x}_{0,i}^n) \\
\mathbf{x}_{2,i}^{n+1} &= \mathbf{x}_{1,i}^{n+1} + k_2(\mathbf{g}_{2,i}^n - \mathbf{x}_{1,i}^n) \\
\mathbf{x}_{3,i}^{n+1} &= \mathbf{x}_{2,i}^{n+1} + k_3(\mathbf{g}_{3,i}^n - \mathbf{x}_{2,i}^n) \\
&\dots \\
\mathbf{x}_{Q,i}^{n+1} &= \mathbf{x}_{Q-1,i}^{n+1} + k_Q(\mathbf{g}_{Q,i}^n - \mathbf{x}_{Q-1,i}^n)
\end{aligned}$$

où $Q$ est le nombre de contraintes s'appliquant sur la particule $i$, $\mathbf{g}_{q,i}^n$ est l'objectif

géométrique de la contrainte $q$ de raideur $k_q$, et $\mathbf{F}_i^n$ est la somme des forces externes[2] agissant sur $i$, au pas de temps $n$. Ainsi, les équations précédentes nous permettent d'exprimer notre schéma ainsi :

$$\begin{aligned} \mathbf{x}_i^{n+1} &= \mathbf{x}_{Q,i}^{n+1} \\ \dot{\mathbf{x}}_i^{n+1} &= \frac{\mathbf{x}_i^{n+1} - \mathbf{x}_i^n}{\Delta t}. \end{aligned}$$

### 5.2.1 Ressort linéaire

Nous utilisons des contraintes agissant comme des ressorts linéaires pour simuler plusieurs types de forces physiques : l'étirement, le cisaillement et une force s'apparentant à la fois au frottement statique et à une contrainte de déformation volumique, utilisée pour garder une surface près du modèle animé, comme décrit dans la section précédente et semblable à la méthode de Müller *et al.* [MHTG05].

L'équation, pour l'objectif géométrique $\mathbf{g}_i$ d'une particule $i$, pour un ressort reliant deux particules notées $i$ et $j$, est :

$$\mathbf{g}_i = \begin{cases} \mathbf{x}_i & \text{si } i \text{ est fixe} \\ \mathbf{x}_i + k(\|\mathbf{x}_j - \mathbf{x}_i\| - l_0)\frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|} & \text{si } i \text{ est attachée à } j \text{ fixe} \\ \mathbf{x}_i + k\frac{m_j}{m_i+m_j}(\|\mathbf{x}_j - \mathbf{x}_i\| - l_0)\frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|} & \text{sinon} \end{cases} \tag{5.1}$$

où $\mathbf{x}_i$ et $\mathbf{x}_j$ sont respectivement les positions des particules $i$ et $j$, et $l_0$ est la longueur au repos du ressort. Notons que lorsque $i$ et $j$ ne sont pas fixes[3], le terme de la forme $\frac{m_i}{m_i+m_j}$ est utilisé pour tenir compte de la masse relative des particules et ainsi conserver la quantité de mouvement. De plus, comme la contrainte est appliquée symétriquement aux deux particules, et que trivialement $\frac{m_i}{m_i+m_j} + \frac{m_j}{m_i+m_j} = 1$, le déplacement est entièrement distribué aux deux particules.

### 5.2.2 Contraintes de pliage

Comme il a été décrit dans la section 4.1.2, une contrainte de pliage peut être utilisée sur chaque paire de triangles voisins du maillage de particules simulées, afin de préserver les plis créés lors de la modélisation par l'artiste. Cependant, les effets obtenus ne sont intéressants visuellement que si le maillage animé est suffisamment dense, ce qui n'est

[2] typiquement : gravité, frottement avec l'air, etc.

[3] Si l'une d'entre elles est fixe, elle possède l'équivalent d'une masse infinie ce qui fait que le terme de la forme $\frac{m_a}{m_a+m_b}$ vaut l'unité du point de vue de la particule mobile, et zéro pour la particule fixe.

que rarement le cas pour les modèles géométriques utilisés dans les jeux vidéo. En effet, les plis dans les vêtements des personnages, induits par les contraintes de pliage angulaires, ne peuvent être obtenus que si le maillage possède une résolution suffisante pour représenter les rapides variations de courbure nécessaires. C'est pour cette raison qu'après avoir expérimenté avec ce type de contrainte, nous avons décidé de l'omettre de notre modèle de simulation final.

### 5.2.3 Frottement aérodynamique, contact et gravité

Les forces comme le frottement sont souvent plus difficiles à exprimer comme des contraintes géométriques que des forces élastiques. Il est aussi plus intuitif d'imaginer le frottement comme une dissipation d'énergie cinétique en chaleur, ce qui justifie de l'exprimer comme une impulsion, modifiant directement la vitesse de la particule, juste avant la mise à jour de sa position.

Dans notre modèle, lorsqu'une particule est entrée en collision durant un pas de simulation, elle est ajoutée à la liste des points de contact du système. Chaque pas subséquent, si la particule continue à avoir un composante normale la poussant vers l'intérieur de la surface sur laquelle elle s'appuyait, elle demeure de la liste et nous lui appliquons du frottement dynamique. Pour ce faire, nous utilisons la formulation de Bridson *et al.* [BFA02], que nous avons décrite dans la section 4.1.3.

Pour ce qui est du frottement dans l'air, nous négligeons le frottement sur la surface des triangles du maillage de particules, mais appliquons le frottement par particule, avec l'équation 4.6.

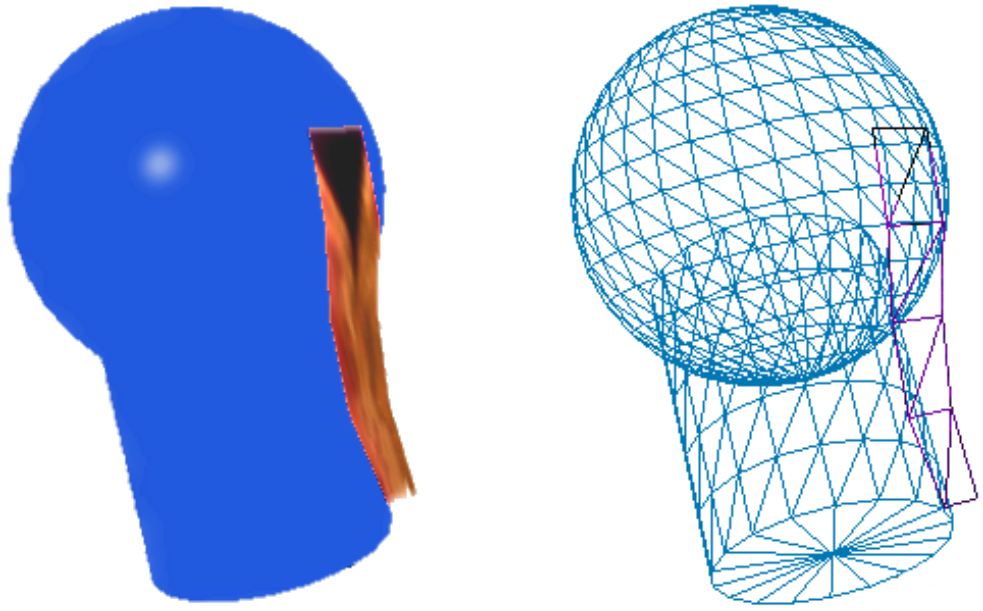
Enfin, la gravité est ajoutée comme une force externe qui déplace chaque particule mobile au début de notre schéma d'intégration.

