UNIVERSITÉ DE LIMOGES

ECOLE DOCTORALE Science — Technologie — Santé

FACULTÉ des Sciences et Techniques — Laboratoire XLIM UMR CNRS 6172

Département Mathématiques Informatique - Projet Synthèse d'images Réaliste

Thèse N°462007

Thèse

pour obtenir le grade de

DOCTEUR DE L' UNIVERSITÉ DE LIMOGES

Discipline / Spécialité : Informatique

présentée et soutenue par

Alexandre AHMAD

le 16 octobre 2007

Animation De Structures Déformables Et Modélisation Des Interactions Avec Un Fluide Basées Sur Des Modèles Physiques

Thèse dirigée par Djamchid Ghazanfarpour

Co-dirigée par Olivier Terraz et Samir Adly

JURY :

rapporteurs:

M. Jean-Pierre Jessel, professeur à l'Université de Toulouse

M. Philippe Meseure, maître de conférences HDR à l'Unversité de Poitiers

examinateurs :

M. Jean-Michel Dischler, professeur à l'Université de Strasbourg

M. Djamchid Ghazanfarpour, professeur à l'Université de Limoges

M. Samir Adly, professeur à l'Université de Limoges

M. Olivier Terraz, maître de conférences à l'Unversité de Limoges

Remerciements

Je voudrais remercier vivement messieurs Jean-Michel Dischler, Jean-Pierre Jessel et Philippe Meseure d'avoir accepté d'examiner ma thèse malgré leur emploi du temps chargé.

Merci également à mes trois encadrants, Samir Adly, Djamchid Ghazanfarpour et tout particulièrement Olivier Terraz de m'avoir permis de réaliser cette thèse.

Merci aussi à la région du Limousin pour m'avoir accordé pendant ces trois années son soutien financier.

Je présente mes honneurs à ma femme qui a su endurer mes absences et mes blagues peu onéreuses pendant cette période[1].

Merci à mes collègues, que dis-je à mes amis ! Stan, tu as fleuri ce bureau grisonnant. Night Teamers, Kosovar, Damien, Fat Kev, vous avez rendu ces nuits tellement improductives, merci de tous vos conseils utiles et inutiles. Merci Damien de m'avoir fait découvrir le côté obscur de la recherche et les joies de la randonnée[2]. Compagnon de nuit, Kosovar, à tout de suite ! Merci également à Sylvain et Yann pour leurs mails, à Geekette, Passe-partout, Benoit (les deux), Switch, Emmanuelle, Nadir et Suzie pour toutes ces discussions. Et comme ce n'est pas le moment d'en écrire tout un roman, merci à tous les autres.

Merci à ma famille sans qui je ne serais certainement pas né, et à tous mes amis tout simplement pour leur présence.

Merci à Jean-Michel Jarre, Gary Jules et autres artistes qui ont su étouffer la pollution sonore des bureaux.

Merci à la communauté internationale scientifique et des logiciels libres qui m'ont fait découvrir un monde dont je fais fièrement partie aujourd'hui.

Sur ce, apt-get install theTomatoProject !

[1] ce n'est qu'un début cependant.

[2] La notion de joie est dépendante du point de vue.

Résumé

Les images de synthèse permettent de reproduire avec réalisme le monde qui nous entoure. Ces images agrémentent, entre autres, les spectaculaires effets spéciaux cinématographiques mais aussi les univers virtuels d'applications interactives. La reproduction réaliste dans les images de synthèse de certains phénomènes qui alimentent notre quotidien, tels que les fluides, n'est possible qu'en introduisant des principes physiques. Dans ce cadre, je me suis intéressé à la simulation d'interactions entre un fluide et des membranes déformables, c'est-à-dire des objets sans épaisseur. Ce sujet ne manque pas d'applications puisque les feuilles, les habits ou même les nageoires de poisson sont considérés comme des membranes et leur interaction avec un fluide est plus que fréquente dans la réalité. Pourtant, peu de travaux en synthèse d'images se sont intéressés à reproduire ce phénomène.

Pour ce faire, je propose un algorithme de réaction de contacts entre un fluide Lagrangien et une membrane déformable qui est indépendant du pas d'intégration temporelle. Afin de pallier les artefacts visuels dus à des triangles, issus du processus d'extraction de surface du fluide, qui passent au travers de la membrane lors d'interactions, je propose une technique qui projette la portion de volume erronée le long de la membrane. Dans ce cadre, une paramétrisation intuitive de coefficients hétérogènes de friction sur une surface est exposée.

Je présente également deux modèles d'optimisations. Le premier modèle fait suite à des études sur le calcul numérique du mouvement de structures Lagrangiennes. Je propose un filtrage des vitesses pour les méthodes explicites afin de réduire la forte contrainte sur le pas d'intégration temporelle. Des analyses fréquentielles permettent de quantifier cette réduction. Le deuxième modèle est une structure de subdivision spatiale dynamique qui détermine efficacement le voisinage d'un point et d'un triangle. Cette structure a pour but de réduire les temps de calculs des processus de détermination du voisinage pour les fluides Lagrangiens, mais aussi ceux de la détection de collisions et enfin ceux de l'extraction de surface.

Table des matières

Table des figures

Chapitre Premier

Introduction

Les images de synthèse connaissent un succès sans commune mesure grâce à leur présence, entre autres, dans les productions cinématographiques. L'utilisation de ces images virtuelles permet de concrétiser les rêves les plus fantastiques et extravagants des réalisateurs. Elles permettent aussi la création de mondes virtuels, comme par exemple dans les jeux vidéo et deviennent par ailleurs un outil pour des applications scientifiques telles que la simulation chirurgicale.

Ces images de synthèse reproduisent la réalité de telle sorte que l'utilisateur ou le spectateur ne peut les discerner lorsqu'elles sont introduites dans un environnement réel. De même, le mouvement "synthétique" de certains éléments naturels et vivants, issus de l'imaginaire par exemple, est tellement plausible qu'il est difficile de croire qu'ils sont irréels. L'obtention d'un tel réalisme est aujourd'hui une demande des spectateurs, des artistes, des utilisateurs, mais est également un défi palpitant pour les scientifiques.

Certaines animations réalistes sont difficiles à concevoir manuellement. La capture de mouvement est un procédé qui vient pallier cette difficulté en numérisant des mouvements réels, par exemple humains. Cependant cette technique a ses limites. Dans certains cas, la physique ou la description formelle de la nature, est la seule méthode qui permet de reproduire ces animations complexes, dites alors animations basées-physique.

Cependant les modèles physiques ont aussi leurs limites. De plus, l'intégration des équations dans un ordinateur n'est pas sans problèmes. En effet, le processus dit de discrétisation engendre des erreurs d'approximations qui mettent en péril l'ensemble de la simulation. Les équations complexes nécessitent notamment des outils

sophistiqués pour les résoudre qui entraînent des temps de calculs importants que la communauté scientifique essaye par tous les moyens de réduire.

Cadre de la thèse

Pourquoi chercher à reproduire l'interaction entre un fluide et une membrane ? Les fluides sont présents dans notre quotidien sous différentes formes, par exemple de l'eau, de l'air ou encore de la fumée. Leur présence dans des univers virtuels plus vrais que nature est donc indispensable. De nos jours, la reproduction réaliste de ces phénomènes n'est possible qu'en ayant recours à l'animation basée-physique. Des scientifiques ont mis au point des modèles géométriques et dynamiques performants qui simulent avec réalisme le comportement des fluides. Mais au delà de ces modélisations, un phénomène spectaculaire, très courant dans notre quotidien, est l'interaction entre un fluide et une structure. La reproduction de ce phénomène n'est pas toujours bien maîtrisée et constitue un axe de recherche primordial en animations basées-physique. Par ailleurs, très peu de recherches ont été consacrées sur l'interaction entre un fluide et des membranes, rigides ou déformables, tels qu'une feuille de papier, une nageoire de poisson ou bien un tissu. De plus, en informatique graphique, les objets volumiques sont généralement représentés uniquement par leur bord ou surface qui est par nature sans épaisseur, donc comme une membrane. L'interaction entre un fluide et des membranes est par conséquent un sujet qui ne manque pas d'applications.

En synthèse d'images, un fluide fait généralement référence à seulement deux états de la matière, les états liquides et gazeux, l'état solide étant généralement modélisé différemment. Pourquoi cette différence de traitement ? La matière est géométriquement et numériquement représentée par deux approches qui peuvent se schématiser par un jeu d'échec. La première approche consiste à observer le mouvement de la matière, d'un point de vue du jeu d'échec, c'est l'échiquier qui constate le déplacement des pièces. La deuxième approche consiste à échantillonner ou discrétiser la matière et, ces échantillons appelés particules, représentent les pièces du jeu d'échec, sans plateau. Ces deux approches sont respectivement intitulées Eulériennes et Lagran-

giennes et ont respectivement leurs avantages et inconvénients. À titre d'exemple, le modèle Eulérien traite sans difficultés les changements de topologie et de plus offre de bonnes performances en terme de temps de calculs. C'est le modèle de choix pour représenter les fluides. Le modèle Lagrangien avec maillage a l'avantage de disposer explicitement d'une surface. C'est le modèle de choix pour représenter des solides ou structures déformables sans changement de topologie, donc des structures déformables autres que les fluides. Un modèle plus récent, intitulé Lagrangien sans maillage, profite des avantages des deux approches précédentes. C'est un modèle "en pleine croissance" qui dispose d'atouts majeurs, avec néanmoins certains inconvénients, mais il s'est imposé comme modèle de référence pour représenter les fluides dans les applications interactives.

L'interaction entre un fluide et une structure déformable consiste à échanger des forces lorsqu'il y a un contact. Cette échange de forces, qui doit respecter les propriétés physiques, n'est pas simple à mettre en oeuvre car les modèles géométriques des fluides et des structures déformables sont généralement distincts. Une autre difficulté s'ajoute lorsque les structures déformables sont sans épaisseur, comme les membranes, car dans ce cas la localisation spatiale et temporelle des contacts nécessite des traitements spécifiques et ce, pour les deux modèles géométriques.

La reproduction réaliste de ces interactions requiert l'utilisation de modèles dynamiques basés-physique. À cause des équations complexes à résoudre et de la fine discrétisation du domaine de simulation nécessaires pour que les fluides et les objets aient un comportement plus vrai que nature, les temps de calculs sont longs par image et par conséquent pour l'animation finale. Réduire ces temps de calculs a donc un aspect économique non-négligeable. De plus, les applications interactives disposent d'un très bref délai pour calculer le mouvement d'un objet. Les modèles d'optimisations spatiales et temporelles sont donc les bienvenus dans les animations basées-physique.

Problématique

Le mouvement de le matière est dicté par les forces environnantes. Un déplacement est un changement de position dû à une vitesse et une accélération non-nulles

qui, d'après le principe fondamental de la dynamique, sont dépendantes des forces. Cependant, l'approximation numérique du passage de l'accélération à la vitesse et de la vitesse à la position, ce qui est intitulé intégration temporelle (car il y a une relation de variation dans le temps), engendre des problèmes de calculs. L'erreur introduite par l'approximation numérique à une certaine échelle empêche l'obtention du résultat escompté, ce qui est appelé instabilité. Afin d'éviter ces problèmes d'instabilité, la contrainte d'échelle ou de pas d'intégration temporelle doit être respectée pour que les calculs convergent. Malheureusement souvent en pratique, cette contrainte est telle que les temps de calculs résultants sont importants. Afin de les réduire, cette contrainte sur le pas d'intégration temporelle doit être supprimée ou amoindrie.

J'ai choisi compte tenu de ses atouts indéniables et de son originalité, de représenter les fluides par un modèle Lagrangien sans maillage, donc par des particules. Cependant ce modèle nécessite d'effectuer des calculs en fonction des particules voisines, mais puisqu'il n'y a pas de relation d'adjacence entre les particules ou de topologie proprement dite, le voisinage doit être déterminé à chaque déplacement du fluide. Cette détermination spatiale est coûteuse en terme de temps de calculs et puisqu'elle est très sollicitée, son coût doit être minimisé. Les structures Lagrangiennes qui disposent d'un maillage sont souvent décomposées en triangles, pour des raisons de simplicité et d'efficacité. La détection des contacts entre les modèles Lagrangiens avec et sans maillage se résume donc à des évaluations de distances particules-triangles. Cependant, une reproduction réaliste de la dynamique des objets requiert un nombre important de particules et de triangles et puisque le nombre de ces évaluations est proportionnel au nombre d'éléments dans une scène, la détection des collisions est un processus coûteux en terme de temps de calculs. De plus, cette localisation est non-seulement dépendante de l'espace, généralement en trois dimensions en synthèse d'images, mais aussi du temps, ce qui complexifie les traitements. Il est alors judicieux, indispensable même dans beaucoup de cas pratiques, de mettre en place un modèle d'optimisation efficace pour améliorer les performances. Il est important de souligner que la détection de collisions, même optimisée, est le deuxième processus le plus coûteux d'une simulation, en terme de temps de calculs, devancé de peu par la résolution des collisions.

À cette détection de collisions en quatre dimensions, non-triviale, s'ajoute la réaction de contacts, c'est-à-dire l'échange des forces. Pour que l'interaction ait un comportement (visuellement) réaliste, cet échange doit respecter les lois de conservation de la physique. Mais lorsque les structures géométriques sont distinctes, que leurs forces appliquées sont modélisées différemment et qu'il y a plusieurs contacts en simultané, cet échange se complexifie.

Deux phases se distinguent dans la conception d'images virtuelles. La première, intitulée modélisation, s'effectue en deux temps : le premier temps consiste à définir la géométrie de la matière, ou des objets ; le deuxième temps consiste à définir la dynamique de la matière, c'est-à-dire son mouvement. À ce stade, les images de synthèse ne sont que des nombres interprétables par un ordinateur. C'est lors de la deuxième phase, le rendu, que ces données sont transformées en couleurs, c'est-à-dire en images compréhensibles par l'oeil humain. Pour ce faire, des calculs d'éclairement sont effectués sur la surface des objets, surface généralement consituée de triangles. Les structures Lagrangiennes sans maillage ne disposent pas de surface explicite, c'est-à-dire constituée de triangles, mais d'une surface implicitement définie autour des particules en bordure. Cette surface implicite est transformée en triangles à l'aide d'un procédé intitulé extraction de surface. La distance séparant les particules des triangles peut engendrer des artefacts visuels lorsqu'ils sont à proximité d'une membrane car l'extraction de surface, sans contraintes, produit des triangles qui passent temporellement au travers de la membrane.

Apports de mes travaux

Je me suis intéressé à l'interaction entre un fluide, plus particulièrement un liquide et des membranes déformables. Dans le but de simplifier les calculs des interactions, j'ai choisi d'utiliser le même modèle géométrique Lagrangien pour le fluide que pour les membranes. Une différence réside cependant, car les membranes sont définies comme des structures Lagrangiennes avec maillage et les fluides comme des structures Lagrangiennes sans maillage. Hormis cette différence, les modèles géométriques sont identiques, c'est-à-dire des systèmes de particules.

Le déplacement d'une particule est obtenu en calculant son accélération en fonction des forces qui lui sont appliquées, puis en intégrant temporellement cette accélération pour déterminer les nouvelles vitesse et position engendrées. L'intégration est en pratique numériquement approximée par des méthodes intitulées méthodes d'intégration temporelle numérique. Il existe deux familles de méthodes d'intégration numérique. La première, dite explicite, est contrainte par le pas d'intégration à l'opposé de la deuxième, dite implicite, qui s'affranchit de cette dépendance temporelle au dépend d'une dissipation des forces. Une contrainte forte sur le pas engendre de longs temps de calculs, mais cette contrainte est en fonction de l'application et donc le choix de la méthode d'intégration dépend également de l'application. Des travaux se sont intéressés à la stabilisation de la première et à l'augmentation de la précision de la deuxième. Je propose une nouvelle méthode, expérimentée sur les structures Lagrangiennes et plus précisément sur des systèmes masses-ressorts, qui vise la stabilisation des schémas explicites par l'application d'un filtrage qui propage les forces des particules dans un proche voisinage. Cette propagation permet de réduire, sous certaines conditions, les erreurs introduites par l'approximation des calculs d'intégration. Les résultats montrent que la restriction sur le pas d'intégration est plus souple, ce qui a pour effet, en pratique, de réduire les temps de calculs de l'ordre de 20% en moyenne. À l'aide d'analyses fréquentielles, la souplesse sur le pas est quantifiée.

L'optimisation de la détection de collisions est un processus qui vise à réduire le coût de la détermination de paires d'éléments potentiellement en contact. Je propose, dans ce même objectif, une structure accélératrice de subdivision spatiale dynamique basée sur une table de hachage hiérarchique, qui affiche des temps de calculs en moyenne trois fois plus rapides par rapport au modèle d'optimisation couramment utilisé avec des fluides Lagrangiens. De plus, au moins deux modèles d'optimisation sont souvent mis en oeuvre pour accélérer la détermination du voisinage de particules, la détection de collisions et le processus d'extraction de surface. La structure proposée peut ainsi servir de structure accélératrice unifiée.

Je présente également un algorithme de réaction de contacts entre des particules d'un fluide Lagrangien et des membranes déformables qui, en utilisant la structure accélératrice exposée et par le biais d'approximations, est indépendant du pas d'inté-

gration temporelle et traite le cas d'impacts multiples.

Enfin, je propose une modification de l'algorithme d'extraction de surface, le *marching cube*, pour que la surface extraite à proximité de membranes n'engendre pas d'artefacts visuels dus à des triangles qui les traversent.

Organisation du mémoire

Le chapitre 2 introduit des notions physiques et mathématiques utiles pour la bonne compréhension des méthodes proposées dans les chapitres suivants, comme par exemple les équations de la dynamique des fluides.

Le chapitre 3 présente un état de l'art des techniques utilisées en synthèse d'images relatant les interactions entre un fluide et des structures. Par ailleurs, cette interaction requiert la connaissance des modèles géométriques et dynamiques des fluides qui sont également discutés dans ce chapitre.

La stabilisation de méthodes d'intégration explicite, appliquée aux systèmes masses-ressorts, est par la suite abordée dans le chapitre 4. Trois modèles figurent dans ce chapitre, un θ-schéma, une stabilisation par filtrage exponentiel et une stabilisation par un filtrage de Kalman.

La structure accélératrice basée sur une table de hachage hiérarchique est présentée dans le chapitre 5.

Le chapitre 6 détaille un algorithme de réaction de contacts. Dans ce chapitre figurent également une proposition de paramétrisation intuitive de coefficients hétérogènes de friction par le biais de cartes, ainsi qu'une discussion sur l'utilisation de cartes de hauteurs pour les collisions.

La modification du processus d'extraction de surface est détaillée dans le chapitre 7. Le chapitre 9, qui fait office d'annexes, présente le modèle dynamique des fluides utilisé, les *Smoothed Particles Hydrodynamics* et discute de l'implantation logicielle que j'ai effectuée.

Chapitre Deux

Bases

Ce chapitre introduit les notions de bases, notamment physiques et mathématiques, utiles pour la bonne compréhension du restant de cette thèse. La conservation de la quantité de mouvement et la mécanique des fluides sont abordées dans la section 2.1. Les deux modèles géométriques les plus couramment mis en oeuvre en synthèse d'images pour calculer la dynamique d'une structure sont comparés en section 2.2. Le déplacement de la matière est réalisé à l'aide d'une intégration temporelle et les méthodes les plus classiques sont présentés dans la section 2.3.

2.1 Mécanique continue

2.1.1 Loi de conservation

Le principe de conservation, rien ne se perd, rien ne se crée, exprime qu'une propriété mesurable d'un système physique isolé reste constante au cours de l'évolution de ce système. Dans cette thèse sont considérées uniquement la conservation de la masse, de l'énergie et de la quantité de mouvement (ou moment), propriétés essentielles pour la reproduction réaliste de déplacements. Les descriptions suivantes sont faites d'un point de vue Lagrangien, c'est-à-dire en suivant l'évolution d'une particule qui représente un volume infinitésimal de la matière. La seconde loi de mouvement d'Isaac Newton, qui décrit la trajectoire $\mathbf{x} = \mathbf{x}(t)$ d'une particule comme une fonction du temps, permet de conserver la quantité de mouvement :

$$\ddot{\mathbf{x}} = f(\mathbf{x}, \dot{\mathbf{x}}, t) \tag{2.1}$$

où $\dot{\mathbf{x}}, \ddot{\mathbf{x}}$ sont respectivement les dérivées première et seconde de $\mathbf{x}$ par rapport au temps et $f()$ est une fonction qui dépend du modèle physique employé. Cette équation différentielle du second ordre peut être transformée en un système d'équation différentielle du premier ordre :

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{v} \\ \dot{\mathbf{v}} = f(\mathbf{x}, \mathbf{v}, t) \end{cases} \tag{2.2}$$

où $\mathbf{v}$ représente la vitesse de la particule. Suite à des déplacements, le volume dV de la particule et sa densité ρ sont susceptibles de varier, contrairement à la masse totale $m = \rho dV$ qui ne changera pas. La conservation de la masse est exprimée comme suit :

$$\frac{D\rho}{Dt} = -\rho \bigtriangledown \cdot \mathbf{v} \tag{2.3}$$

où D/Dt est la dérivée matérielle, c'est-à-dire le taux de changement d'une quantité physique dans le temps :

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{v} \cdot \bigtriangledown \tag{2.4}$$

avec $\bigtriangledown \cdot \mathbf{v}$ représentant la divergence de la vitesse. La dérivée matérielle relie la description Lagrangienne à la description Eulérienne. Cette dernière évalue les changements de quantités physiques à partir de points fixes dans l'espace. La densité d'une particule d'une matière incompressible, comme par exemple de l'eau sous certaines conditions, ne varie pas au cours du temps, et donc $D\rho/Dt = 0$. De ce fait, la conservation de la masse est réduite à l'équation suivante :

$$\bigtriangledown \cdot \mathbf{v} = 0 \tag{2.5}$$

2.1.2 Mécanique des fluides

La reproduction réaliste d'un fluide nécessite l'approximation des équations complexes décrivant sa dynamique dans un espace finement discrétisé. La plupart des

techniques issues de la dynamique des fluides résolvent les équations de Navier-Stokes (NS) ou leurs dérivées. Ce sont des équations différentielles partielles non-linéaires qui nécessitent pour leurs résolutions des outils mathématiques sophistiqués :

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{v} + \mathbf{f} \tag{2.6}$$

$$\nabla \cdot \mathbf{v} = 0 \tag{2.7}$$

où ν est la viscosité du fluide, p est la pression, $\mathbf{f}$ représente les forces extérieures et $\mathbf{v}$ est le champ de vitesses en trois dimensions (un raccourci d'écriture est utilisé pour $\nabla^2 = \nabla \cdot \nabla$). L'équation 2.6 exprime la conservation de la quantité de mouvement. Elle est équivalente à la relation fondamentale de la dynamique, c'est-à-dire à la seconde loi de mouvement décrite dans la section précédente. Les termes à droite représentent la somme des forces, où respectivement :

- $-\nabla p/\rho$ est la force de pression ;
- $\nu \nabla^2 \mathbf{v}$ est la force de viscosité ;
- $\mathbf{f}$ représente les forces extérieures telles que la gravité et les forces de contact.

Les termes à gauche symbolisent l'accélération qui s'exprime de deux façons différentes :

- L'approche Lagrangienne consiste à suivre l'évolution de particules de fluides, l'accélération est par conséquent la dérivée "particulaire" de la vitesse, i.e. $d\mathbf{v}/dt$.
- L'approche Eulérienne consiste à évaluer, en des positions stationnaires, les dérivées partielles des vitesses, i.e. $\partial \mathbf{v}/\partial t$, qui représente l'accélération locale mais aussi un terme d'advection, $(\mathbf{v} \cdot \nabla)\,\mathbf{v}$.

La conservation de la masse est décrite par l'équation 2.7. Ces équations représentent des fluides non-Newtoniens tels que l'eau, c'est-à-dire qui disposent d'une viscosité qui n'est pas changeante au cours du temps (hormis un changement de phase), contrairement à la gélatine et le dentifrice par exemple qui sont des fluides dits Newtoniens.

2.2 Discrétisation spatiale

Différentes méthodes numériques ont été développées pour résoudre les équations de mouvement. Ces approches sont classées en deux catégories, Lagrangienne et Eulérienne, qui respectivement suivent et observent les propriétés de la matière. Par ailleurs, la description Lagrangienne peut être avec ou sans maillage. Les avantages et inconvénients de chacune sont discutés dans les sous-sections suivantes.

2.2.1 Méthodes Eulériennes et Lagrangiennes

Les méthodes Eulériennes évaluent, en des positions fixes, les propriétés de la matière et calculent leurs changements au cours du temps. L'approche Lagrangienne consiste à suivre les éléments de la matière. La figure 2.1 schématise cette différence.

Le domaine fixe de résolution, selon l'approche Eulérienne, offre l'avantage de pouvoir résoudre les équations avec stabilité et donc avec rapidité. De plus, les changements de topologie et les déformations extrêmes de la matière sont implicitement gérés, contrairement aux modèles Lagrangiens qui nécessitent dans de tels cas une adaptation de la discrétisation afin d'éviter des problèmes numériques. Cependant, cette approche Lagrangienne à l'avantage d'avoir une surface explicitement définie par les bords, contrairement au modèle Eulérien où des méthodes complexes sont mises en oeuvre pour suivre la surface. La difficulté de la définition des bords engendre des problèmes de localisation des interactions avec des objets. Par ailleurs, avec cette approche Eulérienne, la définition de points d'observations contraint la simulation à se dérouler dans un espace borné, ce qui n'est pas le cas avec un modèle Lagrangien.