## 5.3 Collisions

Il est typique d'utiliser un modèle géométrique simplifié pour la détection de collisions. Un tel modèle géométrique peut être un maillage polygonal, mais un ensemble de primitives englobantes, comme des sphères ou des cylindres, est souvent employé. Les primitives utilisées dans ce contexte sont en général appelées "primitives de collisions". De même, lorsqu'un maillage est utilisé, nous l'appelons "maillage de collisions"

(*collision mesh*).[4] L'ensemble des primitives de collisions est organisé selon la même structure que le squelette de l'objet animé et chaque primitive est associée à un os.

Lors de l'animation du personnage, les primitives de collisions subissent les transformations rigides des os, et leurs positions et orientations mises à jour sont utilisées dans le test de collisions. La figure 5.2 illustre les primitives utilisées pour les particules servant à simuler une mèche de cheveux sur la tête d'un personnage.



(a) (b)

FIG. 5.2 – Mèche de cheveux simulée avec deux primitives de collisions servant à prévenir l'intersection géométrique de la mèche avec le personnage. La tête du personnage est simplifiée par une sphère et le cou, par un cylindre. Le rendu est fait avec polygones texturés en (a) et fils de fer (*wireframe*) en (b). Notons que la mèche suit la courbure de la sphère et s'appuie correctement sur le cylindre.

Compte tenu que notre méthode simule des mouvements secondaires, il est possible de réduire considérablement les tests de collisions en observant que le maillage simulé demeure généralement à proximité du personnage. La raison de cette proximité est entre autres que l'amplitude du mouvement des particules peut être contrôlée par rapport au modèle animé sur lequel elles sont attachées. Ainsi, au moment de la modélisation

[4]Dans l'industrie des jeux vidéo, l'appellation *collision mesh* s'étend souvent abusivement à toutes les hiérarchies, maillages ou constructions servant à la détection de collisions.

du personnage, il est possible de spécifier d'avance quelles primitives seront utilisées à chaque pas de simulation pour le test de collisions de chaque particule. Le temps de détection de collisions pour un nombre fixe de particules donné devient donc constant, ce qui est un avantage important lorsque le budget computationnel est limité.

Dans les jeux vidéo, la détection de collisions exacte (voir la section 4.3.2) est généralement évitée puisque son temps de calcul croît trop rapidement lorsque la scène se complexifie. Des méthodes de détection et réponse "immédiates" (aussi appelées *history-free* par Baraff *et al.* [BWK03]) sont souvent utilisées pour remplacer les méthodes exactes.

Le type de détection de collisions que nous utilisons est appelé "contrainte de non-pénétration" et elle fait partie des méthodes immédiates. Le schéma d'intégration calcule la position finale de chaque particule au temps actuel, et vérifie après-coup si la particule se trouve dans un endroit interdit par les primitives de collisions qui lui sont associées. Si oui, la particule est sortie des primitives en question de la façon que nous décrivons plus loin, sinon rien n'est fait.

Nous implémentons notre phase de détection et réponse aux collisions comme un ensemble de contraintes géométriques de non-pénétration s'appliquant en dernier, après toutes les contraintes géométriques simulant la dynamique interne du système. Le paramètre de raideur pour ces contraintes est l'unité, *i.e.* le maximum possible. Les appliquer en dernier leur donne le "dernier mot", ce qui fait que les intersections géométriques sont évitées pour le rendu de la scène. Étant donné que la réponse aux collisions ne se fait que sur les particules et non sur les triangles du maillage formé par ces dernières, les primitives de collision sont ajustées pour être plus volumineuses que nécessaire, afin de réduire les artefacts de rendu, comme des triangles de vêtements intersectant le maillage géométrique du personnage animé.

Les collisions que nous modélisons sont sans restitution[5] et les particules qui sont à l'intérieur d'une primitive de collision sont simplement placées à la position la plus proche de la surface de la primitive en question. Nous présumons donc que chaque collision annule la vitesse normale à la surface de la particule et que les particules ne rebondissent pas après l'impact. Lorsqu'une particule est présente dans plus d'une primitive de collisions, les contraintes de non-pénétration, triées en ordre décroissant de

[5]sans effet de rebond

distance de pénétration, sont appliquées séquentiellement.

Remarquons que le type de détection de collisions que nous employons suppose que le déplacement des particules est plus petit que la taille des primitives de collisions. Cette supposition est généralement justifiée pour la simulation de mouvements secondaires dans les jeux vidéo, où les particules demeurent près des objets animés et où le pas de temps est petit.

Les prochaines sous-sections donnent la formulation des contraintes de non-pénétration pour les différents types de primitives avec lesquelles nous avons expérimenté durant nos recherches.

### 5.3.1 Sphère

Pour une contrainte de non-pénétration pour une sphère $S$ de centre $\mathbf{x}_c$ et rayon $r$, si $\mathbf{x}_i$ est la position de la particule $i$ sur laquelle la contrainte s'applique, l'objectif géométrique $\mathbf{g}_i$ de cette particule est donné par :

$$\mathbf{g}_i = \begin{cases} \mathbf{x}_c + r\frac{\mathbf{x}_i - \mathbf{x}_c}{\|\mathbf{x}_i - \mathbf{x}_c\|} & \text{si } \mathbf{x}_i \in \textit{intérieur}(S) \\ \mathbf{x}_i & \text{sinon} \end{cases} \tag{5.2}$$

où $\textit{intérieur}(S) = \{\mathbf{x} \ : \ \|\mathbf{x} - \mathbf{x}_c\| < r\}$.

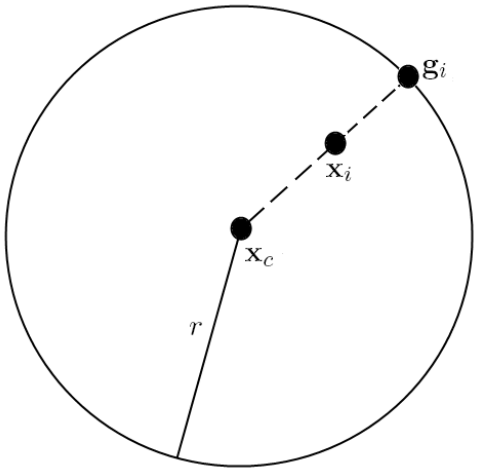



FIG. 5.3 – Sphère et objectif géométrique lors de l'application de la contrainte de non-pénétration.

La figure 5.3 illustre le cas où la particule est extraite. Pour ne pas générer un état invalide, nous supposons que le déplacement de la particule $i$ dans un pas de temps est

inférieur à $r$.

### 5.3.2 Cylindre

Soit $S$, un cylindre, illustré par la figure 5.4, dont la base de rayon $r$ est centrée en $\mathbf{x}_c$ et dont la hauteur orientée est $\mathbf{h}$, tel que $\|\mathbf{h}\| = h$, la hauteur du cylindre. Le vecteur unitaire d'orientation de ce cylindre est $\hat{\mathbf{h}} = \mathbf{h}/\|\mathbf{h}\|$.

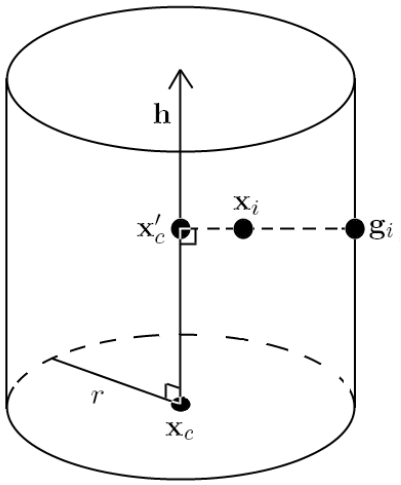



FIG. 5.4 – Cylindre et objectif géométrique lors de l'application de la contrainte de non-pénétration.

L'objectif géométrique $\mathbf{g}_i$ d'une particule $i$ placée en $\mathbf{x}_i$, sur laquelle une contrainte de non-pénétration avec $S$ est appliquée, s'exprime alors de la façon suivante :

$$\begin{aligned} \mathbf{x}'_c &= \mathbf{x}_c + ((\mathbf{x}_i - \mathbf{x}_c)\cdot\hat{\mathbf{h}})\hat{\mathbf{h}} \\ \mathbf{g}_i &= \begin{cases} \mathbf{x}'_c + r\frac{\mathbf{x}_i - \mathbf{x}'_c}{\|\mathbf{x}_i - \mathbf{x}'_c\|} & \text{si } \mathbf{x}_i \in \textit{intérieur}(S) \\ \mathbf{x}_i & \text{sinon} \end{cases} \end{aligned} \tag{5.3}$$

où $\textit{intérieur}(S) = \{\mathbf{x} \ : \ 0 < (\mathbf{x}_i - \mathbf{x}_c)\cdot\hat{\mathbf{h}} < h \quad \text{et} \quad \|(\mathbf{x}_i - \mathbf{x}_c) - ((\mathbf{x}_i - \mathbf{x}_c)\cdot\hat{\mathbf{h}})\hat{\mathbf{h}}\| < r\}$.

### 5.3.3 Triangle

Soit un triangle $S$, illustré par la figure 5.5, dont les trois sommets ont pour positions respectives $\mathbf{x}_1$, $\mathbf{x}_2$ et $\mathbf{x}_3$. Ce triangle a pour normale $\hat{\mathbf{n}} = \frac{(\mathbf{x}_2-\mathbf{x}_1)\times(\mathbf{x}_3-\mathbf{x}_1)}{\|(\mathbf{x}_2-\mathbf{x}_1)\times(\mathbf{x}_3-\mathbf{x}_1)\|}$.

Pour une particule $i$, dont la position est $\mathbf{x}_i$, l'objectif géométrique $\mathbf{g}_i$ de la contrainte de non-pénétration du triangle $S$ est donné par :

$$\mathbf{g}_i = \begin{cases} \mathbf{x}_i - ((\mathbf{x}_i - \mathbf{x}_1)\cdot\hat{\mathbf{n}})\hat{\mathbf{n}} & \text{si } \mathbf{x}_i \in \textit{projection}(S) \\ \mathbf{x}_i & \text{sinon.} \end{cases} \tag{5.4}$$

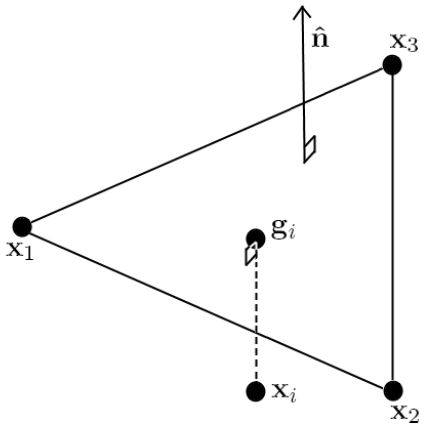



FIG. 5.5 – Triangle et objectif géométrique lors de l'application de la contrainte de non-pénétration. L'extérieur d'un triangle est le côté normal à son plan de support.

Avec $\mathbf{x}' = \mathbf{x} - ((\mathbf{x} - \mathbf{x}_1) \cdot \hat{\mathbf{n}})\hat{\mathbf{n}}$, $projection(S)$ est défini ainsi :

$$\begin{aligned} projection(S) = \{\mathbf{x} \ : \ & \frac{\|(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}' - \mathbf{x}_1)\|}{\|(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)\|} \in [0,1] \\ \text{et} \quad & \frac{\|(\mathbf{x}_3 - \mathbf{x}_2) \times (\mathbf{x}' - \mathbf{x}_2)\|}{\|(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)\|} \in [0,1] \\ \text{et} \quad & \frac{\|(\mathbf{x}_3 - \mathbf{x}_1) \times (\mathbf{x}' - \mathbf{x}_3)\|}{\|(\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1)\|} \in [0,1]\} \end{aligned}$$

où les trois conditions précédentes vérifient que les coordonnées barycentriques de $\mathbf{x}'$, la projection de $\mathbf{x}_i$ sur le plan de support de $S$, font que $\mathbf{x}'$ est borné par les trois arêtes de $S$.

## 5.4 Étendue de l'applicabilité de notre modèle

Malgré le fait que notre schéma d'intégration soit rapide et permette aux contraintes de type ressort d'être très raides, cette raideur n'est que locale. En effet, appliquer séquentiellement ce type de contraintes géométriques ne garantit pas que deux contraintes ne se contredisent pas. Ainsi, même si, dans un système simulé, toutes les contraintes de type ressort sont parfaitement raides, il n'est pas assuré que le comportement du matériau sera complètement dépourvu d'élasticité.

Pour comprendre cette problématique, étudions l'exemple simple, en 1D, montré par la figure 5.6, où un système simulé est composé de trois particules, étiquetées 1, 2 et

3. Supposons que la particule 1 est fixe et que deux contraintes géométriques $c_1$ et $c_2$ de type ressort, toutes deux de longueur de repos $l_0$, relient respectivement les paires de particules (1,2) et (2,3). Posons que la vitesse initiale des particules est nulle et que leurs positions de départ sont $x_1 = 0$, $x_2 = 2l_0$ et $x_3 = 4l_0$.