Le choix du modèle dépend donc de la simulation envisagée. Généralement, les objets sont définis par une approche Lagrangienne, comme les systèmes masses-ressorts par exemple, car la surface est plus facilement obtenue. Les fluides sont par contre souvent de type Eulérien dû à leurs déformations extrêmes et à leur changement de topologie qui engendreraient un remaillage fréquent avec un modèle Lagrangien (d'où les modèles Lagrangiens sans maillage, voir la section suivante). Il existe plusieurs méthodes qui mélangent ces deux approches, par exemple la *Particle-In-Cell* (PIC)

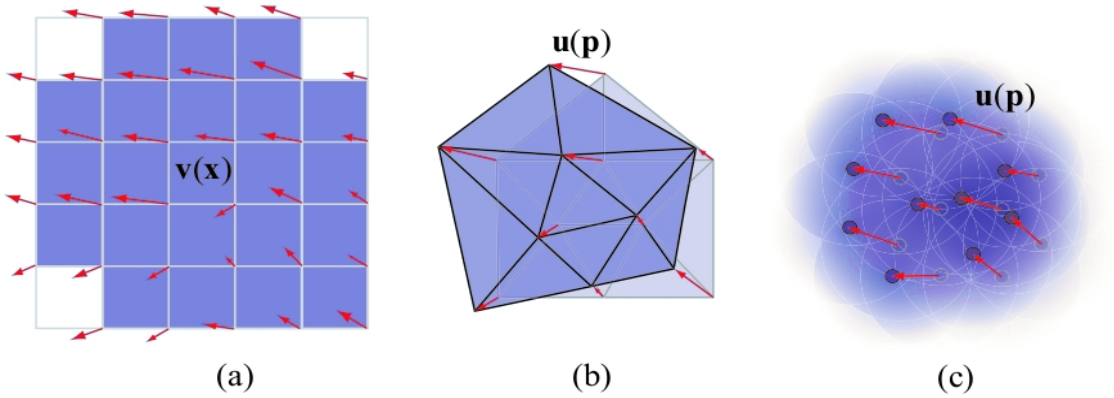

FIG. 2.1: La représentation Eulérienne (a) diffère du modèle Lagrangien qui est défini avec (b) ou sans (c) topologie. La matière est représentée par la couleur bleue, $\mathbf{v}(x), \mathbf{u}(x)$ représentent respectivement le champ de vitesses et le déplacement d'une particule. Dans l'image (c), la matière est définie dans un rayon autour de chaque particule.

de Francis Harlow [45] et la *FLuid-Implicit-Particle* (FLIP) de Jorg Brackbill et H. Ruppel [11] qui combinent une grille Eulérienne avec des particules. Les méthodes d'Olivier Génevaux *et al.* [37], de Eran Guendelman *et al.* [43] et de Frank Losasso *et al.* [63] combinent les interactions d'objets déformables définis selon un modèle Lagrangien avec un fluide Eulérien. Bryan Feldman *et al.* [31] ont utilisé la méthode *Arbitrary-Lagrangian-Eulerian* pour reproduire le mouvements de fluides contenus dans des objets déformables. Cette méthode permet de déplacer les points d'observations indépendemment du mouvement du fluide. L'extension de Bryan Klinger *et al.* [56] remaille à chaque itération le domaine pour mieux épouser les bordures des objets, remédiant ainsi à un des inconvénients des méthodes Eulériennes.

2.2.2 Méthodes Lagrangiennes avec et sans topologie

La plupart des méthodes Lagrangiennes échantillonnent un milieu continu en un nombre d'éléments finis connectés selon une carte topologique, ce qui est généralement appelé un maillage. L'utilisation d'un maillage n'est pas forcément un avantage, surtout dans le cadre de fluide ou la topologie est changeante au cours du temps. "Remailler" un objet est un procédé difficile, de plus c'est un processus coûteux en

terme de temps de calculs et en ressources mémoire. Par ailleurs, de grandes déformations engendrent des problèmes sévères de stabilité et de précision. Les méthodes sans maillage s'affranchissent de ces problèmes en ne conservant pas la connectivité, mais en approximant les propriétés de la matière à l'aide de fonctions d'interpolations. En revanche, ces fonctions d'interpolation sont dépendantes des particules voisines. La détermination de ce voisinage est un processus également coûteux en terme de temps de calculs et en ressources mémoire, plus de détails figurent dans le chapitre 5.

2.3 Discrétisation temporelle

Pour calculer numériquement des simulations, les objets sont transposés du milieu continu (réel) au milieu discrétisé (nécessaire en informatique), c'est-à-dire qu'un certain nombre d'échantillons sont sélectionnés pour représenter les objets. La dimension temporelle en animation est également échantillonnée puisque le temps ne peut être représenté de manière continue. La distance entre deux échantillons de temps, ou le pas d'intégration temporelle Δt, est un problème crucial en synthèse d'images : la stabilité. En effet, à cause de cette discrétisation, la résolution des équations n'est qu'une approximation du résultat et le cumul d'erreurs qui en résulte peut déstabiliser les calculs. Il est à noter que la taille de ce pas engendre entre autres des problèmes de précisions, mais en informatique graphique, la précision est moins importante que la stabilité car il est plus capital de suivre la cadence de l'animation plutôt que l'exactitude des mouvements.

Cette section aborde la résolution numérique des équations différentielles ordinaires (ODE) issues de la loi de conservation de mouvements. Une équation différentielle est une relation entre une fonction inconnue et ses dérivées, par exemple $\mathbf{x}_i(t)$ est un point en mouvement et $\dot{\mathbf{x}}_\mathbf{i}(\mathbf{x}_i, t)$ est la vitesse en ce point.

D'après la seconde loi d'Isaac Newton, soit $f(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{M}\ddot{\mathbf{x}}$:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= \mathbf{M}^{-1} f(\mathbf{x}, \mathbf{v}, t) \end{aligned} \tag{2.8}$$

où $\mathbf{M}_{i,i}, \mathbf{v}_i$ représentent respectivement la masse et la vitesse au point $\mathbf{x}_i$. Les

sections suivantes présentent trois familles de méthodes qui résolvent ces équations en fonction du temps.

2.3.1 Méthodes explicites

En considérant le schéma de différences finies suivant :

$$\mathbf{v}^{t+\Delta t} = \frac{\mathbf{x}^{t+\Delta t} - \mathbf{x}^t}{\Delta t}$$

Le système d'EDO devient :

$$\begin{pmatrix} \mathbf{x}^{t+\Delta t} \\ \mathbf{v}^{t+\Delta t} \end{pmatrix} = \begin{pmatrix} \mathbf{x}^t \\ \mathbf{v}^t \end{pmatrix} + \Delta t \begin{pmatrix} \mathbf{v}^t \\ \mathbf{M}^{-1} f(\mathbf{x}^t, \mathbf{v}^t) \end{pmatrix} \tag{2.9}$$

La nouvelle position est donc obtenue en calculant dans un premier temps la fonction $f(\mathbf{x}^t, \mathbf{v}^t)$, puis dans un deuxième la vitesse $\mathbf{v}^{t+\Delta t}$ et enfin $\mathbf{x}^{t+2\Delta t}$. Une attention particulière doit être portée sur Δt pour que la méthode converge.

La succession suivante d'équations démontre que le pas d'intégration est dépendant d'un coefficient λ, qui est, dans le cas des structures masses-ressorts, la raideur maximale du système :

$$\begin{aligned} \dot{y} &= -\lambda \cdot y, \text{ avec } \lambda > 0 \\ y^{t+\Delta t} &= y^t + \Delta t \cdot \dot{y}^t \\ y^{t+\Delta t} &= (1 - \lambda \cdot \Delta t) \cdot y^t \\ &\rightarrow |1 - \lambda \cdot \Delta t| < 1 \\ &\rightarrow \Delta t < \tfrac{2}{\lambda} \end{aligned} \tag{2.10}$$

En pratique, $\lambda >> 0$ (cas dit raide), donc $\Delta t << 1$. La figure 2.2 illustre une application de cette EDO.

Une autre méthode explicite, intitulé Runge-Kutta d'ordre quatre, est un peu plus stable et précise grâce à une interpolation entre quatre points intermédiaires. Cette

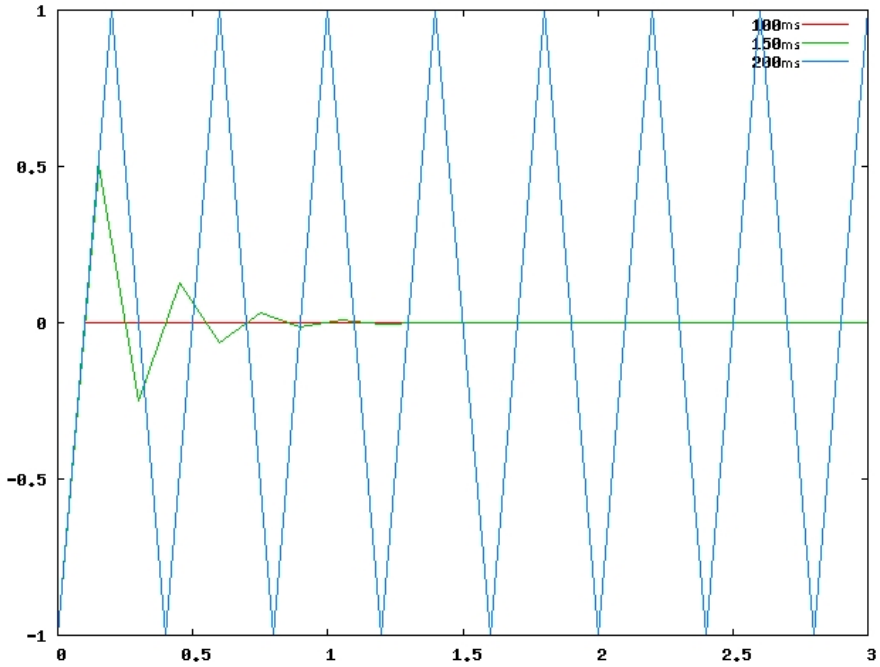

FIG. 2.2: Soit l'EDO suivante : $\dot{x} = -10x$ et $x_0 = -1$. Les trois courbes rouge (convergence immédiate), verte (convergence au bout d'une seconde) et bleue (phase d'oscillations avant la divergence) sont obtenues en résolvant l'EDO de l'équation 2.10 avec les pas d'intégration respectifs : 0.1s, 0.15s, 0.2s, ce dernier étant le pas limite, i.e. $2/\lambda$ soit 2/10 ici.

méthode est définie comme suit :

$$\begin{aligned} \mathbf{x}^{t+\Delta t} &= \mathbf{x}^t + \tfrac{\Delta t}{6} \cdot (\mathbf{k}_1 + 2 \cdot \mathbf{k}_2 + 2 \cdot \mathbf{k}_3 + \mathbf{k}_4) \\ \mathbf{k}_1 &= f(t, \mathbf{x}^t) \\ \mathbf{k}_2 &= f(t + \tfrac{\Delta t}{2}, \mathbf{x}^{t+\frac{\Delta t}{2}} \cdot \mathbf{k}_1) \\ \mathbf{k}_3 &= f(t + \tfrac{\Delta t}{2}, \mathbf{x}^{t+\frac{\Delta t}{2}} \cdot \mathbf{k}_2) \\ \mathbf{k}_4 &= f(t + \Delta t, \mathbf{x}^{t+\Delta t} \cdot \mathbf{k}_3) \end{aligned} \tag{2.11}$$

L'erreur est avec cette approche en $\mathcal{O}(\Delta t^5)$. Dans beaucoup de cas pratiques, la contrainte sur le pas d'intégration temporelle est telle que les simulations résultantes sont très longues à calculer. Une autre formulation, dite implicite, vient pallier cet inconvénient en s'affranchissant de cette contrainte sur le pas d'intégration.

2.3.2 Méthodes implicites

La différence entre les méthodes explicites et implicites réside dans le calcul de la nouvelle position/vitesse, i.e. l'état du système, qui est effectué respectivement à partir de l'état courant et à partir de l'état recherché. Calculer une inconnue à partir d'une inconnue est plus difficile, mais ces méthodes implicites ont la particularité de pouvoir traiter sans difficultés les systèmes dit raides. Dans de telles configurations, ces schémas sont bien plus efficaces que les schémas explicites.

La méthode implicite la plus couramment utilisée en synthèse d'images et la *Backward Euler*[1], qui est de la forme :

$$\begin{pmatrix} \mathbf{x}^{t+\Delta t} \\ \mathbf{v}^{t+\Delta t} \end{pmatrix} = \begin{pmatrix} \mathbf{x}^{t} \\ \mathbf{v}^{t} \end{pmatrix} + \Delta t \begin{pmatrix} \mathbf{v}^{t+\Delta t} \\ \mathbf{M}^{-1} f(\mathbf{x}^{t+\Delta t}, \mathbf{v}^{t+\Delta t}) \end{pmatrix} \tag{2.12}$$

L'évaluation de la vitesse nécessite la résolution d'un système linéaire, voir le chapitre 4 pour plus de détails.

Les équations suivantes démontrent que les méthodes implicites sont indépendantes du pas d'intégration temporelle :

$$\begin{aligned} \dot{y} &= -\lambda \cdot y, \lambda > 0 \\ y^{t+\Delta t} &= y^t + \Delta t \cdot \dot{y}^{t+\Delta t} \\ y^{t+\Delta t} &= \tfrac{y^t}{(1+\lambda \cdot \Delta t)} \\ &\rightarrow (1 + \lambda \cdot \Delta t)^{-1} < 1 \end{aligned} \tag{2.13}$$

Puisque $\Delta t \geq 0$, il n'existe donc pas de contraintes sur le pas d'intégration temporelle, ce qui fait des méthodes implicites les méthodes choisies pour de nombreux cas pratiques.

2.3.3 Méthodes IMEX

Les méthodes IMplicites-EXplicites, ou IMEX, ont pour but de profiter des avantages de chacune, i.e. de la simplicité des méthodes explicites et de la stabilité des

[1] c'est une *Backward Differential Formula* (BDF) du premier ordre, voir l'ouvrage de William Press *et al.* [83]

méthodes implicites. La plupart des approches IMEX décomposent les équations en deux termes, qui sont résolus respectivement par un procédé explicite et implicite. Une autre approche, proposé dans le chapitre 4, consiste à corriger le résultat des méthodes explicites par un procédé issu de la résolution des équations selon le modèle implicite. Par ailleurs, plus de détails sur les méthodes IMEX figurent dans le chapitre précédemment cité.

Chapitre Trois

État de l'art

Les images de synthèse sont très convoitées dans de nombreux domaines, tels que le cinéma, les jeux vidéo ou encore la publicité. Elles permettent, entre autres, de réduire les coûts de production en effectuant des simulations numériques qui disposent d'un retour d'informations visuels à la place d'expérimentations destructives par exemple, ou encore de produire des effets spéciaux réalistes ou irréalistes qui sont dans tout les cas inimitables par d'autres procédés. De plus, elles permettent la création de monde virtuels. Le scientifique, le spectateur ou bien encore le joueur sont souvent demandeurs d'un résultat visuel crédible, basé sur le monde réel. Par conséquent, le souci du réalisme est omniprésent dans les images de synthèse.

Dans un premier temps, les chercheurs se sont intéressés à la reproduction d'images réalistes par des procédés empiriques, puis dans un deuxième temps en augmentant ce réalisme en introduisant des propriétés physiques de la lumière et de la matière. Les animations ont une histoire identique. Le mouvement d'éléments fut en premier lieu gouverné par des procédés géométriques, où le réalisme des animations fait le fruit d'une paramétrisation de ces techniques, qui demande un travail image par image, minutieux et fastidieux, de la part de l'animateur. L'ajout de propriétés physiques du mouvement va offrir d'avantages de possibilités à l'utilisateur, dans le domaine de la création par exemple, mais aussi permettra de générer une animation par un procédé entièrement automatique.

Le reproduction de la dynamique des fluides est le type même de comportement difficile à animer manuellement. Depuis environ deux décennies, le mouvement des fluides en synthèse d'image est partiellement ou entièrement obtenu algorithmiquement et ce, grâce à l'ajout de notions physiques. La résolution des équations de la

dynamique des fluides est difficile et très coûteuse en terme de temps de calculs. Cependant, l'évolution de la puissance des ordinateurs a permis la résolution de ces équations en temps acceptable[1]. Aujourd'hui, la recherche se porte sur l'élaboration de modèles plus fiables, i.e. stables, tout en conservant une rigueur appropriée[2] et aussi sur l'interaction avec l'environnement.

Les simulations numériques scientifiques sont exigeantes en terme de précision et de rigueur où la physique est un élément essentiel. Cependant, dans ce domaine, les physiciens utilisent l'ordinateur pour effectuer des expérimentations, le retour visuel n'est certes pas aussi impressionnant que dans les production cinématographiques mais offre le résultat escompté. Beaucoup de méthodes utilisées en synthèse d'images ne sont pas rigoureuses et, à juste titre, ne sont employées qu'à des fins ludiques.

La puissance toujours grandissante des ordinateurs offre la possibilité d'obtenir des animations de plus en plus réalistes en complexifiant les modèles sous-jacents. Les fluides multi-phases, l'interaction avec des objets déformables, avec ou sans épaisseur sur de grandes étendues est un sujet d'actualité qui découle de l'évolution naturelle de l'ajout de réalisme dans les animations virtuelles. Une autre évolution, tout aussi inéluctable, est l'optimisation des modèles pour que les animations puissent être calculées en temps réel, pour par exemple qu'elles figurent dans des mondes virtuels interactifs.

Cet état de l'art recouvre les techniques utilisées dans les images de synthèse pour modéliser l'interaction entre un fluide et une structure déformable. Cette interaction est dépendante des modèles géométriques et dynamiques qui sont par conséquent abordés dans ce chapitre. Un très bon état de l'art des modèles géométriques et dynamiques des structures déformables, d'Andrew Nealen *et al.* [78], a été publié récemment et j'ai décidé de ne pas refaire ce travail dans cette thèse. J'invite donc le lecteur à lire cet article. Le restant de ce chapitre est divisé en quatre sections :

- la section 3.1 énumère les différents modèles géométriques de fluides existants ;
- la section 3.2 concerne l'application de la dynamique des fluides à l'informatique graphique ;

[1] le terme acceptable n'est pas bien définit, il dépend de la simulation envisagée.
[2] la rigueur est en fonction du domaine d'application.

– la section 3.3 évoque les différents modèles d'interactions entre des fluides et des objets déformables utilisés en synthèse d'images.

Cet état de l'art s'intéresse principalement à un fluide particulier sous sa forme liquide : l'eau. Cependant, des innovations intéressantes relatives à d'autres fluides, comme par exemple des liquides visqueux ou des gaz sont entre autres abordées. J'encourage la lecture de l'état de l'art de Neeharika Adabala et Swami Manohar [2] pour la classification des techniques de dynamique et de visualisation des fluides, mais aussi celui d'Andrés Iglesias [49] qui retrace historiquement les recherches relatant la modélisation et le rendu de l'eau.

3.1 Modèles géométriques des fluides

Trois modèles géométriques sont couramment utilisés en synthèse d'images pour représenter un fluide : le modèle Eulérien, le modèle Lagrangien sans maillage et des approches hybrides, qui sont respectivement détaillés dans les sections 3.1.1, 3.1.2 et 3.1.3.

3.1.1 Géométrie Eulérienne

Principe

La description de la matière, selon le modèle Eulérien, consiste à échantillonner un domaine dans un premier temps, puis dans une deuxième temps à attribuer une quantité d'informations à chaque échantillon. Cette description est comparable au jeu d'échec, ou le plateau représente le domaine et chaque case du damier symbolise un échantillon. Les pions se déplacent arbitrairement[3], c'est-à-dire, d'un point de vue Eulérien, la quantité de matière change d'une case à une autre.

En informatique graphique, la discrétisation du domaine, qui est abordée dans les sous-sections 3.1.1.1 et 3.1.1.2, est généralement effectuée deux et trois dimensions. Les échantillons représentent la géométrie du fluide que s'ils contiennent de la matière, sinon ils sont considérés comme des éléments vides. La plupart du temps, l'utilisateur

[3]je reconnais cependant l'existence de stratégies dans ce jeu.

détermine au préalable la quantité de matière qui sera utilisée lors de la simulation. Il est également possible de générer la matière par des fonctions mathématiques. Cependant, les modèles géométriques sous-jacents ne sont pas différents de ceux présentés dans les deux autres sections.

3.1.1.1 Discrétisation en 2D

Le procédé le plus simple et intuitif pour représenter un fluide est d'en prendre une photographie. Cette photographie, ou image générée algorithmiquement[4], est transformée en texture qui est par la suite appliquée sur un ensemble de triangles par exemples. Les gaz sont par nature volumiques et peuvent être visualisés en juxtaposant une série de textures 2D semi-transparentes pour simuler l'effet de profondeur. Ce procédé est très employé dans les jeux vidéo pour reproduire la fumée. Un avantage majeur de l'utilisation des textures 2D est la possibilité de tirer profit du support matériel ; dès lors les temps de calculs sont considérablement réduits. En visualisation de fluide 2D, une technique très répandue et exposée par Brian Cabral et Leith Leedom [15] est la *Line Integral Convolution* (LIC), où l'image est localement flouée selon une trajectoire définie par un champs de vecteur (par exemple de vitesse), donnant ainsi l'impression d'un écoulement de fluide. J'invite encore une fois le lecteur à se référer à un autre état de l'art, celui de Robert Laramee *et al.* [59] qui récapitule les techniques employées pour reproduire l'apparence d'un fluide dans une image.

3.1.1.2 Discrétisation en 3D

La discrétisation d'un domaine en trois dimensions, par exemple un cube, est comparable à un ensemble de plateau de jeux d'échecs empilés. Le procédé est donc exactement le même que dans la section précédente, avec cependant une dimension supplémentaire. Nick Foster *et al.* [34] et Jos Stam [92] ont été des pionniers dans ce domaine. Les auteurs utilisent une grille régulière tridimensionnelle pour reproduire le mouvement d'un fluide, selon des modèles dynamiques sophistiqués. Frank Losasso

[4]la résolution des équations de la dynamique des fluides par méthodes spectrales, outre leur rapidité, permet d'obtenir des textures qui se répètent, voir les travaux de Larry Yaeger *et al.* [114] et de Jos Stam [92].

et al. [62] ont mis en place une structure de subdivision de la grille par arbre octal pour réduire l'occupation mémoire et les temps de calculs.

Les cellules de la grille sont pour la plupart du temps cubiques et épousent difficilement les bordures des objets. Des effets de crénelage sont visibles autour des obstacles ou des bordures. Pour pallier ce problème, il est tout a fait possible de discrétiser très finement l'espace du fluide, c'est-à-dire d'avoir des petites mais plus nombreuses cellules. Cependant, les temps de calculs sont en fonction du nombre de cellules et la discrétisation a donc ses limites en pratique. Une autre solution, proposée par Bryan Feldman *et al.* [30], consiste à subdiviser le domaine avec des tétraèdres et de ce fait, les cellules épousent avec plus de facilité le contour d'objets complexes. Un exemple de leurs résultats est illustré dans la figure 3.1.

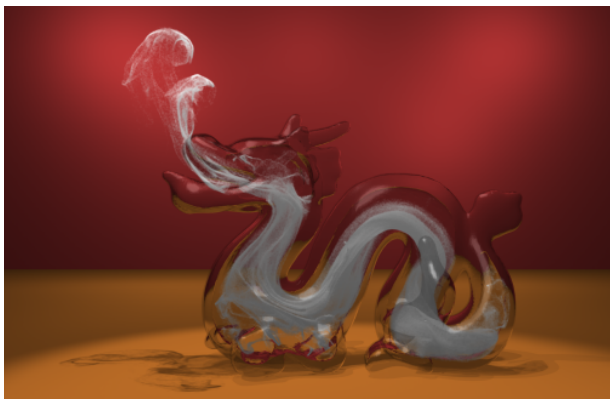

FIG. 3.1: La méthode de Bryan Feldman *et al.* [30] permet aux fluides de mieux épouser des formes géométriques complexes.

3.1.1.3 Génération par des fonctions mathématiques

Les fractales et les fonctions pseudo-stochastiques peuvent déterminer le positionnement de la matière dans un espace donné pour représenter un fluide par exemple. Ce procédé est détaillé dans l'ouvrage de David Ebert *et al.* [23]. Les fractales, obtenus par un procédé déterministe ou stochastique, simulent des phénomènes naturels tels que l'eau, les nuages ou encore les explosions (voir les travaux de Ken Musgrave[5]).

[5]http ://www.kenmusgrave.com

Ken Perlin [80] utilise une fonction mathématique intitulée "turbulence" pour reproduire des surfaces d'eau et autre phénomènes naturels. Cette fonction de turbulence est définie par la somme de fonctions de bruit caractérisés par différentes fréquences et amplitudes. Sébastien Thon [100] propose un modèle hybride fonction pseudo-stochastique/particule pour ajouter des détails d'agitation de vagues sur des surfaces d'eau. Cette surface est obtenue par une triangulation de particules qui simulent physiquement l'écoulement du fluide. L'objectif temps réel souhaité par l'auteur impose l'utilisation d'un nombre restreint de particules et par conséquent la surface du fluide est lisse, i.e. peu détaillée. Pour pallier cet inconvénient, du *bruit de Perlin*, défini dans un espace autour de chaque particule, permet d'imiter ces détails perdus en temps interactifs.

Robert Bridson *et al.* [13] ont ajouté au *bruit de Perlin* la notion de turbulence et d'incompressibilité pour permettre la représentation d'un fluide à partir d'une méthode procédurale. La dynamique du fluide est alors obtenue en temps interactifs pour des simulations en deux et trois dimensions (en fonction de la taille du domaine de résolution bien sûr).

3.1.1.4 Avantages et inconvénients

Le modèle géométrique Eulérien a l'avantage de ne considérer que la variation de la matière qui peut être arbitraire, voir extrême. Ainsi, le changement de topologie est implicitement pris en compte. Un avantage majeur pour la calcul de la dynamique des fluides est que, de par la nature stationnaires des points d'observations, les équations de mouvement sont résolues avec stabilité et par conséquent, avec rapidité. En revanche, puisque ces points d'observation sont fixes, le déroulement de la simulation est par conséquent dans un espace bornée. De plus, la détermination de la surface du fluide, au sein d'une case ou cellule, demande la mise en place de méthodes complexes.

3.1.2 Géométrie Lagrangienne sans maillage

Les méthodes Lagrangiennes, avec maillage, définissent explicitement la surface qui est dans le cadre des fluides changeante au cours du temps. Par exemple, l'agi-

tation d'un récipient contenant de l'eau engendrera une transformation de la surface du fluide, avec peut être même des projections de gouttelettes. Les changements de topologie nécessitent un remaillage de l'objet ce qui est un processus difficile. Les méthodes Lagrangienne sans maillage ont pour but de pallier ce problème en s'affranchissant de la connectivité, c'est-à-dire en ayant des éléments indépendants. La géométrie du modèle est couramment dite floue, et pour représenter de telles géométries, William Reeves [86] a introduit les systèmes de particules. Simples dans leur conception mais très efficaces dans leur utilisation, ils sont abondamment employés en synthèse d'images. En reprenant l'exemple de l'échiquier, les particules Lagrangiennes représentent les pions sans damier.