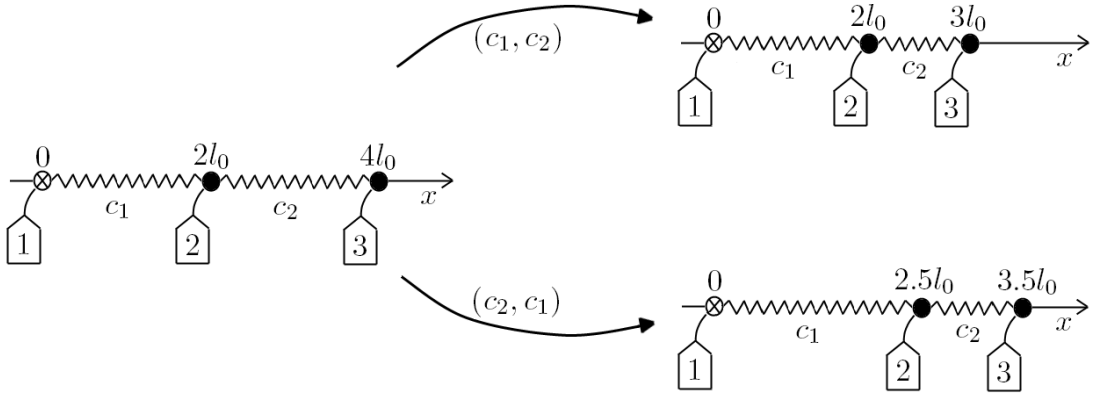



FIG. 5.6 – Différences obtenues lors d'un pas de simulation d'un système de trois particules sur lequel deux contraintes $c_1$ et $c_2$ s'appliquent, selon que l'ordre d'application des contraintes soit $(c_1, c_2)$ ou $(c_2, c_1)$.

Aussi, supposons que le système simulé représente une corde très raide et que les contraintes géométriques sont pleinement appliquées, se traduisant par $k = 1$ dans l'équation 5.1. Nous pouvons remarquer que si nous appliquons les contraintes dans l'ordre $(c_1, c_2)$, les positions finales après un pas de simulation sont $x_1 = 0$, $x_2 = 2l_0$ et $x_3 = 3l_0$ ; si l'ordre $(c_2, c_1)$ est utilisé les positions obtenues sont $x_1 = 0$, $x_2 = 2.5l_0$ et $x_3 = 3.5l_0$. Donc, malgré le fait que le système soit modélisé de façon à obtenir une corde de longueur maximale de $2l_0$, nous obtenons les longueurs $3l_0$ et $3.5l_0$ selon l'ordre dans lequel $c_1$ et $c_2$ sont appliquées.

À la lumière de cet exemple, il semble que non seulement l'ordre d'application des contraintes peut influencer considérablement le résultat, mais aussi que le résultat voulu par l'usager est peut-être inaccessible, ce qui est le cas dans l'exemple simple présenté ci-haut.

Remarquons que l'origine de ce problème réside dans le fait que nous permettons à certaines particules d'être mobiles.[6] La position relative des particules mobiles par

[6] voir la définition des types de particules au début de la section 5.2

rapport aux particules fixes[7] n'est donc contrainte qu'indirectement, par le biais d'une chaîne de contraintes entre des particules mobiles.

Nous avons obtenu de bons résultats en attachant toutes les particules lors de la simulation d'un vêtement de personnage, comme un chandail, mais certains matériaux, comme les bandoulières ou les mèches de cheveux, se doivent d'être plus libres, *i.e.* attachées à peu d'endroits, pour que les résultats soient convaincants.

### 5.4.1 Ordonnancement des contraintes

La solution que nous proposons, et qui permet à notre modèle de simuler des surfaces beaucoup plus raides qu'auparavant, est de modifier l'algorithme d'application des contraintes. Notre solution trie topologiquement les contraintes de type ressort de la surface de l'objet, en partant des particules fixes vers les particules mobiles, de la manière réalisée par l'algorithme 5.1. Notons que notre algorithme ne circule pas le long des ressorts servant à retenir la surface près de l'objet, mais seulement ceux faisant effectivement partie de la surface.

La justification physique pour cet ordonnancement des contraintes est qu'il suit la direction de la propagation du mouvement dans les matériaux réels. Par exemple, quand nous tirons sur l'extrémité d'une corde, ce n'est qu'après que "l'information" du mouvement se soit propagée, par le biais de la tension dans la corde, que l'autre extrémité, celle qui est libre, se déplace.

Durant la simulation, en plus de l'ordonnancement que nous venons de décrire, nous étiquetons les particules ayant déjà été déplacées par une contrainte afin de ne pas les déplacer de nouveau lorsqu'une autre contrainte essaie de s'y appliquer, ce qui évite de perdre les gains déjà obtenus en faveur de la raideur du matériau. Lorsqu'une contrainte de constante de raideur $k$ doit s'appliquer sur deux particules déjà étiquetées, nous la laissons s'appliquer avec une constante de raideur $\gamma k$ où $\gamma$ est contrôlable durant la modélisation. Dans des structures physiques comme des sangles, de telles contraintes s'appliquent souvent dans la direction orthogonale à la direction partant des coins fixes vers l'extrémité libre de la sangle. Ces contraintes ne participent que peu à l'aspect de raideur du matériau, mais elles peuvent être importantes pour maintenir la structure de la surface si cette dernière, par exemple, est drapée sur une primitive de collisions.

[7] fixe dans le référentiel local de l'objet animé

ALGORITHME 5.1 : Tri des contraintes

**Fonction :** TriTopologiqueRessort()

**Type de retour :** Liste{Ressort}

```
1.  Liste{Ressort} ordreRessort ;
2.  FileFIFO{Particule} àVisiter ;
3.  // nous ajoutons les particules fixes dans une file FIFO
4.  pour chaque particule i
5.      si i est fixe alors
6.          àVisiter.ajouterQueue(i)
7.  // visitons topologiquement en suivant les contraintes-ressorts
8.  tant que àVisiter ≠ Ø
9.      Particule i ← àVisister.enleverTête() ;
10.     i.visitée ← vrai ;
11.     pour chaque Ressort c faisant partie de la surface et relié à i
12.         si c.visité = faux alors
13.             c.visité ← vrai ;
14.             ordreRessort.ajouterQueue(c) ;
15.             Particule j ← c.autreParticuleQue(i) ;
16.             si j.visitée = faux alors
17.                 àVisiter.ajouterQueue(j) ;
18. retourner ordreRessort ;
```

Notons que le processus de détection et réponse aux collisions se déroule toujours en dernier et n'est aucunement affecté par l'ordonnancement des contraintes ou l'étiquetage des particules.

Comme nous l'avons mentionné dans la section 5.1, notre simulateur est couplé unidirectionnellement avec le système d'animation. Il est donc possible que l'animation force les ressorts à atteindre une longueur plus grande que permise durant la simula-

tion, par le biais des primitives de collisions ou des particules fixes se déplaçant, et ce, malgré l'ordonnancement des contraintes. Nous tenons à préciser que notre modèle n'a pas pour objectif de régler ce problème, car il est conçu justement dans l'optique de ne pas contraindre l'artiste. En revanche, ce dernier est responsable, lorsqu'il désire que des matériaux soient simulés de façon plausible, de ne pas mettre les objets animés dans des situations incohérentes. En conséquence, la pire situation observable avec notre modèle est que le résultat soit invraisemblable durant le temps où la configuration du personnage est incohérente. Notre modèle est tout de même robuste aux situations invalides, retrouvant un état adéquat dès que la configuration forcée s'amenuise, contrairement à la plupart des autres modèles, pouvant littéralement exploser dans des situations semblables.

## 5.5 Modélisation artistique des paramètres physiques

La formulation mathématique des modèles physiques fait souvent intervenir de nombreux paramètres physiques, caractérisant les propriétés de la matière. Lorsqu'il est question de synthétiser les mouvements secondaires, la raideur de l'objet simulé est un paramètre incontournable. Cependant, d'autres constantes physiques propres au matériau animé ou à son environnement sont à prévoir, comme la constante de frottement entre la surface simulée et les objets statiques de la scène.