Principe

Discrétiser la surface ou le volume du fluide par un ensemble d'éléments infiniment petits. Chaque élément, intitulé particule, représente une partie du volume qui se déplace librement. Puisque il n'y a pas de relations avec le voisinage, il n'y a donc pas non plus de topologie proprement dite. Chaque particule véhicule des informations telles que sa couleur ou encore sa position. Les équations de la dynamique des fluides servent à décrire le mouvement de chaque élément. Toutes les interactions possibles, par exemple inter-fluides et fluides-structures, sont déterminées à partir de ces particules.

3.1.2.1 Particules simples

Cette méthode permet d'appliquer intuitivement des comportements physiques. Les particules sont donc le modèle de choix pour animer un fluide.

Pour la visualisation, trois techniques sont couramment utilisées : le *motion blur* simulant l'effet de rémanence utilisé par Karl Sims [91] (voir la figure 3.2) et par Pierre Rousseau *et al.* [87], la triangulation avec par exemple l'algorithme des *marching cubes* de William Lorensen et Harvey Cline [61] et par des *splats* comme dans les travaux de Bart Adams *et al.* [8].

La triangulation est une méthode qui crée une surface autour des particules. Cette

surface est reconstruite à chaque déplacement de l'ensemble des particules. Sans ce processus, les particules ne possèdent pas de surface et donc pas d'orientation. Par conséquent il est impossible de les soumettre à des modèles réalistes d'illumination. Pour pallier ce problème, les particules sont considérées comme des petites surfaces.

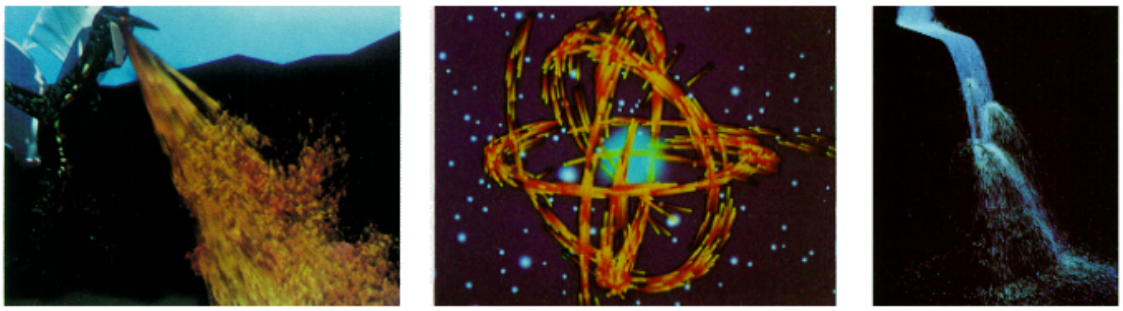

FIG. 3.2: Exemple de particules simples (images de Karl Sims [91]).

3.1.2.2 Particules surfaciques - surfels

Dans cette approche une surface minuscule est associée à chaque particule. L'orientation de la surface et les attributs relatifs à la lumière vont permettre d'illuminer ces éléments. Les particules surfaciques de Jarke Wijk [105] sont considérées comme des éléments ayant chacun un vecteur normal et des coefficients d'illumination pour les deux côtés de la facette. Cependant, les surfaces sont minuscules, des artefacts dus à l'aliasing temporel surgissent et pour éviter cela, l'auteur utilise un filtre passe-bas qui élimine les hautes fréquences. Le rendu de ces éléments fait partie de la discipline *point based graphics*, ou le rendu basé point (voir les travaux de Richard Keiser [55] par exemple).

3.1.2.3 Avantages et inconvénients

Les méthodes Lagrangiennes sans maillage traitent sans difficultés les changements de topologie et la surface est définie par les particules situées sur les bords. Cette méthode profite des avantages des méthodes Lagrangienne et Eulérienne. L'inconvénient principal est que la dynamique du fluide et la détermination de la surface sont cal-

culées par des fonctions d'interpolations qui nécessitent la connaissance du proche voisinage, ce qui est un processus coûteux en terme de temps de calculs. Le chapitre 5 propose un modèle qui accélère ce calcul dans le cadre d'une interaction avec une structure.

3.1.3 Géométrie hybride

Principe

L'objectif de ces méthodes est de tirer profit des avantages des modèles Eulérien et Lagrangien en les associant. Par exemple, une grille est utilisée pour effectuer les calculs de la dynamique des fluides et puisqu'il est difficile de suivre la position de la surface du fluide avec les cellules uniquement, des particules de marquage sont introduites pour délimiter cette surface.

3.1.3.1 Mise en oeuvre

Les méthodes hybrides présentées sont classées en trois catégories et pourtant, les deux premières proposent des modèles géométriques similaires. Puisque ces approches n'ont pas les mêmes objectifs, j'ai donc choisi de les dissocier. La dernière catégorie propose une discussion sur le matériel dédié utilisé en animations basées-physique.

3.1.3.2 Grille et particules

Cette approche consiste à suivre des particules, qui véhiculent des informations, et d'utiliser la grille pour effectuer des calculs d'affichage par exemple. Jos Stam et Eugène Fiume [93] ont utilisé des *blobs* pour représenter les phénomènes de feu et de fumée. Un *blob*, ou *metaball* sont des particules qui ont l'originalité d'être traitées comme des fonctions de densité ou de potentiels. La valeur du potentiel est évaluée à l'aide de la grille.

Nick Foster, Dimitris Metaxas [35], Neeharika Adabala et Swami Manoharr [1] ont introduit des particules sans masse dans la grille pour calculer la densité du fluide. À chaque particule est attribuée une valeur arbitraire de densité, paramétrable par

l'utilisateur pour simuler l'effet souhaité. La visualisation est réalisée en traçant des rayons dans la grille, dans le but d'obtenir l'opacité du nuage de particules vu par l'utilisateur, qui est par ailleurs une méthode similaire à celle proposée par David Ebert et Richard Parent [24].

3.1.3.3 Surfaces /Particules

Les surfaces implicites sont des fonctions mathématiques utilisées pour définir la surface des objets. Si la fonction évaluée à partir d'un point retourne une valeur inférieure, égale ou supérieure à zéro, cela indique que ce point est respectivement à l'intérieur de, sur ou à l'extérieur de l'objet.

Les surfaces implicites, grâce à leur continuité, offrent actuellement le rendu le plus réaliste et sont très utilisées dans le domaine cinématographique par exemple. La méthode *particle level set* est le modèle de référence, c'est une surface implicite qui préserve son volume après diverses déformations et ce, grâce à l'ajout de particules. Il est également possible de suivre l'évolution d'une surface explicite par le biais de particules comme récemment proposé par Adams Bargteil *et al.* [7].

3.1.3.3.1 Level Set *et al.* Cette méthode est une fonction implicite qui, par une évaluation de distance signée, identifie le bord (l'interface), l'intérieur et l'extérieur du fluide. Ce bord est généré par un isocontour lissé à partir d'un champ scalaire. Les *level set* possèdent des propriétés intéressantes comme des facteurs géométriques intrinsèques faciles à calculer, par exemple le vecteur normal. Cependant elles souffrent d'une perte de volume visible lorsque le fluide est animé. Ce problème de perte de volume est dû au suivi de la mise à jour des vitesses par la surface implicite. Pour pallier ce problème, Nick Foster et Ronald Fedkiw [33] ont introduit un procédé hybride qui ajoute aux surfaces implicites des particules explicites pour suivre la surface, préservant ainsi partiellement le volume. Une capture d'écran d'une de leurs expérimentations est illustrée dans la figure 3.3. Cette méthode est appelée *Particle Level Set* (PLS). Les extensions de Doug Enright *et al.* [25, 26] permettent de conserver totalement le volume (voir la figure 3.4). Cette méthode a été premièrement utilisé pour le film "Shrek", depuis elle est devenue très populaire dans le domaine cinéma-

tographique, elle figure par exemple dans les films suivants, pour ne citer qu'eux : "Terminator 3" [85], "Pirates des Caraïbes","Le Jour D'après" et "Harry Potter et la Chambre des Secrets". Les PLS et dérivées sont sujettes à de nombreuses publications et ouvrages.

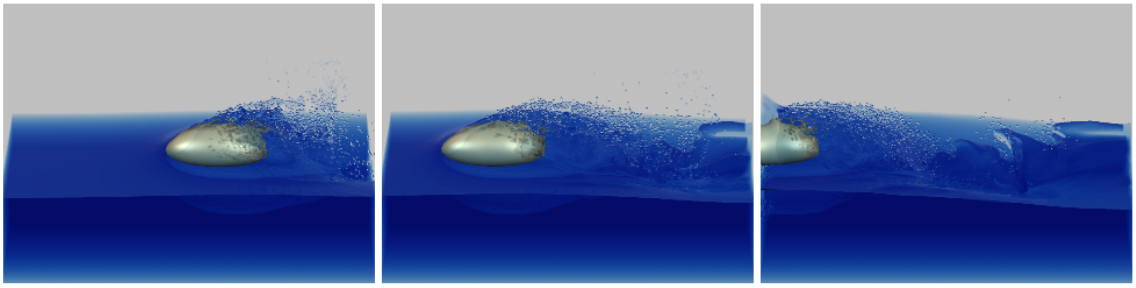

FIG. 3.3: Exemple d'utilisation des PLS pour les fluides.

La visualisation des PLS est effectuée soit en extrayant la surface par la méthode des *marching cubes* de William Lorensen et Harvey Cline *et al.* [61] par exemple, soit en calculant directement les intersections rayon/surface avec du *ray marching*.

Pour optimiser l'espace mémoire utilisé lors de la définition de la surface, Ben Houston *et al.* [48] proposent de compresser la structure de données par un procédé hiérarchique de compression RLE. L'espace mémoire économisé est de l'ordre de 75% en moyenne, permettant ainsi la représentation d'objets complexes en *level set* (LS) sur des ordinateurs grand public. Un exemple de leurs résultats est illustré dans la figure 3.5[6].

3.1.3.3.2 Surface "semi-Lagrangienne" Adams Bargteil *et al.* [7] décrivent la surface d'un liquide Eulérien explicitement, i.e. avec des polygones. L'évolution de cette surface est effectuée à partir d'une fonction implicite de distance signée similaire aux *level set*. Cette fonction implicite permet de traiter sans difficultés les changements de topologie, tandis que l'ensemble polygonal permet de conserver des propriétés telles que les coordonnées de texture (voir la figure 3.6), ce qui est par ailleurs impossible

[6]une vue d'ensemble des structure de données mis en oeuvre pour stocker les LS figurent dans http ://en.wikipedia.org/wiki/Level_set_(data_structures).

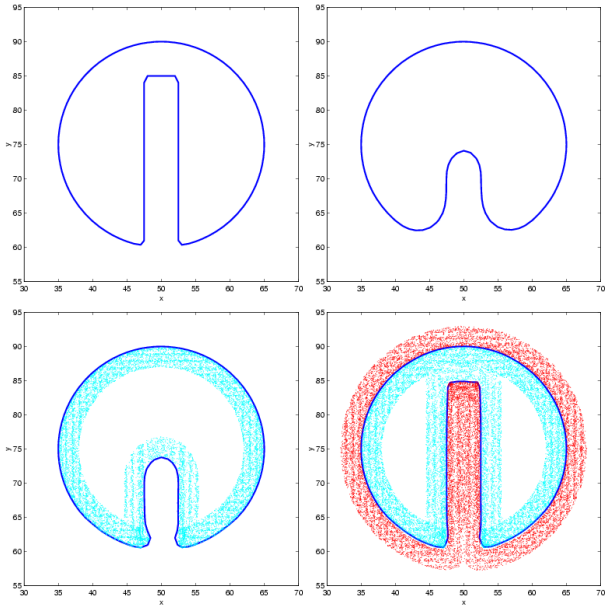

FIG. 3.4: Le but de cette expérience et de montrer la conservation du volume grâce à l'introduction de particules. Une surface LS (*haut gauche*) subit une rotation de 360° (*haut droite*). Clairement le volume n'est pas préservé. *Bas gauche :* avec des particules à l'intérieur uniquement, le volume est un peu mieux conservé. *Bas droite :* avec des particules à l'intérieur et à l'extérieur, le volume est conservé.

avec les PLS. Visuellement la surface obtenue est similaire à celle calculée par des PLS avec des temps de calculs comparables.

3.1.3.4 Discussion sur le matériel dédié

Les processeurs des cartes graphiques (GPU) sont conçus pour effectuer des tâches de visualisation. Dû à leur limitation d'instructions, ces processeurs sont beaucoup plus efficaces que le(s) processeur(s) central(aux) (CPU) pour des opérations spécifiques. Par conséquent l'accélération graphique matériel est très sollicitée en synthèse d'images. Sur ce matériel dédié sont adaptés toutes sortes d'algorithmes, allant de la dynamique des fluides en passant par les collisions et pour l'affichage évidemment. Bien sûr les algorithmes sont exécutés plus rapidement, mais ils sont modifiés, voir "bricolés", pour répondre aux contraintes imposées par les cartes graphiques. Par exemple, la seule structure de données est la texture, c'est à dire une carte. Les sys-

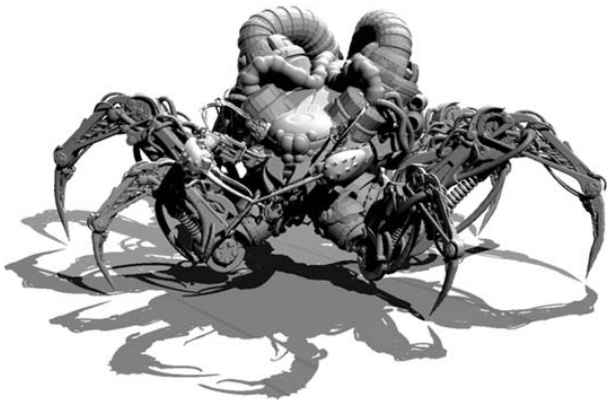

FIG. 3.5: La taille de la boite englobante de ce modèle est de 1691x1223x839, soit 1,7 milliards de voxels. L'espace mémoire requis après compression RLE est de 229,6 Mo.

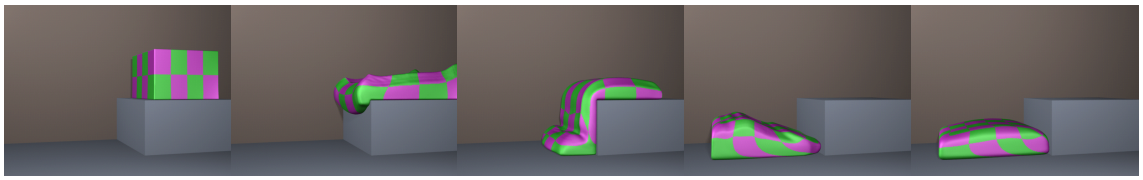

FIG. 3.6: La définition explicite de la surface permet de conserver tout au long de la simulation les coordonnées de texture.

tèmes de particules sont par conséquent assemblés dans ces cartes. Les algorithmes mis en place pour le calcul de la dynamique des fluides ne sont pas très novateurs, mais sont ajustés pour être en accord avec le fonctionnement de la carte graphique. Tous les algorithmes ne sont pas encore adaptés et adaptables. Certains, comme Matthias Müller *et al.* [75, 76], utilisent le CPU pour les calculs physiques puis sollicitent le GPU pour afficher le fluide, i.e. pour en extraire sa surface. La vitesse bridée du bus de communication entre le CPU et le GPU est de plus en plus le goulot d'étranglement des applications. C'est une des raisons principales pour laquelle beaucoup de travaux cherchent à exploiter uniquement le GPU. L'évolution des cartes graphiques offre de plus en plus de souplesse dans l'intégration des algorithmes. Jusqu'alors, les langages bas niveau étaient les seuls disponibles pour piloter les cartes. Des langages nouveaux et de plus haut niveau tels que Sh[7] ou CUDA[8] permettent une plus grande

[7] http ://libsh.org
[8] http ://developer.nvidia.com/object/cuda.html

abstraction par rapport au matériel.

La modélisation basé physique est de plus en plus populaire, de part sa présence dans les productions cinématographiques et dans les jeux vidéos. Les cartes graphiques sont suffisamment sollicités par les calculs d'affichage pour s'occuper en plus du traitement des modèles physiques. C'est pour cette raison que des processeurs supplémentaires dédié aux calculs physiques commencent à voir le jour. Ces processeurs intitulés *Physical Processing Unit* (PPU) sont pour la physique ce qu'est la carte graphique pour l'affichage. La PlayStation 2 de Sony dispose d'une unité de calcul -*Emotion Engine*[9]- servant en partie au calcul de procédés physique. Ageia[10] propose un PPU qui communique directement avec le CPU **et** le GPU. Les consoles nouvelle génération commencent à utiliser du matériel dédié aux calculs physiques, dans la continuité de l'intégration de la parallélisation, d'autant plus que les CPU sont actuellement bridés en fréquences. Par exemple, la PlayStation 3 dispose de 7 processeurs CELL[11] et la XBOX 360[12] est munie d'un tricoeur.

3.1.3.5 Avantages et inconvénients

Différents modèles sont assemblé dans le but de pallier les problèmes de chacun. Une surface implicite combinée avec des particules offre des résultat visuels comparable à la réalité. C'est pour cette raison que ce modèle fait office de référence. Cependant, les temps de calculs sont importants et sont en fonction du nombre de marqueurs/voxels. La parallélisation des calculs, bien que difficile dans ce cas, commence à voir le jour.

L'utilisation d'un carte graphique permet d'accélérer certains calculs et en adaptant les algorithmes, des simulations peuvent aujourd'hui se dérouler en temps interactifs. Cependant les cartes graphiques sont actuellement limitées dans leurs possibilités, mais puisque ce marché est très actif, ces limitations ont tendance à disparaître.

[9] http ://en.wikipedia.org/wiki/Emotion_Engine
[10] http ://www.ageia.com
[11] http ://fr.wikipedia.org/wiki/Cell_(processeur)
[12] http ://fr.wikipedia.org/wiki/Xbox_360

3.2 Modèles dynamiques des fluides

Le calcul des mouvements de fluides soumis à l'action de forces mécaniques quelconques s'intitule la dynamique des fluides. Le modèle dynamique mis en oeuvre est donc responsable des déplacements du fluide. Le calcul de la dynamique s'effectue en fonction d'un modèle géométrique et du temps. La surface qui servira au processus de visualisation est ensuite extraite à partir de la géométrie déformée. Une animation négligée retirera toute la crédibilité de réalisme même en utilisant les techniques de rendu les plus sophistiquées. Le modèle dynamique mis en oeuvre est donc crucial et cette section analyse les différents modèles utilisées en informatique graphique, en dehors des aspects de géométrie et d'interactions.

Beaucoup de travaux existent en synthèse d'image concernant la dynamique des fluides qui peuvent être répertoriés selon trois approches :

1. La première approche s'intéresse à un résultat visuellement réaliste. Peu importe la conception si le résultat est semblable au comportement du fluide. Très populaire dans les domaines ludiques tels que les jeux vidéos, elles ont l'inconvénient d'être physiquement faux et ne peuvent représenter qu'un mouvement spécifique. De plus, une paramétrisation/intégration non-intuitive est laissée à l'utilisateur ;
2. L'objectif de la deuxième approche est d'obtenir un résultat visuellement réaliste tout en autorisant un certain contrôle. Ce contrôle est obtenu par l'introduction de notions physiques ;
3. La troisième approche vise la reproduction conforme à la réalité de la dynamique des fluides en repoduisant les causes du phénomène.

3.2.1 Modèle basé-visuel

Principe

Élaborer un modèle dynamique basé sur l'observation du comportement des fluides. Différentes descriptions du mouvement sont établies et sont placées dans la scène par l'animateur, comme par exemple une source ou un tourbillon.

Mise en oeuvre

Karl Sims [91] a défini quelques primitives de mouvement pour contrôler ses particules. Avec ses opérations de spirale, de rebond, de vortex, il reproduit des phénomènes naturels tels que les chutes d'eau. Pour alléger les calculs, l'auteur propose une méthode de parallélisation du traitement des particules. Jakub Wejchert et David Haumann [50] ont utilisé des primitives similaires de comportement. Nelson Max et Barry Becker [67] proposent d'animer et de déplacer une texture plaquée sur des objets pour donner l'illusion de mouvement.

Avantages et inconvénients

Ces techniques basées-visuel sont relativement simples à mettre en oeuvre, offre des temps de calculs très compétitifs et sont faciles à utiliser. Cependant, ces modèles nécessitent une paramétrisation importante de la part de l'utilisateur. Puisque ces techniques sont façonnées pour reproduire un cas précis, dans le cadre d'interactions fluide-structure elles sont malheureusement inadaptées.

3.2.2 Modèle basé *physique simplifiée*

Principe

Les modèles basés-physique permettent de simuler des effets complexes de la dynamique des fluides et de contrôler leur comportement par des paramètres intuitifs. En revanche, la complexité de ces modèles est telle que les temps de calculs résultants sont longs. Inversement, les modèles basés-visuels sont beaucoup plus rapides en terme de temps d'exécution, mais sont limités dans leur utilisation. Le mélange des deux méthodes vise à profiter des avantages de chacune.

Mise en oeuvre

Michaël Kass et Gavin Miller [53] proposent une méthode basée sur des équations simplifiées de la dynamique des fluides en eaux peu profondes qui permet de reproduire la réflexion et la réfraction des vagues. Le volume de l'eau est approximé par une carte

de hauteur et tant que les forces appliquées sur la surface sont assez faibles, l'erreur introduite par l'approximation surfacique est petite. Cette carte est échantillonnée par un ensemble de particules contrôlées par des équations de mouvement. Le fait de considérer la surface uniquement par rapport au volume de l'eau allège grandement les calculs. En revanche, ce modèle a l'inconvénient de ne pas supporter les effets de cassures dues à des forces importantes.

Cette méthode a ensuite été étendu par James O'Brien et Jessica Hodgins [79] pour prendre en compte des interactions unidirectionnelles avec des objets qui entrent en contact avec le liquide. Puisque la modélisation de l'eau par sa surface uniquement supporte avec difficulté ce genre d'interactions, les auteurs utilisent des colonnes de voxels disposant d'une isotropie verticale pour reproduire le volume du fluide. Lorsqu'un objet entre en contact avec la surface du fluide, la différence de pression liée à la collision se propage par le biais de ces colonnes (voir figure 3.7). David Mould et Yee-Hong Yang [74] ont amélioré ce modèle en ajoutant une anisotropie horizontale.

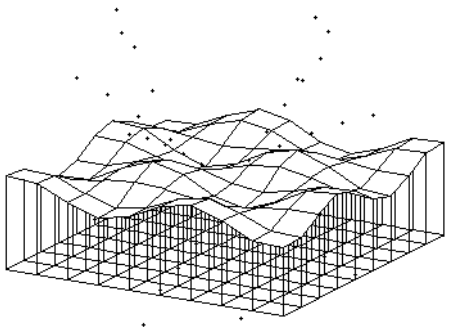

FIG. 3.7: Les colonnes de voxels servent à approximer le volume du fluide, d'après le modèle de James O'Brien et Jessica Hodgins [79].

Cem Yuksel *et al.* [115] déplacent des particules selon une équation de propagation d'onde sur une surface, sans frictions, sans interactions inter-particules. Une hauteur est calculée en fonction de la densité des particules. Les interactions fluide-structure sont réduites à des contacts objets/particules 2D, permettant un large éventail d'interactions bidirectionnelles. De plus, cette méthode intégrée sur GPU offre d'excellents résultats en temps interactifs.

Avantages et inconvénients

l'introduction de notions physiques, même simplifiées, offre un degré de réalisme supplémentaire, ainsi qu'un paramétrage plus intuitif que le paramétrage des modèles basés-visuel. Ces méthodes sont également plus rapides que les modèles basés-physique. Cependant, le champ d'action de ces approches basées sur de la physique simplifiée est plus restreint que le rayon de possibilité proposé par des méthodes plus physiques. De plus, la plupart des interactions fluide-structure avec ces modèles sont unidirectionelles et puisque que mon cadre de recherche est axé sur les interactions bidirectionnelles, je me suis intéressé aux approches basées-physique.

3.2.3 Modèles basés-physique

En animation, la physique aide les artistes à concevoir des mouvements réalistes. Entre autres, après la configuration de quelques paramètres intuitifs, la simulation est autonome. Par exemple, un artiste dessine un plan qui représente le sol ; puis quelques cubes au dessus du plan ; lorsque la simulation est lancée, les cubes entrent en contact avec le plan, dû à la gravité. Grâce à la mise en oeuvre d'un modèle basée-physique, l'utilisateur n'a pas besoin de contrôler manuellement chaque étape de collisions de ce procédé "trivial". Par ailleurs, certaines animations complexes sont fastidieusement élaborées manuellement par les utilisateurs. C'est dans ce cadre que des algorithmes bien déterminés viennent aider la conception de mouvements.

Les bases physiques et mathématiques des équations de la mécaniques des fluides sont brièvement abordées dans la sous-section 2.1.2. Les modèles dynamiques des fluides en synthèse d'images sont répartis en trois catégories :

- approche spectrale : ce procédé consiste à représenter la surface de l'eau par le biais des spectres des vagues.
- approche Eulérienne/semi-Lagrangienne : des points d'observations, ou cellules, suivent le déplacement du fluide. Les équations de la dynamique sont résolues sur chacune de ces cellules. Une dérivée de cette approche est le modèle Lattice-Boltzmann (LBM) ou treillis de Boltzman.
- approche Lagrangienne : le fluide est échantillonné en particules, sur lesquelles

les équations de la dynamique sont appliquées.

3.2.3.1 Approche spectrale

Principe Représenter les surfaces de grandes étendues d'eau par des sinusoïdes ou des trochoïdes. Les amplitudes et fréquences de ces fonctions sont basées sur des spectres de vagues.

Mise en oeuvre Nelson Max [68] a utilisé le modèle d'ondes de Stokes pour animer la surface de grandes étendues d'eau (voir la figure 3.8).

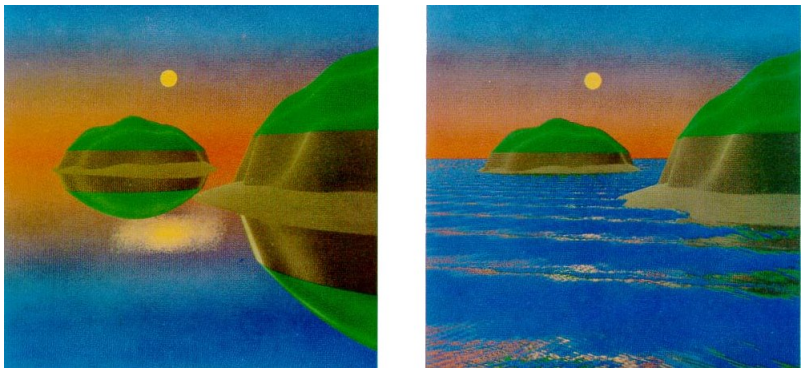

FIG. 3.8: La variation des paramètres d'amplitude et de fréquences des ondes permet d'avoir des eaux calmes (gauche) ou mouvementées (droite).

Alain Fournier et William Reeves [36] et Sébastien Thon *et al.* [101] ont représenté la surface d'océans par des équations de trochoïdes en utilisant des relevés spectraux océanographiques. La profondeur des eaux affecte le mouvement de la surface et permet de reproduire la reflexion des vagues, comme illustré dans la figure 3.9. Leur but est d'analyser la surface de l'eau en cherchant à obtenir les spectres des vagues. De tels spectres décrivent la répartition de l'énergie des vagues en fonction de leurs fréquences et de leurs directions. Ces spectres sont déduis des mesures de la hauteur de la surface de l'eau en fonction du temps. Ces mesures sont enregistrés par des bateaux dotés de capteurs qui sillonnent la mer. En paramétrant des trochoïdes en fonction de ces mesures, de grandes étendues d'eau peuvent ainsi être reproduites avec un aspect physiquement réaliste.

FIG. 3.9: A gauche l'eau se déverse sur une plage. À droite, le phénomène de vague est reproduit. Images Alain Fournier et William Reeves [36].

Avantages et inconvénients Ces approches sont peu coûteuses en mémoire et en temps de calculs et permettent de représenter de grandes étendues d'eau. Cependant, uniquement le mouvement de la surface est considérée : les interactions fluide-structure avec ce genre de modèles sont très limitées et n'entrent donc pas dans mon cadre de recherche.

3.2.3.2 Approches Euleriène et semi-Lagrangienne

Principe Les équations de la dynamique des fluides sont résolues pour chaque échantillon par différences finies.

Mise en oeuvre Le premier travail, en synthèse d'images, dans lequel la simulation dynamique du fluide est basée physique a été réalisé dans un cadre cinématographique par Larry Yaeger *et al.* [114]. Dans ce film "2010", la surface de la planète Jupiter est omniprésente. À l'époque (1986) et même encore aujourd'hui, il est difficile d'obtenir des photographies haute qualité de cette planète, d'où la nécessité de la représenter virtuellement. Pour animer cette surface gazeuse, les auteurs ont utilisé des modèles météorologiques[13] pour simuler l'écoulement du fluide. Les grosses zones de turbulences d'apparences fixes sont considérées comme des frontières (par exemple la célèbre grande tâche rouge[14]). Le résultat de l'écoulement est par la suite enregistré dans une texture qui est plaquée sur le modèle géométrique de la planète.

[13]modèle barotropique : http ://en.wikipedia.org/wiki/Barotropic
[14]http ://en.wikipedia.org/wiki/Great_red_spot

Jim Chen et Niels da Vitoria Lobo [18] ont simulé de l'écoulement d'eau en trois dimensions interagissant avec des obstacles statiques et dynamiques. Dans ce modèle, les équations de Navier-Stokes (NS) sont résolues par différences finies en 2D sur une surface discrétisée par la méthode *Marker-And-Cell* (MAC) de Francis Harlow et Eddie Welch [46] ; la troisième composante, c'est à dire la hauteur de la surface, est définie en fonction de la pression calculée sur la cellule : puisque l'eau est considérée comme étant incompressible, une augmentation de la pression engendrera alors une élévation de la hauteur et inversement une baisse de pression entraînera une diminution de la hauteur.

Nick Foster et Dimitris Metaxas [34] ont été les premiers en synthèse d'images à avoir résolu les équations de NS en trois dimensions pour animer de l'eau. Ces équations sont calculées par différences finies sur une grille de cellules MAC (voir figure 3.10) et intégrées temporellement par une méthode explicite. Les frontières, les obstacles statiques et dynamiques sont définies par l'équation 2.7.

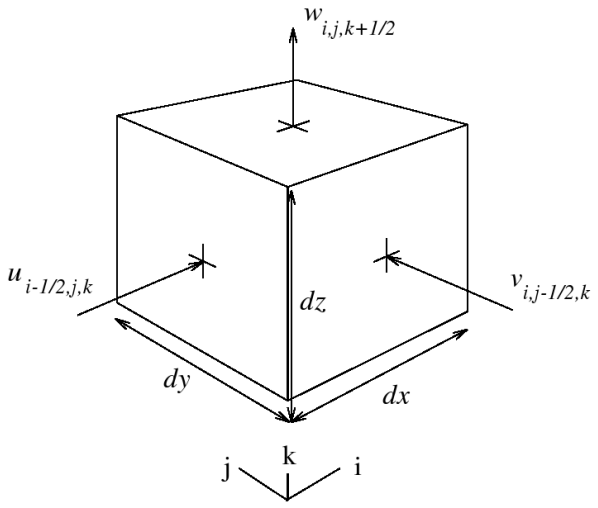

FIG. 3.10: une cellule MAC en 3D.

La méthode proposé par Jos Stam [92] vise la reproduction en temps interactifs de phénomènes gazeux. Ce modèle utilise une technique semi-Lagrangienne et une méthode implicite pour le calcul de l'advection. Le résultat de cette combinaison est que la simulation est temporellement plus stable : le pas d'intégration temporelle étant

moins contraint, le gain en temps de calculs est considérable (voir le chapitre 4 pour plus de détails). Il est important de noter que la stabilité des méthodes explicites pour les fluides dépend d'une forte condition qui impose un pas d'intégration temporelle très petit, d'où des temps de calculs importants. L'inconvénient des méthodes implicites est qu'elles ajoutent un amortissement artificiel qui engendre dans le cadre des gaz un mouvement estompé. Nuisible pour des applications scientifiques, en synthèse d'images cela est moins gênant car les animateurs peuvent, par exemple, ajouter des forces de mouvement en temps voulu pour "ranimer" le fluide. Ce modèle est devenu une référence en dynamique des fluides en synthèse d'images grâce à ses performances. Cependant, une perte de volume est visible en ce qui concerne l'affichage.

Cette perte de volume est acceptable pour les gaz, mais est irréaliste pour des liquides. Pour pallier ce problème, Nick Foster et Ron Fedkiw [33] ont introduit la méthode de *Particle Level Set* (voir la sous-section 3.1.3.3 pour plus de détails). Avec les méthodes semi-Lagrangiennes, qui amortissent le mouvement, l'effet de tourbillonnement des fluides est atténué, voir éliminé. Dans la mesure du possible Ron Fedkiw *et al.* [29] proposent d'amplifier l'effet de tourbillonnement pour le préserver. Andrew Selle *et al.* [88] ont introduit des particules de vorticité dans des simulations Eulériennes.

Mark Carlson *et al.* [16] ont simulé des fluides visqueux et des fontes d'objets. Pour ce faire, les auteurs considèrent que tout est fluide, c'est-à-dire que les solides sont traités comme des fluides à viscosité très élevée. La viscosité de l'objet est modifiée en fonction de la température, la fonte des objets est reproduite en basculant de l'état "solide" à l'état liquide.

Tolga Gotktekin *et al.* [39] ont cherché à simuler des fluides viscoélastiques, c'est-à-dire Non-Newtonien. Pour ce faire, les auteurs ont résolu l'équation de NS avec le terme viscoélastique :

$$\frac{\partial \mathbf{v}}{\partial t} = -(\mathbf{v} \cdot \nabla)\mathbf{v} - \frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{v} + \frac{\mu_e}{\rho} \nabla \epsilon + \mathbf{f}$$

où μ_e est le coefficient d'élasticité et ϵ est la tension élastique de la cellule. Des liquides tels que du dentifrice, de la gélatine ou encore de la pâte à modeler peuvent

être modélisé par ce procédé. La résolution des équations de NS est similaire à celle mis en oeuvre par Nick Foster et Ron Fedkiw [33].

Zhu et Bridson [116] proposent une méthode originale pour animer du sable, en considérant ce dernier comme un fluide. La friction inter-grain est simulée en ajoutant des termes de tension de friction et de rigidité dans les équations de NS. Le calcul du déplacement des grains de sables est effectué en combinant des particules Lagrangienne pour le calcul de l'advection et d'une grille auxiliaire pour une meilleure définition des bordure et de l'incompressibilité, en mettant en oeuvre une méthode combinant les *Particle-In-Cell* (PIC) de Francis Harlow [45] et les *FLuid-Implicit-Particle* (FLIP) de Jorg Brackbill et H. Ruppel [11].

Avantages et inconvénients Ces modèles ont l'avantage de produire une mouvement basé physique qui est réaliste. La structure de grille permet entre autres de résoudre les équations de NS avec stabilité mais aussi facilite l'extraction de la surface du fluide. En revanche, et à cause notamment de la correction globale de la pression, ces méthodes sont difficilement "parallèlisables", par conséquent les temps de calculs sont relativement importants. De plus, pour une conservation de volume ce modèle nécessite l'introduction de particules auxiliaires. Par ailleurs, l'implantation de ces méthodes est difficile.

3.2.3.3 Approche Lattice Boltzmann

Principe La méthode dite de *Lattice Boltzmann Method* (LBM), ou treillis de Boltzmann, est une alternative à la résolution des équations de NS. Ce modèle s'apparente à un automate : pour chaque cellule de la grille et à chaque itération, une particule fictive choisit sa nouvelle cellule en fonction d'un nombre de directions prédéfinies. Cette étape s'appelle la **propagation**. Lorsque plusieurs particules venant de différentes directions se retrouvent dans la même cellule, elles entrent en contact et une nouvelle destination leur est attribuée selon un modèle de **collision**. La méthode LBM repose sur cette règle de collision qui doit préserver le nombre de particules, le moment et l'énergie avant et après la collision.

L'équation discrète de Botzmann permet de simuler l'écoulement d'un fluide Newtonien qui est de la forme suivante :

$$f_i\left(x + \Delta t \cdot \mathbf{v_i}, t + \Delta t\right) = f(x,t) + \Omega_i$$

où f_i est la concentration des particules de vitesse $\mathbf{v_i}$ dans la i-ème cellule. En fonction de $\mathbf{v_i}$ les particules se déplacent dans la cellule voisine au temps $t + \Delta t$ suivant. L'opérateur de collision utilisé Ω_i est souvent celui de Bhatnagher-Gross-Krook (BGK) qui fait tendre la distribution des particules dans un état d'équilibre de distribution $f_i^{eq}(x,t)$:

$$\Omega_i = \frac{1}{\tau}\left(f_i(x,t) - f_i^{eq}(x,t)\right)$$

où τ est le paramètre de relaxation qui détermine la viscosité du fluide par $\nu = (2\tau - 1)/6$. La fonction d'équilibre de distribution est calculée en fonction de la densité et de la vitesse des particules dans la cellule courante.

Mise en oeuvre L'utilisation du modèle LBM est relativement récente en informatique graphique. Néanmoins de très bon résultats découlent de cette approche. À titre d'exemple, la simulation de fluide du célèbre logiciel libre de modélisation et d'animation Blender[15] intitulée El'beem[16] est basée sur ce modèle.

Xiaoming Wei *et al.* [111, 110, 109, 28] ont été les principaux acteurs à avoir utilisé les LBM en synthèse d'images. Les équations des LBM sont simples à résoudre et se prêtent bien à la parallélisation. Par conséquent les auteurs ont adapté les algorithmes sur GPU et ont obtenu des simulations en temps interactifs. La gestion des collisions avec des objets qui peuvent être dynamiques et déformables, est réalisée par l'opérateur de collisions Ω_i en transformant les bordures des objets en cellules de fluides. Nils Thuerey [102], l'auteur principal du greffon de simulation de fluide pour Blender, s'est intéressé à la parallélisation, à la stabilisation, à l'optimisation, à l'interaction des LBM ainsi que la reconstruction de surface.

[15] http ://blender.org/
[16] http ://elbeem.sourceforge.net/

Avantages et inconvénients Les LBM sont simples à implémenter, la conservation de la masse est assurée et elles sont "parallélisables". De plus, les simulations sont effectuées en temps interactifs, pour des grilles de taille raisonnables. Par contre, la nécessité de l'utilisation d'une grille limite les déplacements du fluide dans un espace clos. Également le pas d'intégration temporelle avec ce modèle est très contraint. Par contre, ce modèle est gourmand en terme de ressources mémoires.

3.2.3.4 Approche Lagrangienne

Principe Représenter un fluide par un ensemble de particules dont le comportement est contrôlé par des principes physiques.

Mise en oeuvre Les particules exercent entres elles des forces d'attraction et de répulsion. Lorsqu'elles sont légèrement éloignées, elles s'attirent (force gravitationnelle dominante) ; lorsqu'elles sont trop proches, elles se repoussent (force de pression dominante). Ces forces[17] sont dites intermoléculaires. Gavin Miller et Andrew Pearce [71] et David Tonnesen [103] se sont inspirés de ces forces intermoléculaires pour réaliser des simulations de liquides et de fontes de solides. Dimitri Terzopoulos *et al.* [97] ont utilisé les équations de la thermodynamique pour simuler le transfert de chaleur dans des systèmes masses-ressorts volumiques. Les coefficients de rigidité des ressorts sont déterminés en fonction de la température : une basse température accroît la rigidité des ressorts, simulant ainsi des solides ; à l'inverse lorsque la température est élevée, la rigidité est nulle et un modèle de fluide similaire à celui de Gavin Miller et Andrew Pearce [71] est appliqué sur les particules.

Les *Smoothed Particles Hydrodynamics* (SPH - voir annexe 9.1) sont très sollicitées dans le domaine de la physique en général et particulièrement par le calcul de la dynamique des fluides en informatique graphique. Cette méthode a été inventé par Leon Lucy [64] et Robert Gingold et Joe Monaghan [38] (lire le très bon résumé de Joe Monaghan [73]) pour simuler des phénomènes astrophysiques. C'est une méthode basée particule qui s'affranchit d'un modèle de grille pour les calculs des dérivées : celles-ci

[17] http ://en.wikipedia.org/wiki/Van_der_Waals_Force

sont calculées par différence analytique de formules d'interpolations, et donnent lieux à des équations différentielles ordinaires (EDO) qui sont assez bien maîtrisées. Toutes les interactions possibles entre les particules sont considérées et calculées comme des forces agissant entre chaque paire de particules. C'est une méthode efficace qui permet d'approximer les équations de NS. Le coeur des SPH est une méthode d'interpolation qui autorise l'expression de toutes fonctions en terme de valeurs calculées à partir d'un ensemble de points désordonnés (les particules). La fonction d'interpolation est définie par :

$$A(\mathbf{x}) = \sum_i m_i \frac{A_i}{\rho_i} \omega(\mathbf{x} - \mathbf{x}_i, h) \tag{3.1}$$

où $\mathbf{x}_i, m_i, \rho_i$ désigne respectivement la position, la masse et la densité de la i-ème particule. Il est à noter que les particules en dehors du rayon d'action h du noyau ω n'auront aucun effet, d'où l'intérêt ou la nécessité d'utiliser une optimisation spatiale pour la recherche du proche voisinage (voir le chapitre 5). A_i désigne une quantité quelconque au point $\mathbf{x}_i$. Pour obtenir une dérivée de A, par exemple ∇A, il suffit d'utiliser un noyau différentiable :

$$\nabla A(\mathbf{x}) = \sum_i m_i \frac{A_i}{\rho_i} \nabla\omega(\mathbf{x} - \mathbf{x}_i, h) \tag{3.2}$$

Pour une justification des équations 3.1 et 3.2, il faut utiliser un noyau de type Gaussienne. C'est la première *golden rule* des SPH.

Mathieu Desbrun et Marie-Paule Cani [21] ont introduit les SPH en synthèse d'images. Avec quelques modifications de certains paramètres, notamment la pression, ils ont animé des corps très déformables. En calculant des valeurs de viscosités et de température par la fonction d'interpolation des SPH, Dan Stora *et al.* [95] ont simulé de l'écoulement de lave. Les auteurs utilisent une méthode d'intégration temporelle à pas adaptatif pour accélérer le modèle tout en préservant la stabilité.

Matthias Müller *et al.* [75, 76, 77] ont effectué de grandes avancés dans ce domaine. En formulant de nouveaux noyaux de lissage, les auteurs obtiennent une simulation de fluide en temps réel avec environ 5k particules. Ils ont également mis en place un

modèle représentant les tensions de surface. Dans l'article [75] les auteurs proposent d'extraire la surface du fluide en utilisant soit du *splatting*, soit du *marching cubes* (illustré en figure 3.11). Le schéma d'intégration temporelle utilisé est la méthode dite de *Leap-Frog*. Matthias Müller *et al.* [76] proposent une extension à ce modèle en intégrant des interactions entre un fluide et des tétraèdres (voir la section 3.3 pour plus de détails). Matthias Müller *et al.* [77] se sont par la suite intéressés aux interactions fluide-fluide pour simuler des effets de liquides multi-phases comme par exemple une lampe à lave ou de l'eau qui boue. Pour ce faire, les auteurs utilisent des attributs de polarité et de viscosité définis pour chaque particules. La transition de phase est reproduite en ajustant la densité et le type de particules en fonction de la température. Des particules d'air sont dynamiquement générées lorsque des poches d'air se forment au sein d'un fluide.

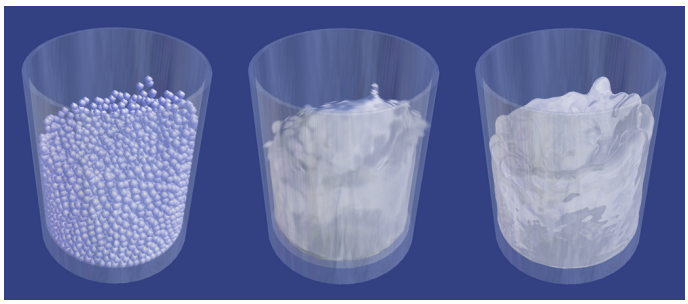

FIG. 3.11: Matthias Müller *et al.* [75] représentent la surface du volume du liquide par (*gauche*) une sphère par particule ∼20 images par secondes (Frames Per Second - FPS -) (*milieu*) des *splats* ∼20 FPS (*droite*) avec l'algorithme des *marching cubes* ∼5 FPS.

Simon Premoze *et al.* [82] ont résolu les équations de NS avec la méthode MPS (*Moving Particle Semi-implicit*). Cette méthode assure une meilleure incompressibilité que les SPH. Une partie de l'équation de NS est résolue explicitement, une autre (la pression) est calculée par un modèle implicite. Pour pallier le calcul difficile du flux entrant et sortant d'un fluide dans un espace donné avec une méthode Lagrangienne, les auteurs utilisent une grille et calculent la convection d'une façon Eulérienne.

Kevin Steele *et al.* [94] ont modélisé des fluides très visqueux avec une méthode similaire aux SPH : des forces de répulsions, de frictions et d'attractions sont calculées

en fonction du voisinage. La méthode d'intégration numérique utilisée est celle de *forward* Euler. Celle-ci est réputée pour être instable, mais l'ajout de la viscosité dans leur modèle correspond à une augmentation du coefficient d'amortissement, stabilisant un peu plus ces équations (voir mes travaux dans [4] et dans la section 4.4). La conservation du volume du fluide est un point faible des méthodes Lagrangiennes. Kevin Steele *et al.* proposent une méthode de relaxation itérative qui distancie les particules afin de préserver le volume global.

Le filtre de relaxation utilisé par Kevin Steele *et al.* [94] permet de ne simuler que des fluides très visqueux. La cas échéant, des artefacts visuels apparaissent comme par exemple des particules qui s'attirent pour former un amas puis qui se repoussent en plusieurs groupes. Pour pallier ce problème, Simon Clavet *et al.* [20] utilisent deux filtres qu'ils nomment *double density relaxation*. Le premier interagit avec les particules situées dans un proche voisinage, tandis que le deuxième s'exerce principalement dans une voisinage un peu plus lointain. Ce procédé permet, entre autres, de modéliser les tensions de surface comme par exemple la formation et la séparation de gouttes d'eau (illustrée figure 3.12). Le schéma d'intégration numérique utilisée est la *prediction-relaxation*, qui est explicite et relativement stable. Les auteurs ont ajouté à leur modèle de la plasticité, de la viscosité et de l'élasticité. La viscosité est calculée d'après la différence de vitesses des particules voisines (un autre filtre) et l'élasticité est simulée en ajoutant dynamiquement des ressorts entre les voisins. La variation de la longueur initiale des ressorts permet de reproduire un comportement de plasticité. Des simulations de gouttes d'eau, de fontaine, de pâte à modeler et des objets flottants sont présentées en temps interactifs pour un maximum approximatif de 1k particules, sans solliciter le GPU.

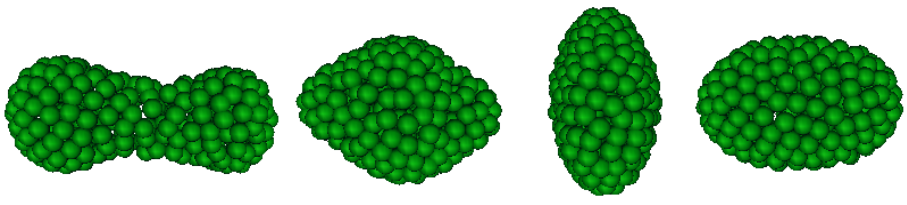

FIG. 3.12: Une goutte de liquide en oscillation après sa formation. Chaque particule est représentée par une sphère.

Avantages et inconvénients Un avantage considérable du modèle Lagrangien par rapport au modèle Eulérien est l'utilisation des particules[18] : les masses sont toujours préservées. Par conséquent l'équation de la conservation de la masse n'a pas à être résolue, réduisant la complexité des calculs. La mise en oeuvre des modèles Lagrangien est relativement simple comparé au modèle Eulérien. En fonction du nombre de particules, les simulations basées sur ce modèle fonctionnent en temps-réel et sont "parallèlisables". L'inconvénient principal des ces approches Lagrangienne est les propriétés telles que la vitesse ou encore la pression sont calculées par des fonctions d'interpolation qui nécessitent la connaissance du voisinage. La détermination de ce voisinage est un processus qui est relativement coûteux. De plus, ces méthodes ont des soucis d'incompressibilité qui engendrent des pertes de volumes.

3.2.3.5 Avantages et inconvénients des modèles basés-physiques

Les notions de physiques apportent un réalisme indéniable dans les animations. De plus, les simulations sont autonomes après une phase de paramétrage intuitive. La dynamique des fluides en synthèse d'images commence à faire ses preuves, car aujourd'hui dans beaucoup de grosses productions cinématographiques, le spectateur discerne avec difficulté le réel du virtuel. Les inconvénients principaux sont que les temps de calculs de simulations détaillées sont longs et que la rigueur scientifique n'est pas toujours au rendez-vous. Dans le seul but d'obtenir un résultat visuellement convaincant, beaucoup de simplifications et d'approximations sont faites et par conséquent, les résultats sont moins précis. Cependant ce manque de rigueur tend à se réduire.

3.3 Interaction Fluide-Structure

Bien qu'un fluide soit un objet déformable, il est considéré ici comme une entité à part entière et les structures déformables regroupent les méthodes à éléments finis telles que les systèmes masses-ressorts.

[18] attention aux *particles level sets* où les particules ne sont que des marqueurs qui servent à préserver le volume de la surface implicite.

En synthèse d'images, il existe un certain nombre de modèles qui simulent séparément le comportement d'objets rigides, de structures déformables et de fluides. Ces modèles sont judicieusement ajustés et optimisés pour reproduire précisément le résultat attendu. Par conséquent, un couplage entre un fluide et un objet doit être élaboré avec soin puisque les structures diffèrent. Il est bien sûr inconcevable de laisser à l'utilisateur la définition manuelle de toutes les interactions possibles entre deux objets en contact et ce, pour chaque simulation. Ce couplage nécessite donc l'élaboration d'une interface de communication entre un fluide et une structure en contact. Plusieurs approches innovantes ont vu le jour et la plus répandue consiste à transformer les structures en fluide, ou en bordure de fluide. Ainsi faisant, les interactions sont implicitement prises en compte par le modèle dynamique du fluide. Cette méthode à l'avantage de conserver les propriétés physiques sans modifications additionnelles.

Peu de travaux concernent les interactions entre fluides (Eulérien ou Lagrangien) et structures (Lagrangien) sans épaisseurs, ou membranes. Le problème des objets fins est connu dans le domaine des collisions car pour éviter que des particules passent au travers de membranes, il est nécessaire de mettre en place un modèle de détection de collisions continue. Par exemple un drap, une feuille ou une nageoire de poisson est typiquement le genre d'objets qui posent problème à cause de leur épaisseur.

Cette section est divisée en trois parties : 1) les modèles Eulérien et semi-Lagrangien 2) les modèles LBM 3) les modèles Lagrangien.

Puisque la gestion des interactions diffère au sein de chaque partie, les avantages et inconvénients sont discutés pour chaque article cité et non globalement par section comme établi précédemment. La figure 3.13 récapitule les différents travaux concernant l'interaction fluide-structure dans le domaine de l'informatique graphique.

3.3.1 Modèles Eulérien et semi-Lagrangien

Cette section est divisée en trois sous-sections, c'est-à-dire les interactions unidirectionnelles entre un solide et un fluide, inversement les interactions unidirectionnelles entre fluide et un solide et finalement les interactions bidirectionnelles.