Les artistes, qui travaillent habituellement de façon intuitive, et même les spécialistes des simulations physiques auront souvent du mal à interpréter rapidement l'impact qu'aura une valeur particulière d'un paramètre physique lorsque les unités sont complexes. Des calculs sont souvent nécessaires pour permettre la compréhension.

Comme nous l'avons déjà mentionné au début de la section 5.1, nous utilisons un schéma d'intégration par contraintes géométriques, impliquant que notre modèle utilise des paramètres physiques sans unité, variant de 0 à 1, et signifiant toujours la même chose : le degré d'application de la contrainte associée au paramètre. Cette particularité de la méthode simplifie beaucoup la compréhension intuitive de leur impact sur la simulation. Ces paramètres au domaine bien défini facilitent la modélisation d'objets simulés.

À partir d'un modèle géométrique, le processus d'ajout de régions simulées par notre

méthode se fait par le biais de différentes textures spéciales, que nous appelons “textures de paramètres”, que l'artiste peint directement sur le modèle, à l'aide de l'outil de *vertex painting* du logiciel de modélisation utilisé. En général, chacune de ces textures est associée à une contrainte interne d'élasticité ou de frottement de l'objet simulé, à l'exception d'une texture de contrôle, servant à passer l'information servant à la conversion des sommets appropriés en particules. Cette information est un ensemble de drapeaux (*flags*) indiquant si une particule est fixe, attachée ou mobile[8], et donnant l'amplitude du mouvement des particules attachées. Une fois toutes ces textures générées, les coordonnées de textures contenues dans chaque sommet, accessibles du côté de l'application interactive, sont utilisées lors de la simulation pour accéder aux valeurs des paramètres et des drapeaux.

[8] voir la définition des types de particules au début de la section 5.2

# Chapitre 6

# Résultats

*Ce qui est affirmé sans preuve peut être nié sans preuve.*

Euclide

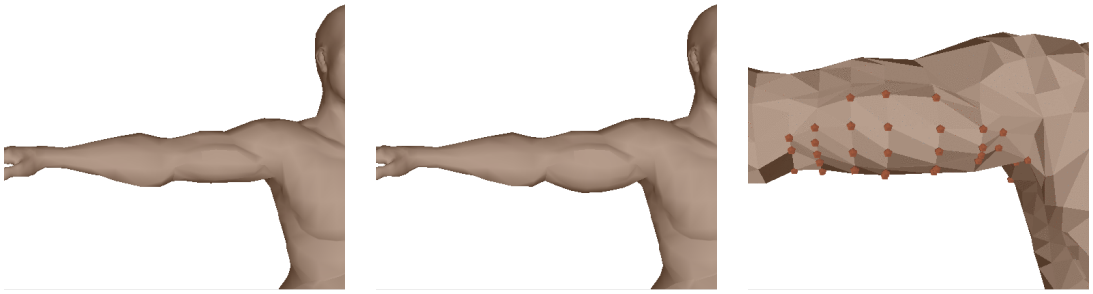
Nous avons implémenté notre modèle de simulation avec le langage de programmation C++, à l'aide des interfaces de programmation *OpenGL* [Ope] et *DirectX* [Mic]. Lors de notre séjour au studio de développement de jeux vidéo d'*Electronic Arts Montréal*, nous avons eu l'occasion de faire interagir notre modèle avec un système d'animation existant, dans lequel des accessoires simulés par notre technique sont ajoutés sur un personnage contrôlé par le joueur.

Nous avons réalisé nos tests sur des modèles géométriques comprenant entre 7000 et 30000 polygones, et les surfaces simulées étaient composées de 200 à 5000 particules. Dans tous nos résultats, les temps de simulation sont en dessous de cinq millisecondes sur un poste de travail à deux processeurs *Intel Xeon* cadencés à 3.6 GHz et possédant une carte graphique *nVidia GeForce 7800 GT* avec 256 Mo de mémoire vidéo.

Des séquences vidéo de notre technique, utilisée dans une application interactive où l'usager peut manipuler les objets possédant des régions simulées, sont disponibles à l'adresse `http://www.iro.umontreal.ca/labs/infographie/theses/simardya`, la page Web associée à ce mémoire.

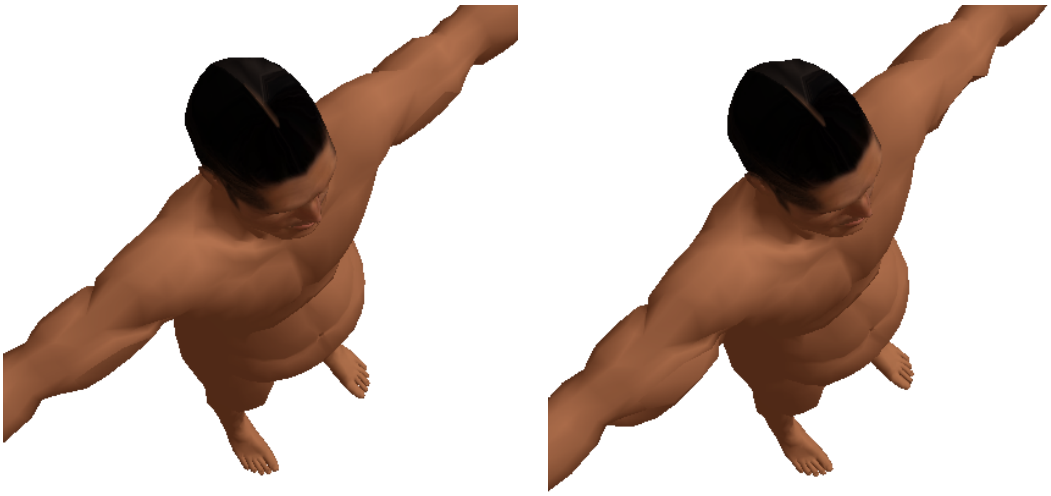
Nous montrons ici, aux figures 6.1 à 6.7, quelques exemples de résultats que nous avons obtenus en simulant de la peau au comportement élastique, ainsi qu'une bandoulière et des sangles raides. Les éléments plus intéressants de chaque résultat sont

expliqués dans leurs légendes respectives.



(a) (b) (c)

FIG. 6.1 – Illustration d'un bras dont le biceps est simulé avec notre méthode. (a) Modèle original. (b) Biceps simulé, s'étirant sous l'effet de la gravité. (c) Particules simulées pour créer un biceps mou.

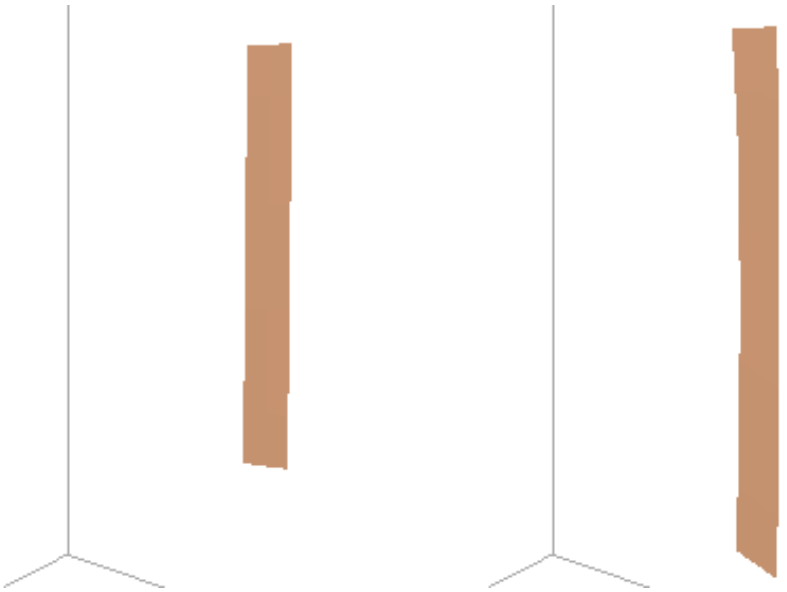


(a) (b)

FIG. 6.2 – Un personnage virtuel possédant des biceps mous. La figure (a) montre le personnage vu de haut, immobile. La figure (b) montre l'étirement des biceps du personnage lorsque ce dernier est en accélération angulaire dans le sens horaire.

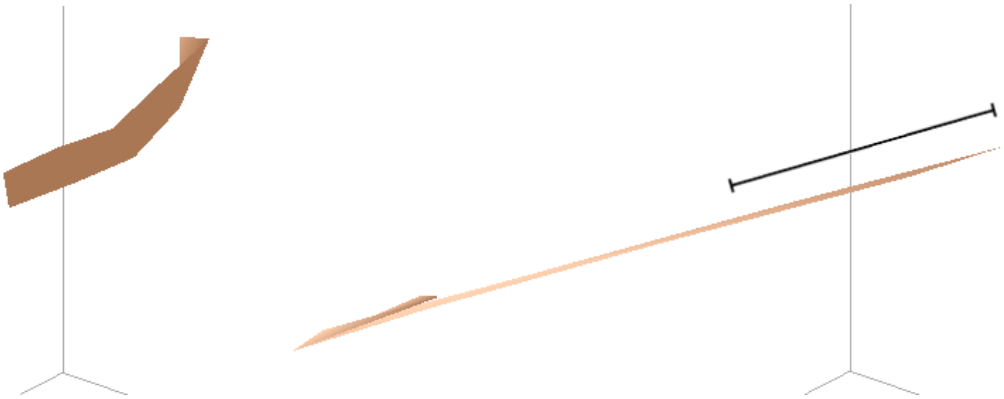
(a) (b) (c)

FIG. 6.3 – Personnage virtuel possédant un nez mou. Trois situations sont ici montrées : le personnage est immobile en (a), le nez s'écrase, dans la figure (b), lorsque le personnage accélère brusquement vers la gauche et le nez s'allonge, dans la figure (c), lorsque le modèle accélère vers la droite.



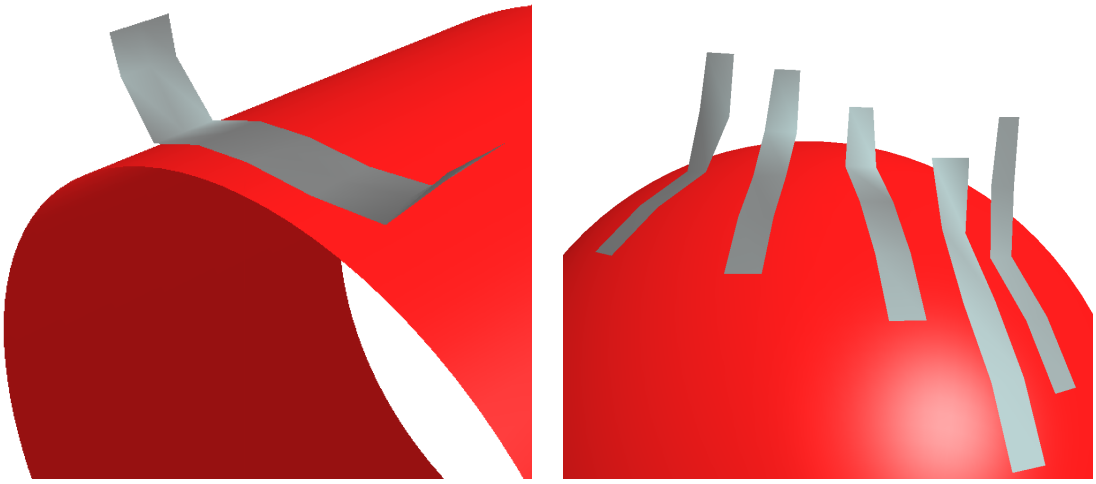
(a) (b)

FIG. 6.4 – Bande de tissu raide, suspendue par ses deux coins supérieurs, une fois immobilisée. Comparaison entre notre méthode avec ordonnancement des contraintes en (a) et sans ordonnancement en (b). Notons que la bande en (a) converge vers la solution théorique à un epsilon machine près ; en (b), la bande est environ 20% plus longue que sa longueur au repos.



(a) (b)

FIG. 6.5 – Accélération vers la droite des points d'attache d'une bande de tissu raide accrochée par ses deux coins supérieurs. La figure (a) illuste le résultat obtenu en utilisant notre modèle avec ordonnancement des contraintes, alors que la figure (b) est obtenue sans ordonnancement. Nous avons indiqué par un échelon noir, dans la figure (b), la longueur de la bande au repos afin de bien faire remarquer que la bande voit sa longueur plus que doubler, alors qu'en (a), elle n'augmente que de 1 % lors d'une accélération semblable.



(a) (b)

FIG. 6.6 – Collisions avec des primitives. La figure (a) montre une bande de tissu retenue par ses deux extrémités en contact avec un cylindre, alors que la figure (b) montre cinq bandes retenues par une seule de leurs extrémités lors du contact avec une sphère.

FIG. 6.7 – Bandoulière simulée attachée à une arme militaire. Remarquons la courbe caractéristique que prend naturellement la bandoulière sous l'effet de la force gravitationnelle, pour les deux inclinaisons illustrées. *Artwork ©2006 Electronic Arts Inc. All right reserved.*

# Chapitre 7

# Travaux futurs

*On ne fait jamais attention à ce qui a été fait ; on ne voit que ce qui reste à faire.*

Marie Curie

Les mouvements secondaires ne se limitent pas aux matériaux que notre modèle est en mesure de simuler. En outre, voici quelques pistes de réflexions sur la progression que pourraient prendre nos recherches.

## 7.1 Modèles à couches multiples

L'un des mouvements secondaires les plus fréquemment observés sur un individu réel est celui de la peau recouvrant ses muscles. Lorsqu'une articulation plie, c'est souvent sous l'action d'un muscle s'étant contracté. La forme de la peau avoisinante est alors influencée par l'articulation, formant des plis, et le muscle, étirant la peau. De plus, les muscles eux-mêmes sont contraints par l'ossature sous-jacente.