		Fluide	*Interaction*	*Objet*
Eulérien-semi-Lagrangien	Jim Chen et Niels da Vitoria Lobo [18]	χ	Les objets sont traités comme les bordures.	$\leftarrow$
	James O'Brien et Jessica Hodgins [79] David Mould et Yee-hong Yang [74]	χ	Les forces d'impacts sont appliquées sur les cellules.	$\leftarrow$
	Nick Foster et Dimitris Metaxas [34]	$\rightarrow$	Les objets sont affectés par les forces de pression et les vitesses des cellules qu'ils recouvrent.	χ
	Nick Foster et Ron Fedkiw [33]	χ	La vélocité et la surface de l'objet servent à ajuster les attributs physiques des cellules qu'il recouvre.	$\leftarrow$
	Olivier Génveaux *et al.* [37]	$\rightarrow$	Un échange de forces est effectué à partir de particules de marquages (situées en bordure de la surface du liquide) et les sommets des objets.	$\leftarrow$
	Mark Carlson *et al.* [17]	$\rightarrow$	Le solide est transformé en liquide extrêmement visqueux, l'interaction solide-fluide est alors naturellement traitée par le modèle de fluide. Les forces de pression sont transformées en vecteurs de rotation et de translation pour réaliser l'interaction fluide-solide.	$\leftarrow$
	Eran Guendelman *et al.* [43]	$\rightarrow$	Un algorithme de lancer de rayon sert à délimiter les bordures des membranes. Les attributs de pressions et de vitesses aux sommets des cellules sont ajustés en fonction de l'objet qu'elles contiennent. Ces forces sont ensuite appliquées sur l'objet pour une interaction bidirectionelle.	$\leftarrow$
	Huamin Wang *et al.* [108]	χ	L'interaction des gouttes d'eau est en fonction d'un angle de contact. La surface du liquides est judicieusement ajustée pour conserver un angle correct.	$\leftarrow$
Lagrangien	Matthias Müller *et al.* [76]	$\rightarrow$	Des particules *fantômes* de fluide sont ajoutées autour des objets pour obtenir une interaction fluide-fluide.	$\leftarrow$
	Kevin Steel *et al.* [94]	χ	Les solides sont transformées en liquides, la friction des contacts est calculée par une matrice d'adhésion	$\leftarrow$
	Simon Clavet *et al.* [20]	$\rightarrow$	À chaque itération, les objets accumulent les impacts des particules de fluide qui se déplacent, puis dans un deuxième temps, les particules situées l'intérieur des objets sont projetées le long de la surface.	$\leftarrow$
LBM	Xiaoming Wei, Lie Wei *et al.* [111, 110, 109] Alexandrova *et al.* [5]	χ	Les cellules recouvertes par un objet sont traitées par le modèle de collision de la LBM.	$\leftarrow$

FIG. 3.13: Ce tableau est un récapitulatif des différents modèles d'interaction fluide-structure mis en oeuvre en synthèse d'images.

3.3.1.1 Interactions unidirectionnelles solide→fluide

Jim Chen et Niels da Vitoria Lobo [18] ont été les premiers, à ma connaissance, à avoir introduit les interactions unidirectionnelles en synthèse d'images avec des obstacles dynamiques. Une cellule bidimensionnelle de leur fluide contenant un objet est considérée comme une bordure. En renouvelant le marquage des cellules à chaque itération, les déplacements des objets dynamiques sont pris en compte. Avec ce procédé, la taille des objets doit être d'au minimum celle d'une cellule sinon il y aura un espace de vide entre l'objet et le fluide.

James O'Brien et Jessica Hodgins [79] ont dans la même année proposé un autre modèle de gestion d'interactions unidirectionnelles. L'objectif de leur travail est de simuler des éclaboussures d'eau (*splashing fluids*) lorsqu'un objet vient heurter ou flotter sur la surface du fluide. Cette surface est une carte de hauteurs constituée d'un ensemble de colonnes qui simule l'échange de flux par le biais de tuyaux virtuels. Le flux communique la différence de pression entre les colonnes adjacentes, selon les lois de l'hydrostatique. Lorsqu'un objet entre en contact avec le fluide, l'impact est transformé en une information de pression. Ainsi faisant, leur modèle simule les interactions unidirectionnelles du solide vers le fluide. Pour amplifier l'effet de contact, des particules simulant des éclaboussures sont générées. La force d'impact est comprise dans un intervalle, soit $f_0 \in [0, f_{max}]$, où f_{max} est une valeur qui annule l'équation d'interaction. Pour simuler des effets de collision qui varient en fonction de la surface de contact des solides, les auteurs utilisent une méthode de recherche heuristique d'une valeur de force dans cet intervalle. Les objets plus petits qu'une cellule engendreront une perturbation du fluide aussi importante qu'une perturbation engendrée par un objet de la taille d'une cellule exactement.

David Mould et Yee-Hong Yang [74] ont amélioré le modèle précédent en s'intéressant d'avantage au mouvement de gouttes d'eau, ainsi qu'à leurs interactions avec le volume du fluide, au phénomène de bulles dans l'eau et au mouvement du fluide en fonction de sa profondeur. Pour ce faire, les colonnes d'eau sont divisées en voxels, qui communiquent leurs différences de pressions avec leurs voisins horizontaux en plus de leurs voisins verticaux. Cela a pour effet, notamment, la simulation de la réfraction

des vagues due à la variation de la profondeur de l'eau. L'échange horizontal des pressions permet de simuler des phénomènes semblables à ceux établis par la théorie de l'hydrodynamique. Hormis le fond marin, la gestion des interactions est identique à celle proposée par James O'Brien et Jessica Hodgins [79].

Dans le modèle de Nick Foster et Ron Fedkiw [33], la vitesse du fluide $\mathbf{v}_p$ dans une cellule contenant un objet est ajustée pour que $\mathbf{v}_p \cdot \mathbf{n}_s \geq 0$, i.e. les particules doivent se déplacer dans la direction du vecteur normal $\mathbf{n}_s$ de la surface de l'objet. Les vitesses des cellules à l'intérieur de l'objet sont égales à la vitesse de l'objet. L'attribution des paramètres de vitesse et de pression, en fonction du vecteur normal de la surface de l'objet existant dans la cellule, limite cette méthode à la présence d'un seul polygone par cellule. Une interpolation de vecteurs normaux est possible lorsque plusieurs polygones coexistent dans une même cellule, mais cela peut engendrer des artefacts visuels.

Huamin Wang *et al.* [108] ont modélisé des fluides à petite échelle, tels que les gouttes d'eau, qui glissent sur une surface. Pour ce faire, les auteurs définissent une *surface virtuelle* pour conserver un bon angle de contact[19], en modifiant la fonction de distance des *level set* représentant la surface du liquide. La forme et le déplacement de la goutte sont définies en fonction de l'angle de contact et des tensions de surface entre l'air, le solide et le liquide. Ce modèle est cependant établi pour représenter des gouttes d'eau et n'est pas utilisable pour décrire des volumes de fluide plus importants.

3.3.1.2 Interactions unidirectionnelles fluide→solide

Nick Foster et Dimitris Metaxas [34] ont proposé un modèle d'interactions qui, selon les forces de flottabilité, permet au liquide de déplacer des solides. Le fluide considère les objets comme des obstacles stationnaires qui sont traités comme des frontières, i.e. que la vitesse en sortie de la cellule est égale à la vitesse entrante et que la pression de la cellule en bordure est égale à la pression des cellules voisines, ceci afin d'éviter toutes accélérations. Chaque objet est échantillonné par un ensemble de points n_i. Sur chaque n_i est appliquée une formule de flottabilité en fonction de

[19] lorsque les fluides réels entrent en contact avec des solides, ils forment un angle caractéristique avec la surface connu sous le nom d'angle de contact.

la pression, de la gravité, de la masse et du volume associé. Dans ce modèle, une fois encore les objets doivent être au moins de la taille d'une cellule pour éviter les artefacts visuels de vide autour des objets.

3.3.1.3 Interactions bidirectionnelles

La première élaboration d'interactions bidirectionnelles entre un solide et un liquide, en synthèse d'images, a été réalisé par des français, Olivier Génevaux *et al.* [37]. Cette méthode consiste à mettre en place une interface entre un fluide et un solide qui transmet les forces d'interactions du solide au fluide et vice-versa. Les auteurs utilisent le modèle de la dynamique du fluide définit par Nick Foster et Ron Fedkiw [33]. Les objets sont modélisés par des structures de masses-ressorts. L'interface consiste à attribuer des forces de répulsions, pour chaque masse des structures, entre la masse et des marqueurs du fluides séparés d'un distance inférieure à un rayon r préétabli. Les forces, calculées par un procédé identique à un ressort, sont ajoutées respectivement à la masse de la structure et aux cellules de l'interface. Cependant ce paramètre de voisinage r n'est pas très intuitif, il dépend entre autres de la discrétisation des solides : ce rayon doit être assez grand et les structures masses-ressorts doivent être judicieusement échantillonnées. Par ailleurs, une discrétisation des forces dans la grille mal définie engendre des courants erronés. Le couplage avec des membranes n'est pas possible avec ce modèle car l'interaction objet-fluide est résolue par un procédé similaire à celui de Nick Foster et Ron Fedkiw [33].

Mark Carlson *et al.* [17] ont eu la bonne idée de traiter l'objet rigide en tant que fluide, d'où le nom de leur méthode *rigid-fluid*. L'objet rigide est discrétisé dans la grille puis est transformé en fluide extrêmement visqueux. Les calculs de collisions sont alors effectués naturellement par le modèle de la dynamique du fluide. Les forces d'interactions inter-fluides sont ensuite transformées en torseurs de déformations pour déplacer l'objet à "l'état" solide. À cause de la discrétisation, l'interaction avec des membranes reste difficile.

Eran Guendelman *et al.* [43] se sont intéressés aux interactions entre des objets fins (déformables ou rigides) et fluides (liquides et gaz). Pour ce faire, les triangles

des membranes sont élargis. Cela permet de classer les particules proches de l'objet en trois catégories et à partir de ces particules triées, la détection de collisions est accomplie par une application du lancer de rayon proposée par Robert Bridson *et al.* [12]. La classification des particules est un point crucial dans leur algorithme, notamment pour le transfert des forces et pour la conservation de la masse. Le modèle de fluide utilisé, par ailleurs légèrement modifié, est celui proposé par Nick Foster et Ron Fedkiw [33]. La méthode de collision est robuste, elle autorise la gestion des interactions entre un fluide et pour la première fois une membrane, indépendamment de la résolution de la grille. Le modèle de rendu mis en oeuvre par Doug Enright *et al.* [26] est modifié pour éliminer des artefacts visuels de la surface extraite qui passe temporellement au travers de la membrane.

3.3.2 Modèle Lattice Boltzmann

Xiaoming Wei *et al.* [110, 109] ont simulé des objets, tels que des plumes ou des bulles de savon, qui suivent le mouvement de l'air. Les auteurs assument que le déplacement de ces objets n'affecte pas le mouvement de l'air car ils sont trop légers. Par conséquent l'interaction est unidirectionelle. Le déplacement des objets est calculé en fonction de la somme des forces de contacts des cellules qu'ils recouvrent. Cette somme est cependant simpliste, elle n'est pas basée physique et des anomalies de déplacements sont visibles.

Alexandrova *et al.* [5] ont simulé le mouvement de méduses, modélisées avec des surfaces NURBS qui interagissent avec le fond marin. La méduse est un objet dynamique et déformable. Toutes les cellules contenant les points de contrôle de la surface NURBS ainsi que toutes les cellules situées dans l'interpolation linéaire entre deux points de contrôle sont considérées comme des bordures LBM. L'interaction (unidirectionnelle) est obtenue en communiquant, à chaque itération, des attributs de vitesses des cellules à l'intérieur de la bordure (vitesse de l'objet) à celles situées à l'extérieur de la bordure selon le procédé de propagation.

Les méthodes LBM sont conditionnellement stables, et dans le cas d'interactions fluide-structure, de brusques mouvements peuvent rendre le système subitement in-

stable. Ces méthodes sont donc à utiliser avec précautions dans ce cadre précis.

3.3.3 Modèle Lagrangien

Matthias Müller *et al.* [76] ont simulé des interactions bidirectionnelles fluide-structure en couplant des fluides, régis par la méthode des SPH, avec des objets Lagrangiens, modélisés par une méthode à élément finis, en introduisant des particules virtuelles de fluide ou *ghost particles*. Les interactions entre les particules de fluide et ces particules virtuelles sont naturellement prises en compte par le modèle des SPH. Les auteurs proposent de positionner judicieusement les particules virtuelles le long de la surface des objets par une méthode adaptative. Cette technique fonctionne en temps réel pour un nombre limité de particules de fluides et d'élément finis. Par ailleurs, puisque les interactions sont effectuées à partir d'un ensemble polygonal, le modèle physique qui régit le mouvement de l'objet est indépendant du modèle d'interaction.

Kevin Steele *et al.* [94] se sont intéressés aux liquides visqueux tels que le miel. Un procédé similaire aux SPH permet de calculer le déplacement des particules. La gestion des interactions unidirectionnelles avec des objets solides s'apparente à la méthode dite de *rigid fluid* proposée par Mark Carlson *et al.* [17]. Les objets sont convertis en liquide, c'est-à-dire en particules de fluide. Les interactions entre solide et liquide se résument donc aux interactions liquide-liquide. Pour ce faire, l'attraction et la répulsion sont définies par des fonctions regroupées dans ce que les auteurs nomment une matrice d'adhésion. Identiquement aux SPH, cette matrice permet de calculer les interactions dans un proche voisinage. L'utilisateur peut également paramétrer son modèle de friction solide-fluide en définissant d'autres fonctions d'adhésion, par exemple.

L'interaction fluide-solide mise en place par Simon Clavet *et al.* [20] est une adaptation de la méthode proposée par Eran Guendelman *et al.* [42]. L'algorithme de collision est modifié pour prendre en compte le phénomène de flottabilité dans un procédé en trois temps : dans un premier temps, l'objet et le fluide sont déplacés selon leur modèle dynamique ; puis les forces des particules en contact avec le solide

sont dans un deuxième temps accumulées et utilisées pour un second déplacement de l'objet ; dans un troisième temps, les particules à l'intérieur des solides sont projetées le long de la surface du solide et leur nouvelle vitesse est calculée en fonction du taux de pénétration et de la vélocité de l'objet (voir la figure 3.14). Entre autres, le phénomène de "collage" sur les solides est mis en oeuvre par une équation qui est en fonction d'une distance prédéterminée. Cette méthode ne prend en compte que les solides volumiques et par conséquent ne prend pas en compte les membranes déformables. À ce titre, je propose une extension de leur méthode pour justement reproduire l'interaction entre une membranes déformables et un fluide Lagrangien sans maillage en chapitre 6.

Récemment, Takahiro Harada *et al.* [44] a proposé de simuler entièrement sur GPU des interactions bidirectionnelles entre un fluide régit par des SPH et une membrane modélisée par des systèmes masses-ressorts. Les auteurs utilisent une grille régulière pour localiser les interactions. Lorsque des particules de fluide sont en contact avec les triangles, par ailleurs élargis, elles ont tendance à se regrouper et former des amas car il y a une différence de pression entre les deux côtés de la membrane. Pour éviter cet inconvénient, des particules virtuelles sont ajoutées de l'autre côté de la membrane pour ajuster les forces de pression.

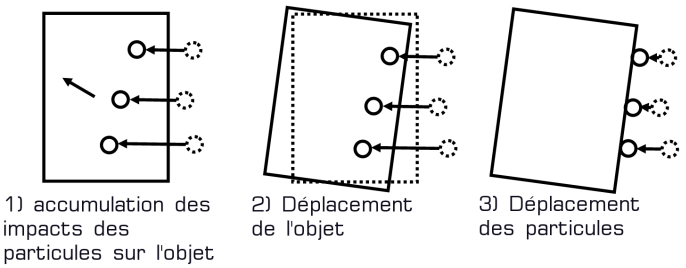

FIG. 3.14: Gestion des collisions entre solides et particules selon le modèle proposé par Simon Clavet *et al.* [20].

Les interactions entre fluides et solides sont de plus en plus convoitées. Par une extension des SPH, Matthias Müller *et al.* [77] ont modélisé le phénomène d'ébullition, et de mélange de liquides de densités différentes. Ces interactions sont directement prises en compte par les équations de mouvements qui sont quelque peu modifiées.

Par ailleurs dans ce modèle, des bulles d'air sont générées selon un certain critère et sont par la suite détruites lorsqu'elles atteignent la surface.

3.4 Conclusion

Cet état de l'art couvre trois aspects importants de la modélisation d'un fluide : le choix du modèle géométrique, le choix du modèle dynamique et le choix du modèle d'interaction. Bien que ne figurant pas comme catégorie explicite, les modèles d'illumination et d'extraction de surface sont discutés tout au long de ce chapitre. Il existe une multitude d'articles et d'ouvrages dans les domaines tels que la physique ou la mécanique relatant de la modélisation d'un fluide et, bien qu'en nombre plus restreint, également en synthèse d'images. Pour résumer cet état de l'art, la représentation d'un fluide peut être divisée en trois catégorie : 1) les modèles basés-visuel et semi-physiques, 2) les modèles physiques Eulérien et 3) les modèles physiques Lagrangien. Ces choix ont été fait en fonction de l'évolution de la puissance des ordinateurs. Les premiers travaux, jusqu'à 1996, étaient principalement basés-visuel ou physiquement simplifiés. Depuis l'article de Nick Foster et Dimitris Metaxas [34], la recherche s'est portée sur les modèles physiques Eulérien. Deux grandes avancées étant notamment le travail de Jos Stam [92] qui propose un modèle temporellement stable de fluide mais aussi, le travail de Nick Foster et Ron Fedkiw [33] qui introduit les PLS. L'utilisation d'une grille restreint la simulation à se dérouler dans un espace clos mais facilite l'extraction de la surface ou le calcul direct rayon-surface implicite. La correction globale de la pression rend difficile la parallélisation des méthodes Eulériennes (excepté les LBM) et donc l'implantation sur des calculateurs distribués. Les structures solides ou déformables sont pour la plupart du temps modélisées par une approche Lagrangienne. De ce fait, les interactions fluide-structure ne sont pas naturelles et nécessitent l'élaboration de modèles complexes.

Pour pallier ces inconvénients, des recherches en plein essor s'intéressent à la modélisation de fluide par des méthodes Lagrangiennes sans maillage, qui offrent des avantages "Lagrangiens" de parallélisation et de conservation de la masse, ainsi que des avantages "Eulérien" tels que la gestion implicite des changements de topologie.

Ces avantages sont au dépend d'un calcul non-trivial et coûteux du voisinage. Dans un souci d'optimisation, je me suis intéressé à cette problématique et propose un modèle hiérarchique dans le chapitre 5 qui accélère par trois fois en moyenne ce calcul du voisinage dans le cadre d'interactions.

Les interactions fluide-membranes étaient difficilement représentées avec les approches Eulériennes qui a été solutionné par le modèle d'Eran Guendelman *et al.* [43]. À ma connaissance, il n'y a pas de travaux en synthèse d'images qui proposent un modèle robuste d'interaction fluide Lagrangien et membranes déformables. Le récent modèle de Takahiro Harada *et al.* [44] et de Nobuhiro Kondoh *et al.* [58] suppose qu'une particule ne peut pas parcourir plus qu'une distance préétablie, ainsi avec un simple élargissement des triangles les contacts ne sont jamais manqués. Cette supposition n'est valable que dans certains cas précis et en pratique est assez contraignante. Dans ce cadre, je propose dans le chapitre 6 un modèle d'interaction entre fluide Lagrangien sans maillage et membranes déformables sans contraintes sur le déplacement des particules. Il est à noter que les interactions sont toujours répulsives, i.e. qu'elles empêchent les interpénétrations. Mais d'autres interactions, comme l'absorption par exemple, n'ont pas encore à ma connaissance été représentées.

La représentation de la surface est un point crucial synthèse d'images. La méthodes qui offre les résultats les plus réalistes est la PLS. L'extraction de surface des fonctions implicites est souvent effectuée par l'algorithme des *marching cubes* ou encore les *marching tetrahedras*. Pour une surface détaillée, il est plus judicieux, pour des raisons d'efficacité, de ne pas trianguler la surface mais de la calculer directement par un algorithme de *ray-marching* par exemple. Cependant, l'algorithme des *marching cubes* est toujours très sollicité et dans ce cadre je propose dans le chapitre 7 de corriger la surface extraite qui passe temporellement au travers des membranes en projettant le volume éronné le long des membranes.

Cependant, je reconnais que les *splats* offrent une alternative intéressante aux ensembles de triangles grâce à leur multi-résolution et affichent par ailleurs des résultats en temps interactifs, comme par exemple dans les travaux de Bart Adams *et al.* [8]. Récemment, Tim Weyrich *et al.* [113] se sont même intéressés à l'élaboration d'une carte graphique dédiée aux calculs de *splats*.

Chapitre Quatre

Stabilisation des méthodes d'intégration explicites

Ce chapitre aborde la stabilisation de méthodes d'intégration temporelle numériques explicites, appliquées aux systèmes masses-ressorts. Après une introduction à la problématique en section 4.1, un état de l'art survole les travaux relatifs à ce domaine en section 4.2. Trois procédés visant à stabiliser les méthodes d'intégrations sont abordés dans les sections suivantes :

1. le premier a pour but de supprimer la dépendance du pas d'intégration temporelle par la variation d'un paramètre θ qui ajuste automatiquement les forces du système. Malheureusement ce schéma ne fonctionne que pour une unique valeur de θ. Il est détaillé dans la section 4.3 ;

2. le deuxième est un procédé de filtrage qui stabilise sous une condition quantifiée les méthodes d'intégrations explicites. En effet, des analyses d'animations montrent que les hautes fréquences, c'est-à-dire des mouvements locaux et rapides, engendre des cas d'instabilités. Par conséquent, l'application d'un filtrage basse-fréquences va permettre de supprimer ces mouvements problématiques. Ce travail a fait l'objet d'une publication [4] et est détaillé en section 4.4 ;

3. un filtrage simpliste stabilise sous conditions les méthodes d'intégration explicites. Le troisième procédé consiste à améliorer ce filtrage en utilisant un filtre puissant, celui de Kalman. Ce dernier devrait permettre de stabiliser avec moins de contraintes, voir aucune contrainte, les méthodes d'intégration explicites. Les résultats obtenus et les difficultés rencontrées sont expliqués dans la section 4.5.

4.1 Introduction

La physique du mouvement est devenue un outils indispensable pour les artistes, par exemple, car elle facilite l'élaboration d'animations réalistes. Un modèle basé-physique très employé est le modèle masses-ressorts. Il est utilisé dans de nombreuses applications, notamment en synthèse d'images, telles que la simulation du déplacement de serpents et de vers proposée par Gavin Miller [70], ou encore pour la locomotion de poissons présenté par Xiaoyuan Tu et Demetri Terzopoulos [104], mais aussi pour reproduire la dynamique de fluides comme expliqué par Gavin Miller et Andrew Pearce [71], également pour l'animation de tissus telle que dans le modèle de David Baraff et Andrew Witkin [6] et de plus en plus pour la chirurgie médicale comme par exemple dans les travaux d'Anderson Maciel *et al.* [66].

Les systèmes masses-ressorts sont des systèmes de particules qui sont spatialement contraintes. Chaque particule, qui est définie à une position donnée, véhicule des informations telles que sa masse. Les ressorts sont des forces qui contraignent les particules connectées à conserver une distance prédéfinie. Les déformations d'objets sont obtenus par le déplacement de ces particules sous l'influence de forces extérieures. Pour calculer les changements de positions des particules, par le biais de ces forces, les équations d'un système masses-ressorts sont transformées en équations différentielles ordinaires (EDO).

Résoudre numériquement des EDOs n'est pas une tâche aisée. Les méthodes explicites sont les plus simples pour les résoudre mais comportent de mauvaises propriétés de stabilité : le pas d'intégration temporelle est souvent contraint à être petit, ce qui implique un nombre important d'itérations par image et donc de longs temps de calculs. Les schémas explicites sont dans de nombreux cas en pratique peu utilisés. Puisqu'il n'y a pas de restrictions sur le pas, la formulation implicite, bien que nécessitant la résolution d'un système linéaire, est plus efficace. Elle est donc en pratique très employée. La plupart des productions cinématographiques mettant en oeuvre des habits virtuels ont adopté cette formulation.

Pourquoi un schéma implicite est-il stable sans condition ? et pourquoi les méthodes explicites sont-elles stables sous de fortes contraintes ?

Une animation est un ensemble fini d'état ou d'images. Chaque image est un échantillon de temps qui représente un instant t pendant Δt secondes, intitulé pas d'intégration temporelle. Puisque les objets sont en mouvements, les nouvelles positions des particules des systèmes masses-ressorts sont calculées, ou plutôt approximées, à l'aide de leurs dérivées, c'est-à-dire les vitesses. Mais la vitesse est également une inconnue qui doit être déterminée par sa dérivée, c'est-à-dire l'accélération.

L'accélération est directement proportionnelle aux forces appliquées sur une particule à un instant précis. De grandes forces produisent de grands mouvements ou de fortes frictions. L'erreur engendrée par l'approximation de la dérivée augmente en fonction de Δt. De grandes erreurs sur de grandes valeurs engendrent une divergence, c'est-à-dire un cumul d'erreur qui empêche l'obtention d'un résultat correct.

Par conséquent, une attention particulière doit être portée sur les forces. Pour ce faire, je propose de décomposer l'équation mettant en oeuvre les forces en deux termes, c'est-à-dire linéaire et non-linéaire. Le terme linéaire, qui est sujet à de grandes variations, est résolu à l'aide d'un schéma implicite. Le terme non-linéaire, qui est borné, est résolu par un procédé explicite. Pour ne pas avoir de dépendances à Δt, un critère intermédiaire θ favorise les forces résolues implicitement ou explicitement en fonction d'un critère d'instabilité explicite. Ce procédé est détaillé dans la section 4.3.

Toujours dans l'optique de contrôler la variation des forces, je propose une analyse de la résolution d'un système linéaire issu d'une formulation implicite qui révèle la présence d'un filtrage des forces, dépendant de Δt. L'application de ce filtrage sur des méthodes explicites améliore leur stabilité. Plus de détails figurent dans la section 4.4.

J'ai mené une bref étude et des tests sur l'application d'un filtrage évolué par un filtre de Kalman, présentés en section 4.5.

4.2 État de l'art

Le calcul du déplacement des systèmes masses-ressorts repose sur la méthode d'intégration numérique choisie. Une vue d'ensemble des schémas d'intégration mis

en oeuvre en synthèse d'images figure dans les travaux de Pascal Volino et de Nadia Magnenat-Thalmann [107] ainsi que dans ceux de Michael Hauth *et al.* [47]. La stabilité et la précision sont les deux critères principaux pour choisir une méthode d'intégration. Comme indiqué dans l'état de l'art d'Andrew Nealen *et al.* [78], dans le domaine de l'animation basée-physique en informatique graphique la stabilité est souvent plus importante que la précision.

Les deux familles de méthodes d'intégration numériques, c'est-à-dire explicite et implicite, peuvent être divisées en deux catégories : les schémas à pas unique et ceux à pas multiples. Les méthodes à pas multiples ont recours à, au minimum, deux valeurs précédemment calculées pour obtenir la valeur suivante. Puisque les valeurs précédentes sont inconsistantes dans le cadre d'interactions utilisateurs ou bien de collisions, ces méthodes ne sont pas par conséquent adaptées au calcul d'animations. À l'exception des BDF2 (voir la section 2.3.2), la plupart des méthodes utilisées sont des modèles à pas unique.

Demetri Terzopoulos *et al.* [32, 96] ont intégré temporellement des déformations et des fractures d'objets dites "élastiques" et "inélastiques" par des méthodes implicites. La dynamique d'un fluide en 2 dimensions à été simulé en se basant également sur cette formulation stable par Michael Kass et Gavin Miller [53]. David Baraff et Andrew Witkin [6] ont obtenus d'excellent résultats en employant un modèle implicite dans le domaine de l'animation de tissus. Depuis, les améliorations de ce modèle se sont principalement intéressées aux optimisations des temps de calculs et à l'ajout de détails tels que les pliures. Pascal Volino et Nadia Magnenat-Thalmann [106] ont eu l'idée d'exploiter la propriété creuse de la matrice en ne s'intéressant qu'aux éléments non nuls et ont proposé une structure de données de matrices plus adaptée, qui est par conséquent plus efficace. Toujours dans l'optique d'accélérer la résolution du système linéaire, principalement composée de l'algorithme du gradient conjugué, Mark Meyer *et al.* [69] ont décomposé les forces en termes linéaires et non-linéaires et ont précalculé la matrice inverse du système dans le but d'obtenir des performances en temps réel. Les pliures résultent d'une forte rigidité intrinsèque au matériau et donc induisent des forces internes importantes. Le modèle élaboré par Kwang-Jin Choi et Hyeong-Seok Ko [19] simule avec stabilité et réalisme les pliures, notamment grâce à l'utilisation

de la méthode implicite du second ordre BDF2.

En ce qui concerne les schémas explicites, Gavin Miller [70] a adopté la méthode intitulée *forward* Euler pour animer des vers et des serpents. Les modèles déformables de Demetri Terzopoulos et Kurt Fleischer [96] sont intégrés temporellement par cette même formulation. Les schémas de Runge-Kutta, d'ordre 2 et plus, ont recours à des valeurs intermédiaires, engendrant une précision et une stabilité meilleure. L'intégration de Verlet, aussi connue sous d'autres appellations, est probablement la méthode la plus employée en animation basée physique car, tout en conservant la simplicité de la méthode de *forward* Euler, elle est plus stable. À titre d'exemple, elle est la méthode choisie par le studio cinématographique Industrial Light & Magic[1] d'après les travaux de Robert Bridson *et al.* [12] sur les tissus et de Zoran Kačić-Alesić *et al.* [54] pour les systèmes masses-ressorts. Elle est aussi la méthode la plus sollicitée dans les jeux vidéo.

Des travaux ont porté sur la fusion des deux grandes catégories IMplicite/EXplicite, c'est-à-dire IMEX, pour profiter des avantages de chacune. Xiaoyuan Tu et Demetri Terzopoulos [104] ont défini un squelette de poisson constitué de masses et de ressorts. Puisque ce squelette est un modèle unique, la structure de données de la matrice du système linéaire est ajustée pour obtenir des temps de calculs optimaux. Entre autres, le système linéaire est résolu explicitement dans l'espace et implicitement dans le temps. Un gain en performances est issu de cette décomposition. Bernhard Eberhardt *et al.* [22] et Robert Bridson *et al.* [14] ont dissociés les forces en termes linéaires et non-linéaires. Le terme linéaire, qui est sujet à des instabilités, est résolu par une méthode implicite. Le terme non-linéaire est donc calculé par une formulation explicite allégeant ainsi la résolution numérique du système linéaire. Eddy Boxerman et Uri Ascher [10] ont défini un seuil d'instabilité pour les méthodes explicites. Lorsque celui-ci est détecté, le système est résolu par un procédé implicite, puis rebascule à une formulation explicite lorsque la phase d'instabilité est terminée. Les auteurs séparent la matrice du système linéaire en matrices plus petites, en utilisant un procédé de décomposition de graphe. Chaque matrice réduite étant résolue indépendamment,

[1] http ://www.ilm.com

les calculs d'animations sont alors plus rapides.

Hormis le fait que les recherches s'intéressent d'avantages aux approches implicites et IMEX, les méthodes explicites sont toujours d'actualité, comme démontré par Zoran Kačić-Alesić *et al.* [54]. Le problème principal des formulations explicites est leur dépendance à la valeur propre maximale des vecteurs de forces, qui est dépendante des coefficients de raideurs des ressorts. Les systèmes disposant d'une raideur importante, dits systèmes "raides", ne sont résolus efficacement qu'avec une méthode implicite. Mikio Shinya [90] propose de résoudre explicitement les Jacobiennes des forces. Les calculs matriciels sont effectués soit à intervals réguliers où soit lorsque une divergence est détectée mais pas à chaque itération. Ainsi faisant, cette méthode stable est plus performante qu'une méthode implicite en terme de temps de calculs. Cependant, dans l'article précédemment cité, la détection de la divergence n'est pas bien définie. Dans la même optique de stabiliser les méthodes explicites, trois propositions figurent dans les sections suivantes :

1. une balance des forces par un θ-schéma en section 4.3 ;
2. un filtrage "correcteur" en section 4.4 ;
3. un autre filtre "correcteur" mais plus évolué en section 4.5.

4.3 Stabilisation par un θ-schéma

L'objectif de cette recherche est d'obtenir une méthode d'intégration numérique stable et rapide. Pour ce faire, ce travail propose de supprimer la dépendance des forces au pas d'intégration temporelle et d'utiliser au maximum une intégration explicite. La sous-section 4.3.1 explique la division des forces en deux termes, c'est-à-dire un terme linéaire et un terme non-linéaire. Le terme linéaire est sujet à de grandes variations, il est par conséquent résolu par un procédé implicite. Le terme non-linéaire, qui est borné, est moins sujet à des instabilités et il est par conséquent résolu par un procédé explicite. La sous-section 4.3.2 décrit la variation du paramètre θ qui équilibre les deux termes. En fonction d'un critère d'instabilité du système, θ va favoriser le terme linéaire pour assurer la convergence de la méthode. Lorsque la phase d'in-

stabilité est terminée, θ favorise d'avantage le terme non-linéaire afin de ne résoudre les équations que par un procédé explicite. Ainsi, la méthode peut être optimale en terme de temps de calculs tout en étant stable, quelque soit le pas. Le paramètre θ sera donc responsable de l'ajustement des forces en cas d'instabilités.

4.3.1 Décomposition des forces

Soit l'équation différentielle suivante dans le cas général :

$$\dot{y} = u(t, y(t)) \tag{4.1}$$

Cette équation peut être décomposée en deux parties, soit :

$$\begin{aligned} \dot{y} &= f(t, y(t)) + g(t, y(t)) \\ \dot{y} &= \theta \cdot f(t, y(t)) + (1 - \theta) \cdot g(t, y(t)) \end{aligned} \tag{4.2}$$

où $f()$ et $g()$ sont respectivement des termes linéaires et non-linéaires et $\theta \in [0 \dots 1]$. En ce qui concerne les systèmes masses-ressorts, l'équation 9.7 (dans les annexes) qui définie l'attraction et la répulsion entre deux particules produit les plus grandes forces internes du système. Puisque l'amortissement n'augmente pas les forces internes mais au contraire va les réduire, il n'est pas pris en compte dans ce θ-schéma. La force d'attraction et de répulsion est décomposée en développant l'équation 9.7 :

$$\begin{aligned} \mathbf{f}^{raide}(\mathbf{x}) &= k \cdot \mathbf{x} - \tfrac{k \cdot l \cdot \mathbf{x}}{\|\mathbf{x}\|} \\ \mathbf{f}^{raide}(\mathbf{x}) &= \theta \cdot \mathbf{f}(\mathbf{x}) + (1 - \theta) \cdot \mathbf{g}(\mathbf{x}) \end{aligned} \tag{4.3}$$

avec

$$\mathbf{f}(\mathbf{x}) = k \cdot \mathbf{x} \tag{4.4}$$

$$\mathbf{g}(\mathbf{x}) = -\frac{k \cdot l \cdot \mathbf{x}}{\|\mathbf{x}\|} \tag{4.5}$$

4.3.2 Variation de θ et condition de stabilité

Lorsqu'une instabilité du système est détectée, alors $\theta = 1$, sinon $\theta = 0$. Une transition douce de ce paramètre va permettre de basculer des forces linéaires aux forces non-linéaires sans changement brusque. Après diverses tentatives d'élaboration d'un critère de stabilité, j'ai adopté la condition présentée dans les travaux d'Eddy Boxerman [9], qui a d'ailleurs été développée dans des temps parallèles :

$$\frac{\Delta t}{m}(\Delta t k + 2d) \leq 0.5 \tag{4.6}$$

où k représente la rigidité du système, voir la sous-section 4.4.3 et d représente l'amortissement du système. Si tous les ressorts disposent des mêmes coefficients de raideur et d'amortissement, alors k et d représentent respectivement les valeurs de rigidité et d'amortissement de ces ressorts. Le paramètre θ est donc associé à l'équation 4.6 avec une mise à l'échelle.

4.3.3 Application et résultats

Ce schéma appliqué aux systèmes masses-ressorts n'est fonctionnel que pour une seule valeur, $\theta = 0.5$. Les forces linéaires et non-linéaires ne sont malheureusement pas indépendantes :

- lorsque le terme $\mathbf{f}()$ est privilégié, les objets subissent un agrandissement de k, car en fait ce terme est en quelque sorte la répulsion ;
- inversement lorsque le terme $\mathbf{g}()$ est favorisé, les structures rétrécissent car ce terme représente grossièrement l'attraction.

Les objets rétrécissent et s'agrandissent en fonction de la variation θ, ce qui n'est pas du tout le résultat escompté.

4.3.4 Conclusion

L'élaboration d'une méthode d'intégration temporelle efficace n'est pas possible suivant le procédé décrit dans cette section. Cette nouvelle méthode proposée est instable à cause du terme calculé explicitement, de plus elle oblige la résolution d'un

système linéaire à cause du terme linéaire. Cette méthode n'a donc à priori aucun avenir.

4.4 Stabilisation conditionelle par un filtrage exponentiel

Cette section présente l'application d'un filtrage exponentiel sur des méthodes explicites. Des comparaisons montrent que les animations calculées par ces schémas contiennent des hautes fréquences, tandis que les mêmes animations calculées avec une méthode implicite contiennent des fréquences plus basses. Des analyses de la résolution du système linéaire induit par la formulation implicite révèlent qu'un filtrage exponentiel est appliqué automatiquement sur les forces. Ce filtrage est appelé *amortissement artificiel*. L'application d'un filtrage similaire sur des méthodes explicites permet de préserver leur simplicité et d'améliorer leur stabilité, donc de diminuer les temps de calculs. De plus, le filtrage est indépendant de la méthode d'intégration numérique utilisée et peut donc être employé en post-traitement sans avoir à modifier le coeur de la méthode.

Des expériences sur les schémas d'intégration classiques révèlent une réduction des temps de calculs d'animations de 20% en moyenne, incluant un coût de filtrage de 5%. Pour des maillages modestement rigides, les temps de calculs avec cette approche et un modèle implicite sont proches. La contre partie de ce filtrage est l'ajout d'un amortissement global. Les animations résultantes sont par conséquent plus "adoucies", ce changement est visible à l'oeil nu. Des expérimentations sur des maillages réguliers et homogènes, tels que des tissus et irréguliers avec des coefficients hétérogènes, tels qu'un modèle de poisson, sont présentées.

Une analyse du vecteur des forces indique que le filtrage réduit leur valeur propre maximale qui est directement responsable de la taille du pas d'intégration temporelle. Suite à cette analyse, une condition de stabilité est définie, et selon certains cas, elle indique que le pas d'intégration temporelle peut être doublé en utilisant un filtrage puissant.

La section 4.4.1 décrit le problème de la propagation des forces. La section 4.4.2 explique l'effet du filtrage exponentiel appliqué sur des méthodes explicites. Des analyses fréquentielles, la détermination d'un critère de stabilité et des comparaisons de temps de calculs figurent dans la section 4.4.4. Les conclusions et les travaux futurs sont discutés en section 4.4.5.

4.4.1 Définition du problème et d'une solution

Cette section définie un des problèmes rencontré par la résolution explicite d'EDOs, appliqués au système masses-ressorts, c'est-à-dire la propagation des forces. Ce problème est également expliqué dans les travaux de Michael Kass [52] et de Mark Meyer *et al.* [69]. La figure 4.1 illustre le cas unidimensionnel d'une corde.

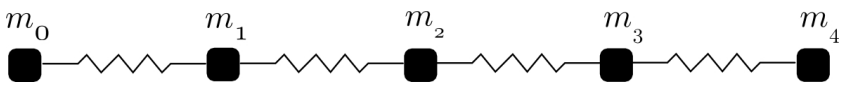

FIG. 4.1: Un système masses-ressorts 1D.

Un ensemble de particules uniformément espacées $\{m_0, \ldots, m_4\}$, de masse m identique, sont connectées par des ressorts avec un coefficient de raideur $km > 0$. À t_0 aucune force n'est appliquée sur le système, les ressorts sont à leur longueur de repos, c'est-à-dire que le système est en équilibre statique. Si km est petit et qu'une force est exercée sur la particule m_0, alors celle-ci devrait se déplacer librement et n'affecter que légèrement les autres particules. En revanche si la valeur de km est importante, du fait de la propagation des forces, la structure entière devrait se déplacer instantanément.

4.4.1.1 Système d'EDO

La mécanique des systèmes masses-ressorts peut être formulée comme une équation différentielle de la forme :

$$\mathbf{M}\ddot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}) \tag{4.7}$$

où $\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}$ représentent respectivement les vecteurs de position, de vitesse et d'accélération de taille $3n$ (dans un repère 3D), n représente le nombre de particules, $\mathbf{f}$ est la fonction de forces et $\mathbf{M}$ est une matrice diagonale contenant les masses (de taille $3n \times 3n$, c'est-à-dire $diag(\mathbf{M}) = m_1, m_1, m_1, m_2, m_2, m_2, \ldots$). En faisant le changement de variable suivant, $\mathbf{v} = \dot{\mathbf{x}}$, l'équation 4.7 est transformée en un système d'EDO :

$$\frac{d}{dt}\begin{pmatrix} \mathbf{x} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{v} \\ \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}, \mathbf{v}) \end{pmatrix} \tag{4.8}$$

La section suivante analyse le comportement des schémas explicite et implicite appliqués à l'équation 4.8.

4.4.1.2 Un des problèmes de l'intégration explicite

La formule de la méthode *forward* Euler est la suivante :

$$\begin{pmatrix} \mathbf{x}^{t+\Delta t} \\ \mathbf{v}^{t+\Delta t} \end{pmatrix} = \begin{pmatrix} \mathbf{x}^{t} \\ \mathbf{v}^{t} \end{pmatrix} + \Delta t \begin{pmatrix} \mathbf{v}^{t} \\ \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}^{t}, \mathbf{v}^{t}) \end{pmatrix} \tag{4.9}$$

Connaissant les conditions initiales, l'équation 4.9 est résolue itérativement. Les ressorts agissent uniquement sur les deux particules connectées aux extrémités. Dans cette équation $\mathbf{v}^t$ est utilisée pour mettre à jour $\mathbf{x}^{t+\Delta t}$ et non $\mathbf{x}^t$ (une extension simple connue sous le nom de *forward-backward* (FB) Euler tire profit de $\mathbf{v}^{t+\Delta t}$). Par conséquent un déplacement de la particule m_0 affectera m_1 après deux itérations. Les forces agissant sur m_0 sont donc propagées sur m_4 après $2 \times (n-1)$ itérations. Il en résulte que les particules ont une certaine liberté de mouvement sans effet immédiat sur l'ensemble de la structure, c'est-à-dire des variations locales. La propagation des forces devrait être instantanée, ce qui est le cas avec la méthode de *forward* Euler qu'après $2 \times (n-1)^2$ itérations par pas d'intégration temporelle. Les temps de calculs liés à cette complexité sous forme quadratique sont alors très long, les méthodes explicites ne sont donc pas utilisables dans beaucoup de cas en pratique. La formulation implicite vient pallier ce problème de propagation de forces.

4.4.1.3 La solution implicite

La méthode intitulée *backward* Euler est la plus connue des formulations implicites. Elle est définie comme suit :

$$\begin{pmatrix} \mathbf{x}^{t+\Delta t} \\ \mathbf{v}^{t+\Delta t} \end{pmatrix} = \begin{pmatrix} \mathbf{x}^{t} \\ \mathbf{v}^{t} \end{pmatrix} + \Delta t \begin{pmatrix} \mathbf{v}^{t+\Delta t} \\ \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}^{t+\Delta t}, \mathbf{v}^{t+\Delta t}) \end{pmatrix} \tag{4.10}$$

Deux variables inconnues, c'est-à-dire $\mathbf{x}^{t+\Delta t}$ et $\mathbf{v}^{t+\Delta t}$ sont à prendre en compte lors de l'évaluation de la fonction de forces. Cette fonction $\mathbf{f}(\mathbf{x}^{t+\Delta t}, \mathbf{v}^{t+\Delta t})$ est donc approximée en calculant ses dérivées, comme présenté par David Baraff et Andrew Witkin [6]. En posant :

$$\Delta\mathbf{x} = \mathbf{x}^{t+\Delta t} - \mathbf{x}^{t} \quad \Delta\mathbf{v} = \mathbf{v}^{t+\Delta t} - \mathbf{v}^{t}$$

$$\mathbf{f}(\mathbf{x}^{t} + \Delta\mathbf{x}, \mathbf{v}^{t} + \Delta\mathbf{v}) = \mathbf{f}^{t} + \frac{\partial\mathbf{f}}{\partial\mathbf{x}}\Delta\mathbf{x} + \frac{\partial\mathbf{f}}{\partial\mathbf{v}}\Delta\mathbf{v}$$

Les dérivées partielles, ou matrices Jacobiennes, des équations des forces 9.7 et 9.8 sont les suivantes :

$$\left(\frac{\partial\mathbf{f}}{\partial\mathbf{x}}\right)_{ij} = \left[\frac{\mathbf{x}_{ij} * \mathbf{x}_{ij}^{\mathbf{T}}}{\mathbf{x}_{ij}^{\mathbf{T}} * \mathbf{x}_{ij}} + \left(\mathbf{I} - \frac{\mathbf{x}_{ij} * \mathbf{x}_{ij}^{\mathbf{T}}}{\mathbf{x}_{ij}^{\mathbf{T}} * \mathbf{x}_{ij}}\right) * \left(1 - \frac{L}{|\mathbf{x}_{ij}|}\right)\right] \times k$$

$$\left(\frac{\partial\mathbf{f}}{\partial\mathbf{v}}\right)_{ij} = \mathbf{I} \times d$$

en effectuant le changement de variable suivant :

$$\Delta\mathbf{x} = \Delta t \cdot (\mathbf{v}^{t} + \Delta\mathbf{v})$$

l'équation est de la forme :

$$\left(\mathbf{M} - \Delta t\frac{\partial\mathbf{f}}{\partial\mathbf{v}} - \Delta t^{2}\frac{\partial\mathbf{f}}{\partial\mathbf{x}}\right)\Delta\mathbf{v} = \Delta t\left(\mathbf{f}^{t} + \Delta t\frac{\partial\mathbf{f}}{\partial\mathbf{x}}\mathbf{v}^{t}\right)$$

qui peut se réduire à :

$$\mathbf{A}\Delta\mathbf{v} = \mathbf{b} \tag{4.11}$$

où $\mathbf{A}$ est une matrice, souvent creuse, symétrique et définie positive (ou transformée pour l'être), $\Delta\mathbf{v}$ est le vecteur de vitesse cherché et $\mathbf{b}$ est un vecteur temporaire. Il existe différentes méthodes pour résoudre les systèmes linéaires dont les plus connues figurent dans l'ouvrage de William Press *et al.* [83]. Pour en nommer quelques unes, la factorisation de Cholesky et la méthode du gradient conjugué sont les plus utilisées en animation, avec une grande préférence pour cette dernière due à sa vitesse de résolution qui peut même être accélérée par des préconditions bien déterminées (comme par exemple dans les travaux d'Eddy Boxerman et de Uri Ascher [10])[2]. La suite des analyses est basée sur la factorisation de Cholesky car elle est plus simple à étudier.

Selon cette factorisation, la matrice $\mathbf{A}$ est décomposée en deux matrices triangulaires, c'est-à-dire une inférieure $\mathbf{L}$ et sa transposée $\mathbf{L^T}$.

$$\mathbf{A} = \mathbf{L}\mathbf{L^T} \tag{4.12}$$

En appliquant ce procédé au système illustré dans la figure 4.1, $\mathbf{A}$ et $\mathbf{L}$ ont les formes suivantes :

$$\mathbf{A} = \begin{pmatrix} a_{0,0} & a_{1,0} & & & \mathbf{0} \\ a_{1,0} & a_{1,1} & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & a_{3,3} & a_{3,4} \\ \mathbf{0} & & & a_{4,3} & a_{4,4} \end{pmatrix} \mathbf{L} = \begin{pmatrix} l_{0,0} & & & & \mathbf{0} \\ l_{1,0} & l_{1,1} & & & \\ & l_{2,1} & \ddots & & \\ & & \ddots & l_{3,3} & \\ \mathbf{0} & & & l_{4,3} & l_{4,4} \end{pmatrix} \tag{4.13}$$

La solution $\Delta\mathbf{v}$ est obtenue par une substitution *forward* (équation 4.14) et *backward* (équation 4.15).

[2]la méthode du gradient conjugué est assez complexe. Le document à l'adresse suivante, http ://www.cs.cmu.edu/ quake-papers/painless-conjugate-gradient.pdf, fourni une explication détaillée.

$$\mathbf{Ly} = \mathbf{b} \tag{4.14}$$

$$\mathbf{L^T}\Delta\mathbf{v} = \mathbf{y} \tag{4.15}$$

L'analyse de ce calcul permet de comprendre le procédé mis en oeuvre par les méthodes implicites pour propager les forces. En considérant l'équation 4.14, appliquée à notre exemple, le premier élément est simple à calculer, c'est-à-dire $y_0 = b_0/l_{0,0}$. Les autres éléments sont calculés avec la récurrence suivante : $y_i = (b_i - y_{i-1} \cdot l_{i,i-1})/l_{i,i}$. Comme indiqué par Michael Kass [52], le cas où $l_{i,i-1} = 1/a$ et $l_{i,i} = (a-1)/a$ représente des valeurs usuelles en dehors des extrémités. Cette récurrence peut être récrite sous la forme suivante :

$$y_i = \lambda \cdot b_i + (1-\lambda) \cdot y_{i-1} \tag{4.16}$$

où $\lambda = a \cdot (a-1)^{-1}$. L'équation 4.16 est un filtre récursif. La nouvelle valeur y_i est une pondération entre la valeur précédente y_{i-1} et l'élément courant b_i. Les valeurs précédentes représentent les forces des particules voisines connectées par un ressort. Une pondération est donc effectuée entre les différentes particules connectées. C'est en fait un filtre passe-bas avec un noyau de lissage déterminé par λ (qui est proportionnelle à Δt).

Un filtrage passe bas est imbriqué dans la résolution du système linéaire. En d'autres termes les particules propagent leurs forces à leurs voisines, aux voisines des voisines, etc., et ce, durant un unique pas d'intégration. Plus le pas est grand, plus le filtrage est important et inversement. Par conséquent la propagation des forces n'est plus un problème de stabilité. En prenant avantage de ce procédé, je propose d'effectuer un filtrage identique sur des systèmes masses-ressorts dont le déplacement est calculé par des méthodes explicites, afin d'autoriser des pas d'intégration temporelle plus grands.

4.4.2 Filtrage de méthodes d'intégration explicites

En réponse au problème de propagation des forces, la formulation implicite diffuse, lorsque c'est nécessaire, les forces des particules adjacentes. Ainsi les variations locales, c'est-à-dire les hautes fréquences, sont atténuées et donc la stabilité est améliorée.

Cette section détaille une réponse possible au problème de la propagation des forces, destinée pour les méthodes explicites. Pour ce faire, un filtrage est mis en oeuvre. Mais contrairement à une formulation implicite où la force est propagée, le modèle présenté ici propage la vélocité. Les expérimentations révèlent que le filtrage des forces offre des résultats similaires, ce qui n'est pas surprenant puisque les vitesses sont calculées en utilisant ces mêmes forces. Filtrer les vélocités permet d'appliquer cet algorithme en post-traitement, sans avoir à modifier le modèle d'intégration existant. La méthode d'intégration numérique est donc indépendante du modèle de filtrage. Les résultats illustrés dans la figure 4.9 indiquent que l'augmentation de la stabilité de la méthode est dépendante du noyau de lissage du filtre.

L'idée est de calculer la vélocité d'une particule en fonction d'une pondération de ses voisines. Pour ce faire, la nouvelle vitesse est en premier lieu calculée par une méthode d'intégration numérique quelconque. En second lieu, la vélocité est modifiée en fonction des vitesses voisines (voir l'algorithme 1). En dernier lieu les positions sont mises à jour.

4.4.2.1 Filtrage proposé

La résolution du système linéaire issu d'une formulation implicite induit un filtrage exponentiel. Par conséquent, dans le souci de reproduire le même comportement, un filtrage identique est appliqué sur les méthodes explicites. Les expérimentations avec une fonction gaussienne ont montré des résultats similaires, c'est-à-dire des augmentations du pas d'intégration dans les mêmes proportions et un amortissement du mouvement visuellement comparable. Le filtre exponentiel est défini par :

$$\beta(d) = \lambda \cdot e^{-\lambda|d|} \tag{4.17}$$

où d est la distance algébrique entre deux particules. La figure 4.2 illustre le paramétrage du filtre en fonction du coefficient λ. Un filtrage adaptatif est donc possible via la variation de ce coefficient. Il est important de noter qu'un filtrage excessif, c'est-à-dire lorsque $\lambda < 1$, engendre des effets indésirables avec, notamment, un mouvement trop amorti. Dans ce cas les vitesses des particules voisines ont trop d'influence et réduisent considérablement l'action de la particule filtrée. Il n'y a alors plus de variations locales, ne subsiste qu'un mouvement global.