L'ajout de ce type d'interactions entre différentes couches de matériaux augmenterait la variété de situations où notre modèle peut être utilisé pour simuler des mouvements secondaires.

Pour arriver à introduire la relation entre les couches de matériaux, il nous semble possible de décrire cette dernière en termes de contraintes reliant chaque particule d'une couche à une ou plusieurs particules se trouvant dans un voisinage sur chaque couche adjacente. Turner et Thalmann [TT93] ont décrit une idée semblable pour la simulation

de peau, où une surface est attachée à un volume simulé sous-jacent, mais nous proposons une idée plus générale, qui simulerait chaque couche selon notre paradigme basé sur les contraintes géométriques.

Chaque particule $i$ peut être située dans le référentiel de son voisinage associé, un référentiel correspondant à chaque couche adjacente. Une contrainte géométrique peut servir ensuite à ramener la particule $i$ vers la position en espace monde correspondant à son objectif géométrique dans le référentiel de chaque voisinage. Comme plusieurs de ces contraintes s'appliquent alors sur chaque particule, une somme pondérée des différents objectifs géométriques, chaque poids étant fonction de la raideur de sa couche correspondante, peut être employée pour déterminer l'objectif géométrique final. Il s'agit d'une généralisation à couches multiples de l'idée d'avoir un seul objectif géométrique par particule, décrite par Müller *et al.* [MHTG05].

## 7.2 Implémentation distribuée

La simulation de mouvements secondaires s'adapte bien au calcul distribué. Les régions simulées des objets sont en général indépendantes géométriquement les unes des autres et ne s'affectent pas physiquement, ce qui permet des gains de performances en prenant avantage du parallélisme.

En ce sens, deux avenues contemporaines nous semblent prometteuses pour l'amélioration de notre méthode : l'implantation sur GPU et l'utilisation de la capacité des nouveaux processeurs à noyaux multiples (*multicore*).

### 7.2.1 Calcul sur GPU

Le matériel d'accélération graphique est capable de traiter plusieurs sommets à la fois dans les *vertex shaders* et plusieurs fragments à la fois dans les *pixel shaders*. À titre d'information, une carte graphique récente, comme la *nVidia GeForce 7800 GTX*, peut traiter huit sommets en parallèle et 24 fragments en parallèle.

Il est d'ailleurs maintenant possible d'implémenter une vaste gamme d'algorithmes numériques sur le GPU, comme le soulignent Krueger et Westermann [KW03]. De plus, remarquons que l'un des goulots d'étranglement de notre méthode sur ordinateur est la transmission de la géométrie de la mémoire principale vers le matériel graphique

pour l'afficher, se faisant relativement lentement par le biais du bus de transmission de données.

À partir du *Shader Model 3.0*, la version du standard de programmation sur GPU la plus récente au moment où ce mémoire est écrit, il est possible de lire dans des textures à partir d'un *vertex shader*. Comme les *pixel shaders*, de leur côté, peuvent diriger leurs résultats dans une texture, il est possible d'implémenter notre modèle de simulation comme une technique de rendu à multiple passes. Plutôt que de générer une image, les *pixel shaders* produisent une texture géométrique, contenant position, vitesse, normale, etc., des sommets simulés, dont le *vertex shader* se sert dans une phase finale positionnant les sommets avant le rendu pour affichage de l'objet.

Avant que ne commence la simulation, une texture de connectivité doit être générée, de la manière décrite par Calver [Cal04], ainsi qu'une texture contenant l'état géométrique initial du système. Pour chaque pas de la simulation, une passe de *pixel shader* est appliquée, où chaque fragment "rendu" correspond à une particule simulée. Les contraintes sont alors appliquées en fonction de la connectivité du maillage simulé. Finalement, juste avant le rendu conventionnel à l'écran, les sommets géométriques passent par un *vertex shader* qui les place aux positions appropriées, calculées par le *pixel shader* lors de la phase précédente.

### 7.2.2 Microprocesseur à noyaux multiples

L'une des particularités excitantes des consoles de jeu de la prochaine génération (*next-gen*), comme la *Microsoft Xbox 360* [Mic] et la *Sony PlayStation 3* [Son], est la présence d'un microprocesseur à noyaux multiples. L'utilité d'un tel microprocesseur est de permettre à plusieurs *threads* matériels de s'exécuter en parallèle. De plus, dans la *Xbox 360*, la mémoire principale est partagée avec la mémoire vidéo, ce qui permet d'éviter le transfert coûteux de géométrie vers le *pipeline* de rendu lorsqu'elle est d'abord traitée sur CPU.

Peu de changements sont à effectuer pour faire s'exécuter notre méthode en parallèle, sinon de s'assurer de synchroniser les données nécessaires (poses des personnages influençant les mouvements secondaires) avant de s'en servir.

Étant donné que le nombre de *threads* matériels est de six sur la *Xbox 360* et sera d'au moins autant sur la *PlayStation 3*, il est même possible d'utiliser plusieurs *threads*

à la fois pour le calcul de mouvements secondaires, permettant ainsi d'incorporer notre méthode à un très grand nombre d'objets animés de la scène.

# Chapitre 8

# Conclusion

*Trop de lecture peut étouffer le génie.*

Jean le Rond d'Alembert

Dans les jeux vidéo, la simulation des mouvements secondaires d'un personnage augmente le niveau de détails et le réalisme de l'animation. Plusieurs techniques existantes simulent les déformations de corps mous ou de tissus pour les films d'animation et les applications interactives.

Ce mémoire décrit notre modèle de simulation de mouvements secondaires servant à synthétiser l'évolution dans le temps de différents types de surfaces élastiques. Nos contributions sont : un modèle simple à utiliser avec des modèles géométriques animés dans le contexte de jeux vidéo, un schéma d'intégration par contraintes géométriques pour simuler avec grande rapidité et stabilité la dynamique interne des surfaces élastiques, des paramètres de simulation intuitifs à choisir et modéliser, un ordonnancement des contraintes pour les surfaces raides, ainsi qu'un ensemble de contraintes de non-pénétration pour lequel le temps de calcul est globalement constant.

Les surfaces simulées sont représentées comme un ensemble de masses ponctuelles (particules) reliées entre elles et attachées à un objet animé. Les méthodes jusqu'ici utilisées pour simuler ces surfaces déformables dans les environnements virtuels utilisent soit des schémas d'intégration explicites, n'étant pas stables pour les matériaux raides, soit des schémas implicites ou semi-implicites, relativement coûteux en mémoire et en temps de calcul. Plutôt qu'être basée sur la mesure de forces, la méthode que nous proposons utilise un schéma d'intégration par contraintes géométriques, ce qui lui donne à

la fois une stabilité garantie et une grande rapidité. De plus, grâce à un ordonnancement des contraintes de type ressort, réalisé par un tri topologique se propageant dans les surfaces simulées en partant des particules fixes, notre modèle arrive à animer de façon plausible des matériaux presque dépourvus d'élasticité comme des sangles ou des mèches de cheveux.

Les régions du maillage polygonal choisies par l'artiste pour être simulées sont converties en particules et contraintes géométriques de type ressort, où les sommets du maillage deviennent des masses et les arêtes, des ressorts. Des contraintes géométriques de non-pénétration, traitées à la manière des autres contraintes par le schéma d'intégration, sont utilisées pour que les particules conservent une position valide. Les contraintes de non-pénétration sont définies par des primitives volumétriques choisies au moment de la modélisation et suivant le squelette de l'objet animé.