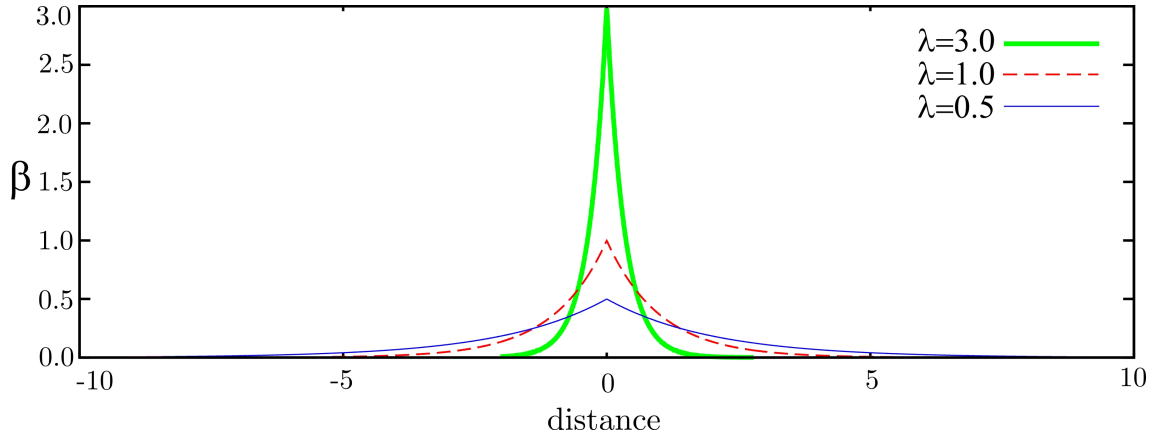

FIG. 4.2: Le filtre exponentiel

En considérant la i-ème particule, sa nouvelle vitesse filtrée est calculée comme suit :

$$\mathbf{v}_{i_{filtre}}^{t+\Delta t} = \left(\lambda \cdot \mathbf{v}_i^{t+\Delta t} + \sum_{j \in voisins(i)} \beta_j^{t+\Delta t} \cdot \mathbf{v}_j^{t+\Delta t} \right)$$

où $\beta_j^{t+\Delta t}$ est la pondération des valeurs voisines. Les coefficients $\beta_j^{t+\Delta t}$ varient au cours du temps et doivent être normés. Cependant l'échange entre deux particules i et j n'est pas symétrique et peut produire des effets indésirables car la conservation du moment angulaire n'est pas assurée. Ce problème est corrigé par la formule suivante :

$$\mathbf{v}_{i_{filtre}}^{t+\Delta t} = \left(\lambda \cdot \mathbf{v}_i^{t+\Delta t} + \sum_{j \in voisins(i)} \beta_j^{t+\Delta t} \frac{\mathbf{v}_j^{t+\Delta t} + \mathbf{v}_i^{t+\Delta t}}{2} \right)$$

L'algorithme présenté dans cette section est semblable à celui proposé dans Mark Meyer *et al.* [69]. Il y a cependant des différences notables : les auteurs proposent de simplifier la matrice **A** du système linéaire, une matrice inverse est précalculée et sert de filtrage pour les forces. Leur but est d'accélérer la résolution implicite. Je propose un filtre générique pour les méthodes explicites qui permet d'agrandir le pas d'intégration temporelle, diminuant ainsi les temps de calculs. Je présente en plus une analyse fréquentielle.

Algorithme 1 L'algorithme de filtrage

Nécessite: vecteur vel, vecteur pos, λ

initialisation du vecteur 3D v^{filtre}
pour tout particule i **faire**
  $norme \leftarrow \lambda$
  $v_i^{filtre} \leftarrow \lambda \times v_i$ {la particule courante}
  pour tout particule $j \in voisins(i)$ **faire**
    $d \leftarrow dist(x_i, x_j)$ {calcul des distances}
    $\beta = \lambda \times e^{-\lambda|d|}$ {calcul du coefficient}
    $v_i^{filtre} \leftarrow v_i^{filtre} + \beta \times (v_j + v_i)$ {filtrage}
    $norme \leftarrow norme + \beta$
  fin pour
  $v_i^{filtre} \leftarrow v_i^{filtre} / (2 \times norme)$ {mise à l'échelle}
fin pour
return v^{filtre}

4.4.2.2 Structure du modèle

Des expérimentations ont été effectuées avec plusieurs structures constituées de masses et de ressorts. Le modèle de Kwang-Jin Choi et Hyeong-Seok Ko [19] a été utilisé pour la simulation de tissus. Des objets tridimensionnels avec des coefficients hétérogènes, tels que le modèle de poisson de Xiaoyuan Tu et Demetri Terzopoulos [104], ont également été expérimentés.

4.4.2.3 Mise en oeuvre

La mise en oeuvre de cette méthode est facile pour des calculateurs d'EDO existants. Le filtrage s'est montré opérationnel pour toutes les méthodes explicites testées,

c'est-à-dire : *forward* Euler, FB Euler, Verlet et Runge-Kutta 4 (RK4).

Dans toutes les expérimentations, les forces de friction avec le fluide environnant définies par Xiaoyuan Tu et Demetri Terzopoulos [104] ont été utilisées. Les forces des ressorts mises en oeuvre figurent dans les équations 9.7 et 9.8.

Les forces d'amortissement des ressorts ne sont pas nécessaires puisque le filtrage produit implicitement un amortissement. Il est dans un tel cas plus judicieux d'avoir recours au modèle dit de l'amortissement projeté (voir équation 9.10), car le filtrage élimine les hautes fréquences qui ont lieu dans l'axe du ressort mais aussi dans les autres axes, tandis que l'amortissement projeté atténue les oscillations uniquement dans l'axe des ressorts. Ainsi, un double amortissement est effectué dans l'axe des ressorts, ce qui n'est pas très gênant en pratique car ces oscillations sont souvent indésirables.

4.4.2.4 Analyses du filtrage

Dans cette sous-section figurent des analyses fréquentielles d'animations générées par des méthodes explicites, implicites et explicites avec filtrage. Les expériences suivantes démontrent que puisque les méthodes explicites donnent lieu à des variations locales, les animations résultantes sont riches en détails, tandis que les mouvements calculés par une approche implicite ou explicite filtrée sont plus lissés, ou amortis, car les hautes fréquences sont éliminées.

Soit l'expérience suivante : un maillage représentant un mouchoir en tissu, attaché en deux points, est soumis aux forces environnantes (gravité et air) pendant 5 secondes (illustrée dans les figures 4.4, 4.5, 4.6 et 4.7). Trois simulations sont testées avec respectivement les méthodes suivantes : FB Euler (explicite), *backward* Euler (implicite) et FB Euler filtrée (FFB Euler - explicite filtrée). Aucun algorithme de pas adaptatif n'est utilisé et les trois simulations sont identiquement paramétrées. Pour analyser les vitesses, le champ de vélocité est converti de l'espace 3D à l'espace RGB. La figure 4.3 montre la transformation d'un maillage 2D en tableau 1D.

Dans la figure 4.8[3], des blocs de couleurs similaires coïncident temporellement

[3]l'axe horizontal représente le temps avec un pas de 0.01 seconde.

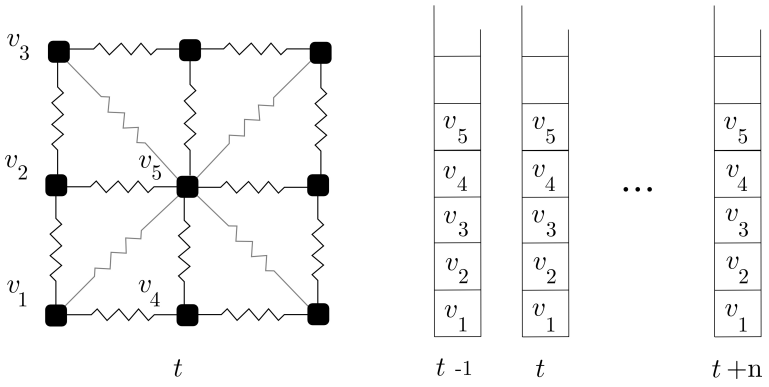

FIG. 4.3: À chaque pas d'intégration, les vitesses sont stockées dans un tableau unidimensionnel.

dans les trois images (a)(b)(c). L'image (a), qui représente un mouvement intégré avec une méthode explicite, contient beaucoup de perturbations. Celles-ci indiquent la présence de variations locales, c'est-à-dire de hautes fréquences. Comme prédit, le tracé de (b), qui représente un mouvement intégré avec schéma implicite, contient peu de perturbations puisqu'elles sont automatiquement filtrées. La résolution du système linéaire en utilisant la décomposition de Cholesky et la méthode du gradient conjugué produisent des résultats identiques et à l'évidence, avec ces deux procédés, un filtrage est généré. La figure (c) représente un mouvement intégré par la méthode FFB Euler. L'animation résultante est visuellement proche de celle calculée par la formulation implicite. Le filtrage est donc la cause de l'élimination des hautes fréquences.

Afin de visualiser le domaine fréquentiel du mouvement, une transformée discrète de Fourier 2D (DCT) est effectuée sur les vitesses du maillage en réalisant les mêmes expérimentations. Des captures d'écran des animations sont visibles dans le figures 4.5, 4.6 et 4.7. La valeur au centre des images de la ligne du bas, c'est-à-dire le domaine fréquentiel, indique la présence de basses fréquences. Les valeurs éloignées du centre correspondent aux hautes fréquences. Encore une fois les résultats répondent aux attentes : l'animation calculée avec la méthode de FB Euler (figure 4.5) contient des fréquences plus élevées que les deux autres, tandis que l'animation calculée avec la méthode implicite (figure 4.6) comporte des fréquences plus basses (près du centre). Des résultats visuellement similaires au comportement implicite sont obtenus en fil-

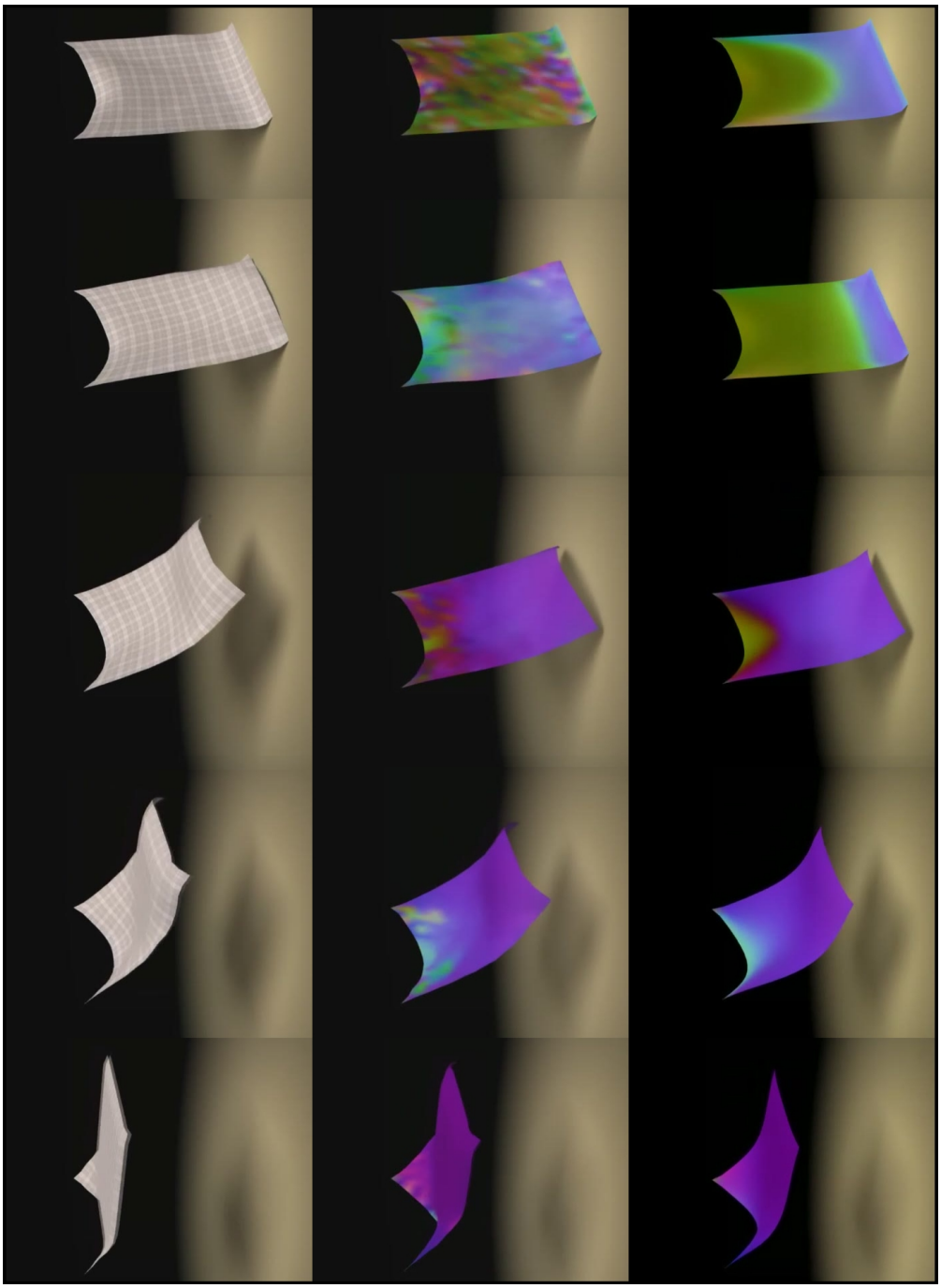

FIG. 4.4: *Haut-en tournant la page de 90°- :*animation d'un tissu soumis aux forces environnantes, *milieu :* la même expérimentation intégrée avec *forward* Euler, les couleurs représentent les vitesses, *bas :* le filtrage est activé et le mouvement est plus adouci.

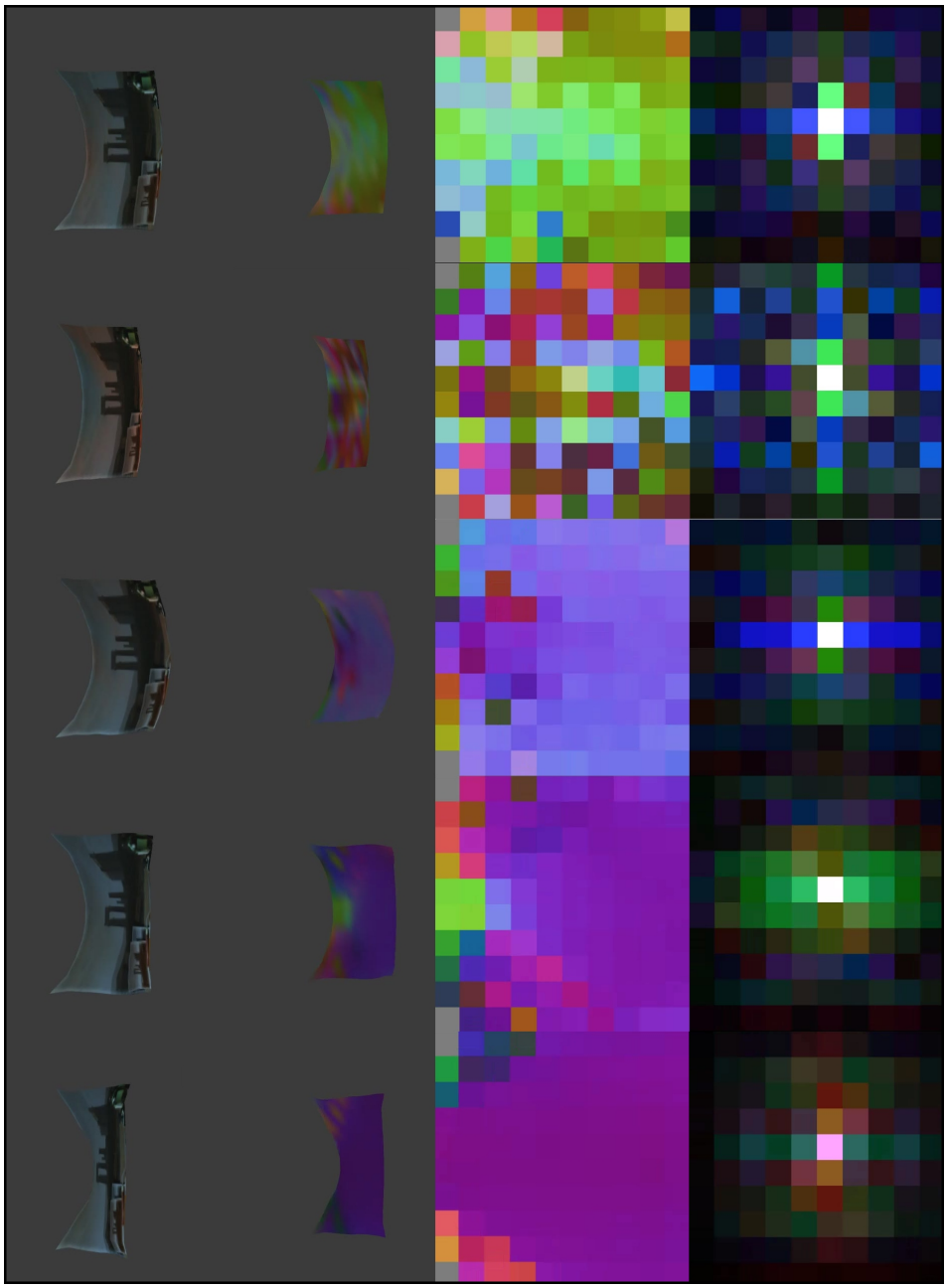

FIG. 4.5: *Première ligne en partant du haut et en tournant la page :*un drap texturé est soumis à la gravité en utilisant une méthode d'intégration numérique explicite, *deuxième ligne :* les couleurs représentent les vitesses, *troisième ligne :* qui sont affichées dans le même plan, *quatrième ligne :* le domaine fréquentiel obtenu par une transformée de Fourier. Les images coïncident temporellement de haut en bas.

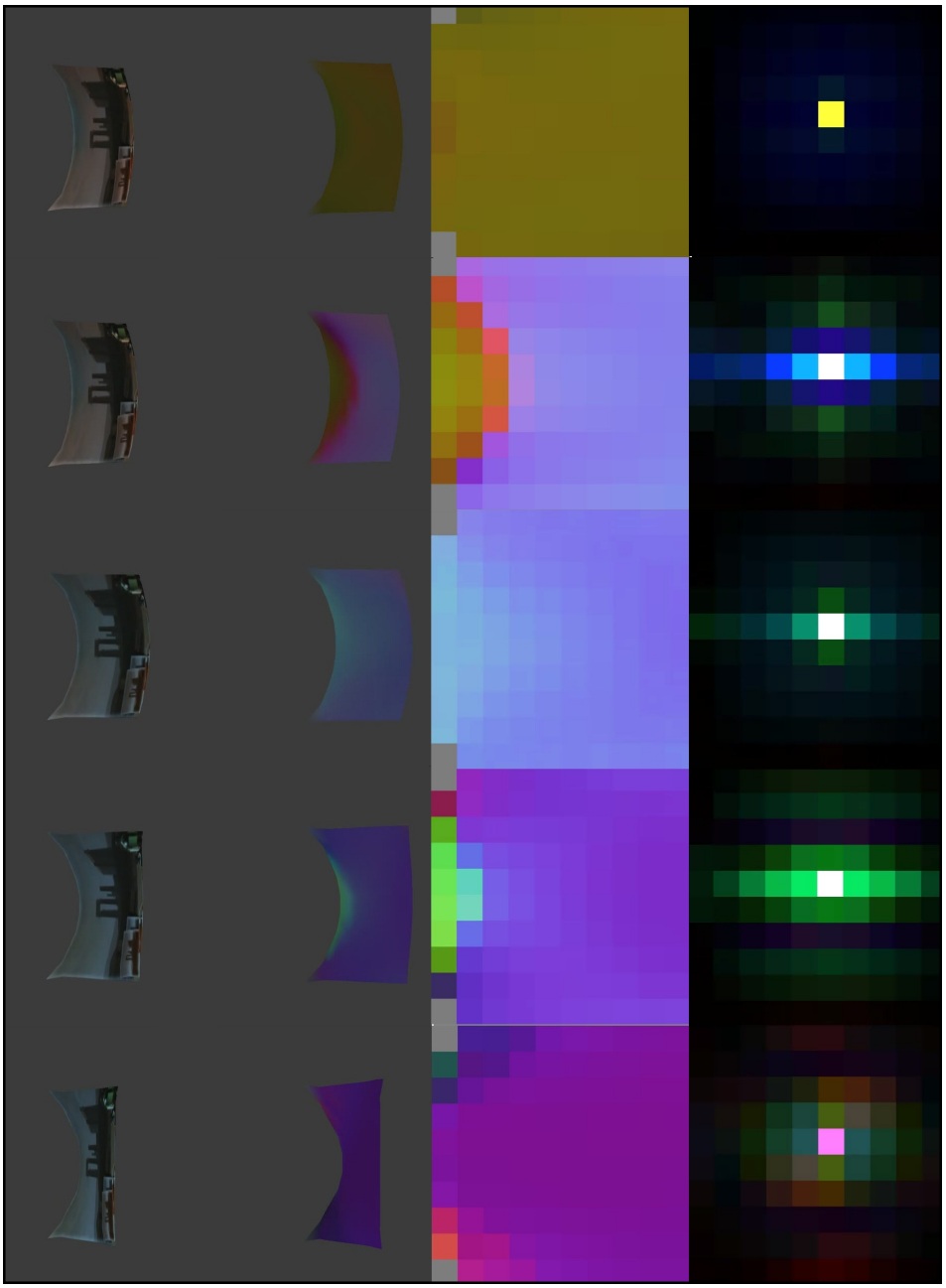

FIG. 4.6: *Première ligne en partant du haut et en tournant la page :*un drap texturé est soumis à la gravité en utilisant une méthode d'intégration numérique implicite, *deuxième ligne :* les couleurs représentent les vitesses, *troisième ligne :* qui sont affichées dans le même plan, *quatrième ligne :* le domaine fréquentiel obtenu par une transformée de Fourier. Les images coïncident temporellement de haut en bas.

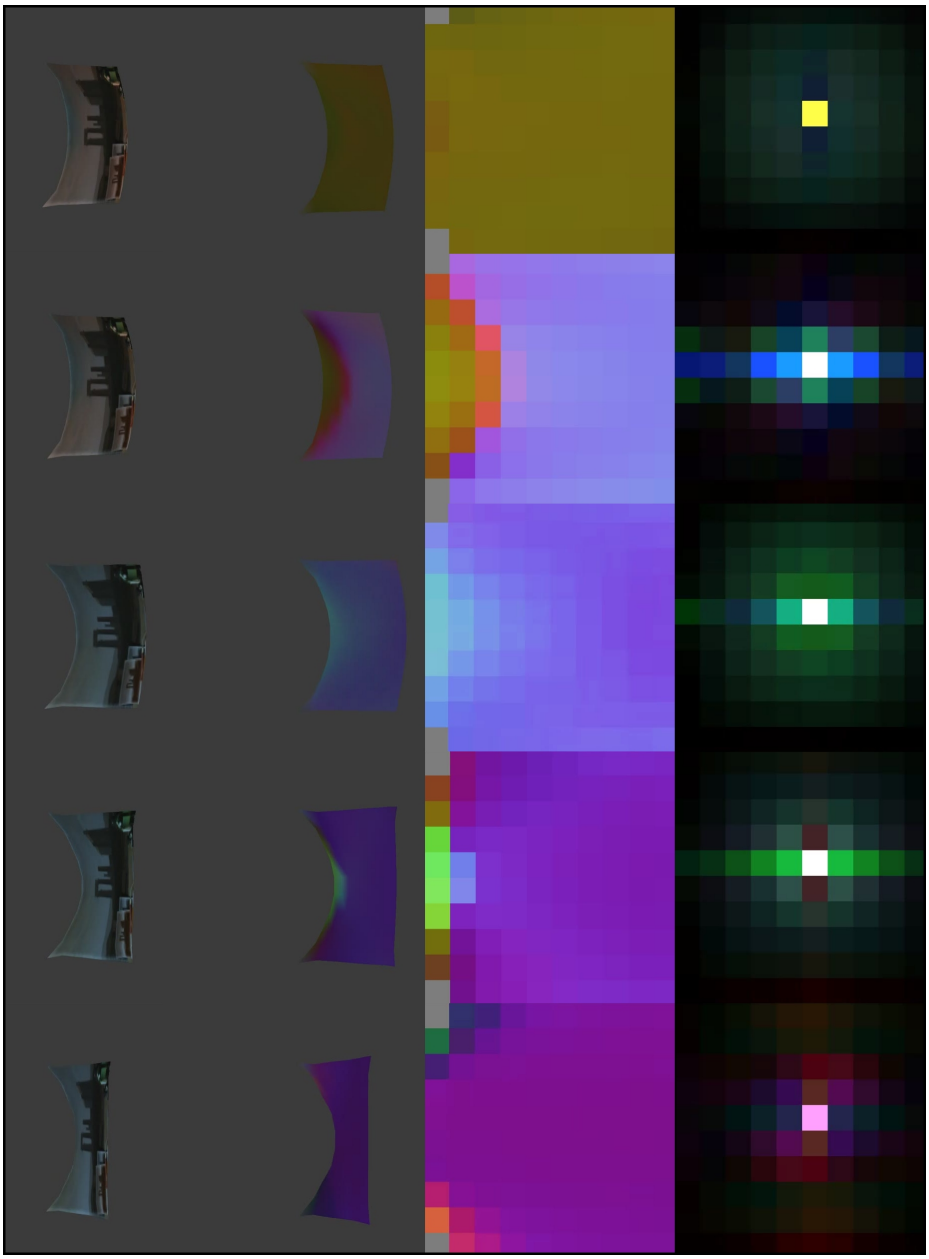

FIG. 4.7: *Première ligne en partant du haut et en tournant la page :*un drap texturé est soumis à la gravité en utilisant une méthode d'intégration numérique explicite avec filtrage, *deuxième ligne :* les couleurs représentent les vitesses, *troisième ligne :* qui sont affichées dans le même plan, *quatrième ligne :* le domaine fréquentiel obtenu par une transformée de Fourier. Les images coïncident temporellement de haut en bas.