Contrairement aux différentes unités que possèdent les constantes physiques des autres méthodes, les constantes de simulation utilisées par notre technique sont très intuitives à choisir car elles représentent le niveau d'application de la contrainte y étant associée. Comme chaque constante varie entre 0 et 1, le choix par l'artiste d'une valeur de constante se fait en peignant directement en tons de gris la valeur voulue sur la surface de l'objet modélisé.

En somme, notre méthode peut servir pour simuler un éventail intéressant de mouvements secondaires comme de la peau, des tissus et des sangles raides, tout en étant très appropriées aux importantes contraintes de performances propres aux jeux vidéo.

# Bibliographie


[Age] Ageia. `http://www.ageia.com/`.

[Ali] Alias. `http://www.alias.com/`.

[BFA02] Robert Bridson, Ronald Fedkiw et Jon Anderson. « Robust Treatment of Collisions, Contact and Friction for Cloth Animation ». Dans *Proc. SIGGRAPH 2002*, pages 594–603, 2002.

[BH92] David E. Breen et Donald H. House. « A Physically-Based Particle Model of Woven Cloth ». Dans *The Visual Computer*, pages 264–277, 1992.

[BMF03] Robert Bridson, Sebastian Marino et Ronald Fedkiw. « Simulation of Clothing with Folds and Wrinkles ». Dans *Proc. Eurographics/SIGGRAPH Symposium on Computer Animation*, pages 28–36, 2003.

[BW97] David Baraff et Andrew Witkin. « Physically-based Modeling : Principles and Practice ». Dans *SIGGRAPH 1997 Course Notes*. Association for Computing Machinery, ACM SIGGRAPH, août 1997.

[BW98] David Baraff et Andrew Witkin. « Large Steps in Cloth Simulation ». Dans *Proc. SIGGRAPH 1998*, pages 43–54, 1998.

[BWK03] David Baraff, Andrew Witkin et Michael Kass. « Untangling Cloth ». Dans *Proc. SIGGRAPH 2003*, pages 862–870, 2003.

[Cal04] Dean Calver. *ShaderX$^3$ : Advanced Rendering with DirectX and OpenGL*, chapitre 1.1. Charles River Media, 2004.

[CGC+02a] Steve Capell, Seth Green, Brian Curless, Tom Duchamp et Zoran Popović. « Interactive Skeleton-Driven Dynamic Deformations ». Dans *Proc. SIGGRAPH 2002*, pages 586–593, 2002.

[CGC+02b] Steve Capell, Seth Green, Brian Curless, Tom Duchamp et Zoran Popović. « A Multiresolution Framework for Dynamic Deformations ». Dans *Proc. Eurographics/SIGGRAPH Symposium on Computer Animation*, pages 41–48, 2002.

[Dis] Discreet. `http://www.discreet.com/`.

[DMT98] Issam Doghri, Arthur Muller et Robert L. Taylor. « A general three-dimensional contact procedure for implicit finite element codes ». *Engineering Computations*, volume 15, numéro 2, pages 233–259, 1998.

[DSB99] Mathieu Desbrun, Peter Schröder et Alan Barr. « Interactive Animation of Structured Deformable Objects ». Dans *Proc. Graphics Interface '99*, pages 1–8, juin 1999.

[FvdPT97] Petros Faloutsos, Michiel van de Panne et Demetri Terzopoulos. « Dynamic Free-Form Deformations for Animation Synthesis ». *IEEE Trans. on Visualization and Computer Graphics*, volume 3, numéro 3, pages 201–214, juillet 1997.

[GHDS03] Eitan Grinspun, Anil H. Hirani, Mathieu Desbrun et Peter Shröder. « Discrete Shells ». Dans *Proc. Eurographics/SIGGRAPH Symposium on Computer Animation*, pages 62–67, 2003.

[Hav] Havok. `http://www.havok.com/`.

[HB00] Donald H. House et David E. Breen, éditeurs. *Cloth Modeling and Animation*. A K Peters, 2000.

[ICK+99] Kenneth E. Hoff III, Tim Culver, John Keyser, Ming Lin et Dinesh Manocha. « Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware ». Dans *Proc. SIGGRAPH 1999*, pages 277–286, 1999.

[Kaw80] Sueo Kawabata. « The Standardization and Analysis of Hand Evaluation ». Osaka, 1980. The Textile Machinery Society of Japan.

[KW03] Jens Krueger et Ruediger Westermann. « Linear algebra operators for GPU implementation of numerical algorithms ». *ACM Transactions on Graphics*, volume 22, numéro 3, pages 908–916, 2003.

[MDDB01] Mark Meyer, Gilles Debunne, Mathieu Desbrun et Alan H. Barr. « Interactive Animation of Cloth-like Objects in Virtual Reality ». *The Journal of*

*Visualization and Computer Animation*, volume 12, numéro 1, pages 1–12, mai 2001.

[MHTG05] Matthias Müller, Bruno Heidelberger, Matthias Teschner et Markus Gross. « Meshless Deformations Based on Shape Matching ». Dans *ACM Transactions on Graphics*, volume 24, pages 471–478, 2005.

[Mic] Microsoft. `http://www.microsoft.com`.

[Ope] OpenGL. `http://www.opengl.org`.

[OZH00] James F. O'Brien, Victor B. Zordan et Jessica K Hodgins. « Combining Active and Passive Simulations for Secondary Motion ». *IEEE Computer Graphics and Applications*, volume 20, numéro 4, août 2000.

[Pro95] Xavier Provot. « Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior ». Dans *Proc. Graphics Interface*, pages 174–154, 1995.

[SF89] Juan C. Simo et D. D. Fox. « On stress resultant geometrically exact shell model. Part I : formulation and optimal parametrization ». *Computer Methods in Applied Mechanics and Engineering*, volume 72, numéro 3, pages 267–304, 1989.

[Sof] Softimage. `http://www.softimage.com/`.

[Son] Sony. `http://www.sony.com`.

[SP86] Thomas W. Sederberg et Scott R. Parry. « Free-Form Deformation of Solid Geometric Models ». Dans *Proc. SIGGRAPH 86*, volume 20, pages 151–160, août 1986.

[TKH+05] Matthias Teschner, Stefan Kimmerle, Bruno Heidelberger, Gabriel Zachmann, Laks Raghupathi, Arnulph Fuhrmann, Marie-Paule Cani, François Faure, Nadia Magnenat-Thalmann, Wolfgang Strasser et Pascal Volino. « Collision Detection for Deformable Objects ». *Computer Graphics forum*, volume 24, numéro 1, pages 61–81, mars 2005.

[TPBF87] Demetri Terzopoulos, John Platt, Alan Barr et Kurt Fleischer. « Elastically Deformable Models ». Dans *Proc. SIGGRAPH 1987*, pages 205–214, 1987.

[TT93] Russell Turner et Daniel Thalmann. « The Elastic Surface Layer Model for Animated Character Construction ». Dans *Proc. Computer Graphics International 93*, pages 399–412, 1993.