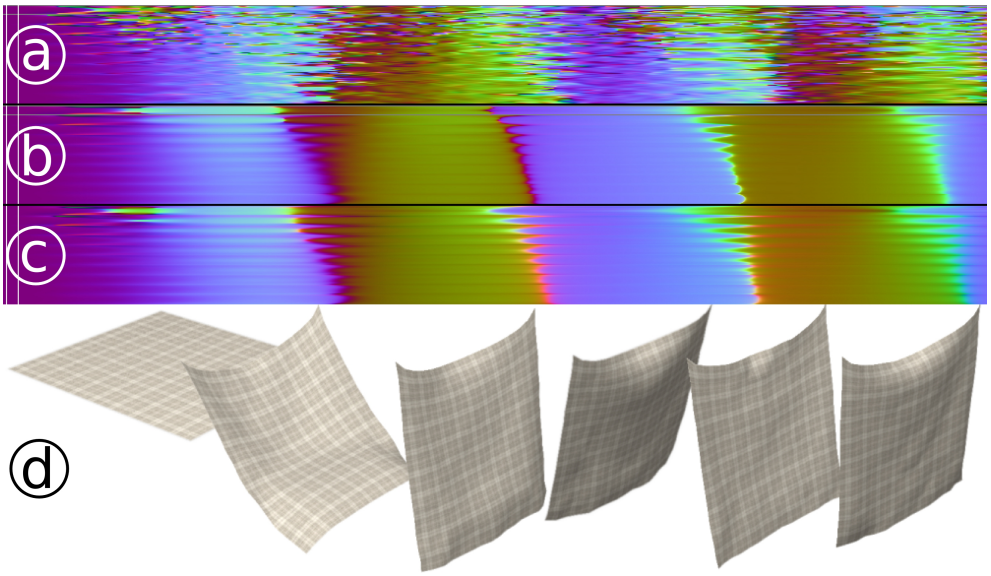

FIG. 4.8: L'évolution des vitesses calculées par : (a) FB Euler (b) *backward* Euler (c) FFB Euler (d) des captures d'écran illustrent aux instants respectifs le mouvement du tissu.

trant la méthode explicite FB Euler (figure 4.7).

4.4.3 Critère de stabilité

Puisque l'amortissement artificiel est partiellement responsable de la stabilité des méthodes implicites, l'application d'un filtrage similaire sur des schémas explicites doit augmenter leur stabilité. Dans cette section est détaillé un algorithme qui permet de quantifier l'augmentation de cette stabilité.

Le pas d'intégration des méthodes d'intégration numériques appliquées au modèle masses-ressorts est directement proportionnel à la valeur propre maximale des vitesses/forces. Pour les méthodes de *forward* Euler, de FB Euler et de Verlet, le pas maximal qui assure leur convergence peut être calculé comme suit :

$$\Delta t_{max} = \frac{2}{\sqrt{k_0}} \tag{4.18}$$

où k_0 est la valeur propre maximale calculée à partir de la matrice de rigidité (voir

les travaux de Michael Hauth [47] et de Mikio Shinya [90]). Une méthode simple et efficace pour déterminer k_0 est la méthode des puissances. Pour le besoin, j'ai modifié cette méthode en y intégrant le filtrage. Ce procédé est détaillé dans l'algorithme 2 et un graphique de résultat est montré dans la figure 4.9.

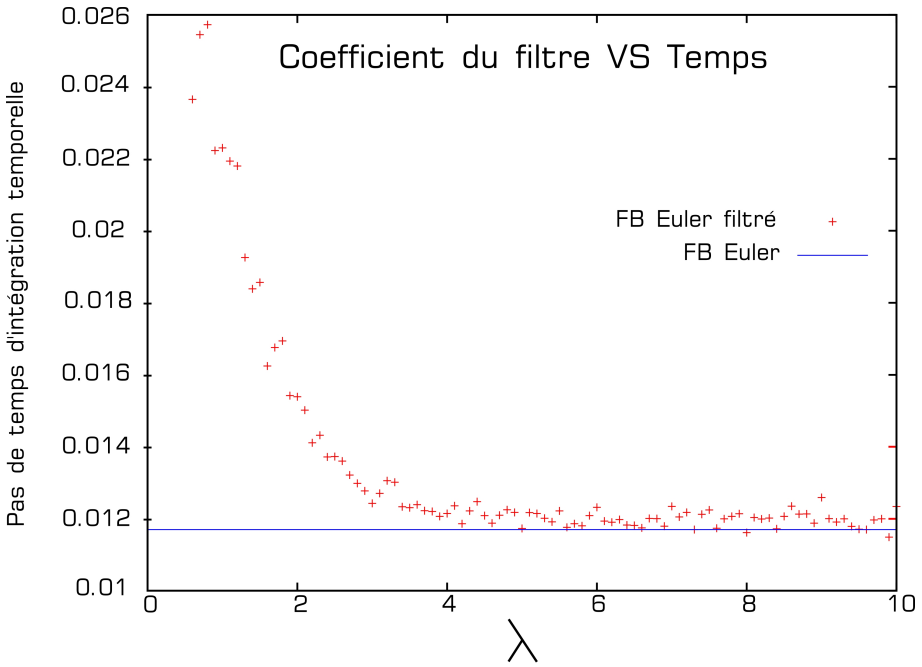

FIG. 4.9: La taille du pas d'intégration temporelle est déterminée par λ.

Identiquement à l'amortissement artificiel issu des formulations implicites, plus le pas d'intégration est grand, plus le filtrage doit être important. Lorsque $\lambda = 1$, le pas d'intégration est doublé. Les animations sont alors calculées quasiment deux fois plus rapidement. Dans ce cas k_0 est divisé par quatre (voir l'équation 4.18). Si les coefficients des ressorts et des masses ne changent pas au cours de l'animation, ce qui est en pratique souvent le cas, alors k_0 est du même ordre, ce qui implique un pas d'intégration temporelle maximal constant. Cet algorithme est donc exécuté une seule fois pour toute, en guise de précalcul. La complexité de l'algorithme 2 est de l'ordre de $\mathcal{O}(n)$, en utilisant la structure de données proposée par Pascal Volino et Nadia Magnenat-Thalmann [106], car peu d'itérations suffisent pour assurer la convergence de la méthode.

Algorithme 2 La méthode des puissances modifiée

Nécessite: vecteur pos, lambda

1: v vecteur (aléatoire)
2: **pour** $k = 1, 2, 3, \ldots$ jusqu'à la convergence **faire**
3: $\quad v \leftarrow A \cdot v$
4: $\quad \alpha \leftarrow max(v)$
5: $\quad v \leftarrow v/\alpha$
6: $\quad v \leftarrow filtrer(pos, v, \lambda)$
7: **fin pour**
8: **retourne** α

Un phénomène curieux se produit lorsque le pas d'intégration, en activant le filtrage, est égal au seuil de la stabilité, calculé par l'algorithme 2. Des motifs périodiques horizontaux et verticaux se dessinent sur le maillage, de plus un léger mouvement également périodique est visible. Une capture d'écran illustre ces propos en figure 4.10. Bien que je n'ai pas d'arguments scientifiques fondés pour expliquer ce comportement, je suppose qu'il est le résultat d'une propagation de forces légèrement erronées. Ces erreurs engendrent une phase d'oscillations exactement comme montré dans la figure 2.2. En tout cas, la présence de fréquences est indéniable.

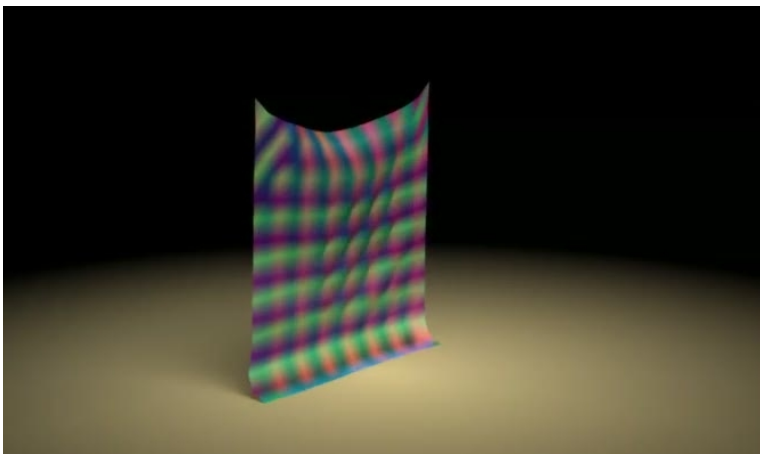

FIG. 4.10: Lorsque la limite de l'état de stabilité est atteinte, des curieux motifs périodiques se dessinent sur le maillage.

4.4.4 Résultats

Dans cette section figurent quelques discussions et des comparaisons de temps de calculs. Pour toutes les expérimentations présentées, la plateforme de test utilisée dispose d'un processeur cadencé à 2Ghz, de 512 Mo de mémoire vive et du système d'exploitation Linux.

4.4.4.1 Discussion sur le filtrage

Certaines des expérimentations ont supporté sans inconvénients une augmentation double du pas. Ainsi les temps de calculs d'animations sont réduits de presque 50%. D'autres expérimentations ne l'ont pas permise. La raison est que la propagation des forces n'est pas la seule source d'instabilité des méthodes explicites. L'explication suivante tente de clarifier le phénomène : une instabilité est issue d'un cumul d'erreur trop important. Ce cumul ne cesse de croître et se propage dans tout le système et des valeurs NaN (*Not a Number*) en résultent. Ceci est appelé une explosion numérique. le filtrage atténue les vitesses, et ce, constamment. Il peut corriger une vélocité erronée d'une particule en fonction des vitesses des particules voisines. C'est-à-dire que si un cumul d'erreur sur une particule isolée risque de rendre le système instable, grâce à une pondération des vitesses des particules voisines, cette vélocité peut être corrigée. Bien sûr tout dépend de l'ordre de l'erreur. En revanche si toutes les vitesses sont erronées, c'est-à-dire que le cumul d'erreur existe dans un ensemble de particules proches, le filtrage ne peut corriger les vitesses il ne peut pas empêcher une explosion numérique. C'est un scénario typique des système rigides. Tous les exemples expérimentés ont cependant autorisé une augmentation de 25% du pas. L'utilisation d'un tel pas avec une méthode explicite, sans filtrage, rend le système numériquement instable, c'est-à-dire qu'il explose (voir les captures d'écran dans la figure 4.11). Le système devient stable lorsque le filtrage nécessaire est appliqué.

Le nombre de voisins est constant. Il est usuellement beaucoup plus petit que le nombre de particules n, donc la complexité de l'algorithme de filtrage est en $\mathcal{O}(n)$. Avec des pas d'intégration 25% plus grands, les temps de calculs d'animations sont 20% plus court, ce qui implique que le filtrage a un coût de 5%.

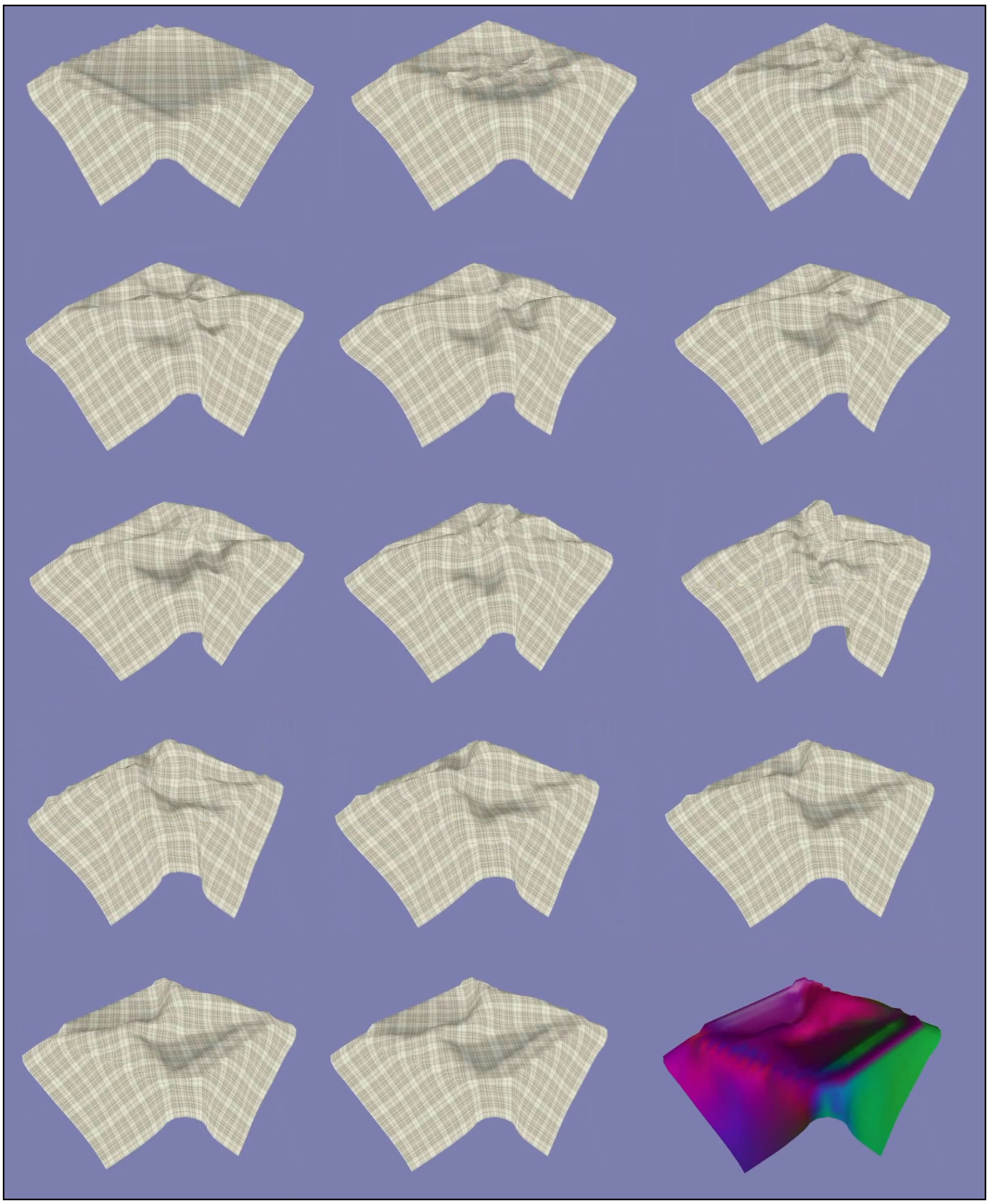

FIG. 4.11: Un drap, intégré avec le schéma *forward Euler*, recouvre une table. Le pas est volontairement augmenté pour qu'il dépasse la limite autorisée. Rapidement des masses ont un comportement aléatoire. Avant que le système "explose", le filtrage est activé (les deux dernières lignes) et par conséquent le système retourne dans une phase stable. La dernière image montre les vitesses des masses dans l'espace *(r,g,b)*.

4.4.4.2 Comparaisons des temps de calculs

Les méthodes numériques sont dépendantes du pas d'intégration temporelle (qui est linéairement dépendant de la valeur propre maximale k_0 des vecteurs vitesses/forces, dû à la rigidité des ressorts) et du nombre de particules n. Les temps de comparaisons présentés ci-après résultent d'expérimentations effectuées sur des schémas explicites, implicites et explicites filtrés en variant les paramètres k_0 et n.

Les méthodes expérimentées sont les suivantes : 3 explicites (FB Euler, RK4, et Verlet), une méthode implicite (*backward* Euler) et la méthode IMEX proposée par Bernhard Eberhardt *et al.* [22]. Pour des comparaisons justes, la structure de données utilisée pour les méthodes IMEX et implicite est celle proposée par Pascal Volino et Nadia Magnenat-Thalmann [106]. Pour ces deux modèles, le pas d'intégration temporelle est fixée à 0.01s. Pour les méthodes explicites FB Euler et Verlet, le pas d'intégration maximal est calculé en utilisant l'équation 4.18, et pour le schéma de RK4 par l'équation suivante : $\sqrt{8.75}/\sqrt{k_0}$ (voir les travaux de Mikio Shinya [90]). Cependant, pour ces modèles le pas d'intégration est toujours inférieur ou égal à 0.01s. Les collisions sont désactivées pour des comparaisons justes car elles sont en général traitées par des procédés différents. La figure 4.12 montre des comparaisons de temps de calculs en fonction du nombre de particule (avec k_0 fixé à 10^6).

Les résultats révèlent une accélération des temps de calculs d'environ 20% quand les schémas explicites sont filtrés avec une valeur de $\lambda = 1$ et avec un pas d'intégration 25% plus grand. La méthode de Verlet n'est pas montrée pour des raisons de clarté mais est sujette à des améliorations équivalentes.

Il est important de noter que l'algorithme du gradient conjugué, qui occupe majoritairement le calcul des méthodes implicites, dépend grandement de son critère de convergence, c'est-à-dire le seuil d'erreur ϵ et le nombre d'itération maximum. Dans toutes les expérimentations, j'ai fixé un seuil relativement grand, c'est-à-dire $\epsilon = 0.01$ (stable cependant). Mais lorsque ϵ est réduit, la résolution est alors plus longue et dans ce cas la méthode de FFB Euler est plus rapide. La figure 4.13 montre des comparaisons de temps de calculs lorsque k_0 varie. Les méthodes implicites et IMEX deviennent incontestablement plus efficaces lorsque $k_0 > 10^6$. Il est alors dans un tel

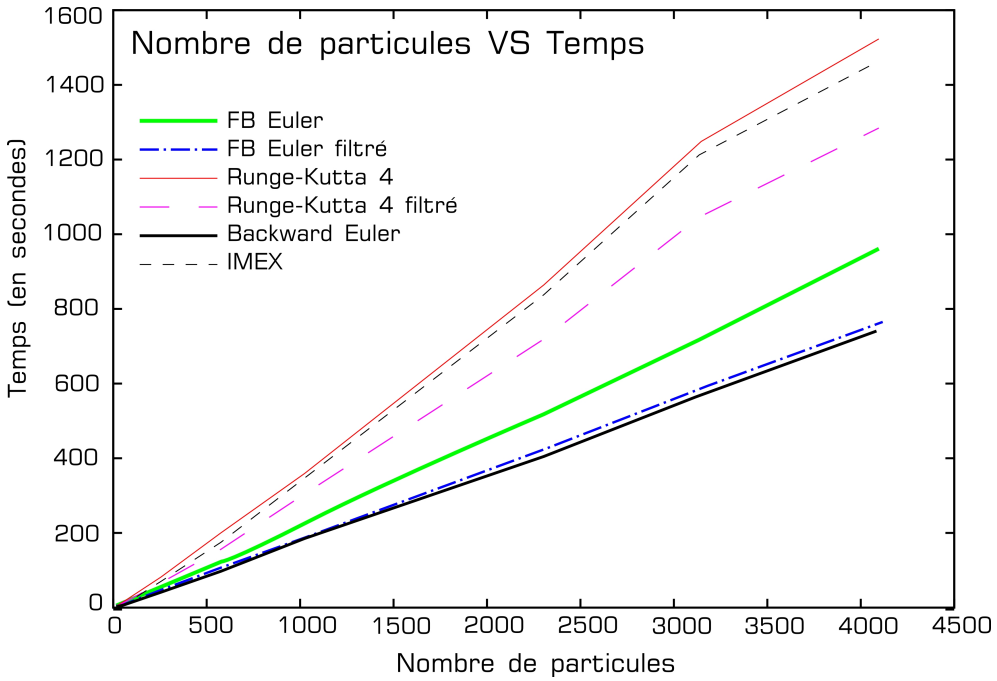

FIG. 4.12: Comparaisons entre les temps de calculs et le nombre de particules, pour $k_0 = 10^6$.

cas plus judicieux d'éviter l'utilisation des méthodes explicites.

L'application du filtrage sur le maillage de poisson a révélé des résultats du même ordre, c'est-à-dire une augmentation possible du pas d'intégration de 25%. Les simulations calculées par les trois méthodes, explicite, implicite et explicite filtrée sont visuellement similaires. Il existe cependant une légère différence dans la propulsion uniquement quand le pas est inférieur à $0.01s$. Une capture d'écran de l'animation résultante est montrée dans la figure 4.14.

4.4.5 Conclusions et travaux futurs

Les méthodes explicites disposent de propriétés intéressantes. Elles sont simples à mettre en oeuvre et sont relativement précises. Malheureusement elles sont inutilisables dans beaucoup de cas en pratique car elles nécessitent des pas d'intégration temporelle petits. Pour des systèmes rigides par exemple, les temps de calculs sont alors importants. Le filtrage des forces induit par la formulation implicite permet

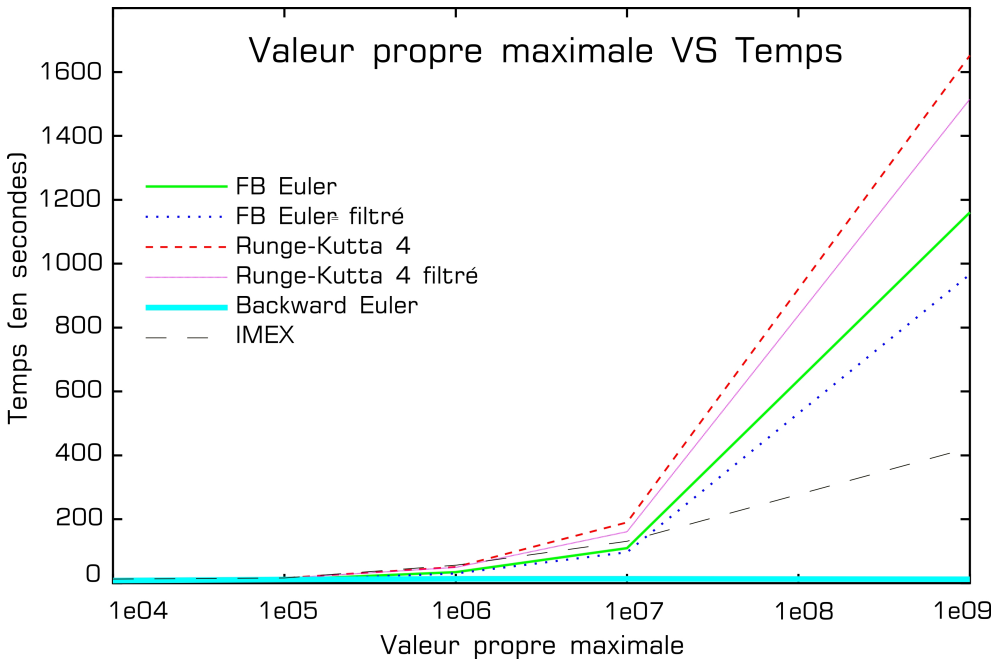

FIG. 4.13: Les méthodes explicites ont des temps de calculs compétitifs jusqu'à ce que $k_0 > 10^6$.

d'éviter certains problèmes de stabilité. En tirant profit de ce procédé, j'ai appliqué un filtrage similaire sur des schémas explicites. Les résultats et les analyses montrent que les pas d'intégration temporelle peuvent être agrandis de 25% jusqu'à 100%, et par conséquent les temps de calculs sont écourtés de 20% à presque 50%. Avec certains exemples, comme par exemple avec $k_0 \leq 10^6$, les temps de calculs obtenus par la méthode de filtrage et par une formulation implicite sont compétitifs. Cependant l'ajout de l'amortissement engendré par le filtrage est un inconvénient car les animations sont, en fonction du coefficient λ, plus adoucies. Les méthodes explicites sont inutilisables pour des exemples avec des coefficients de rigidité plus importants, même en intégrant le filtrage proposé. Pour résoudre ce problème, il est possible d'appliquer d'autres procédés issus des formulations implicites tels que la *linéarisation* ou le développement en premier ordre des Jacobiennes des forces. En optant pour un filtre plus puissant tel que celui de Kalman, récemment mis en oeuvre pour des travaux connexes de Marcus Grote and Andrew Majda [41], la stabilité des méthodes

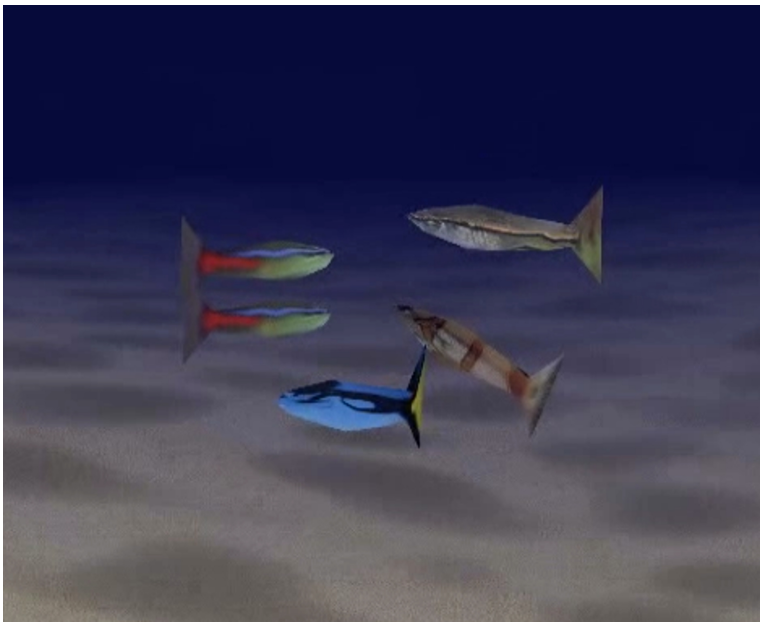

FIG. 4.14: L'intégration des poissons est une grande question sociale...

pourrait être encore agrandie. Ceci est abordé dans la section 4.5. Il serait également intéressant de pouvoir quantifier les fréquences indésirables afin de ne supprimer que celles-ci, en utilisant par exemple une analyse de Von Neumann.

4.5 Stabilisation conditionelle par un filtrage de Kalman

Le filtrage présenté ci-après est la suite directe des travaux présentés dans la section précédente 4.4. La résolution du système linéaire engendré par la formulation implicite génère un filtrage basse-fréquence. Ce filtrage exponentiel est partiellement responsable de la stabilité des méthodes implicites. L'application d'un filtrage similaire sur des méthodes explicites révèle une augmentation de leur stabilité. Cependant, ce filtrage est basique. Un filtrage plus évolué et plus puissant, tel que celui de Kalman présenté dans cette section, devrait permettre d'accroître d'avantage la stabilité des méthodes explicites. Malheureusement, ce modèle n'a pu être mené à terme et aucun

résultats ne se sont avérés positifs. Ni le modèle ni son application sur les méthodes explicites ne sont cependant remis en cause. Simplement, à cause du peu de temps passé, les travaux n'ont pas abouti.

Dans cette section figure des discussions sur les directions prises, sur l'élaboration et l'application de ce filtre dans le cas des méthodes d'intégrations explicites et des systèmes masses-ressorts.

Après une introduction en sous-section 4.5.1, les modèles de perturbations et de capteurs nécessaires pour le bon fonctionnement du filtre sont détaillés respectivement en sous-sections 4.5.2 et 4.5.3. Les résultats obtenus et les conclusions sont discutés en sous-section 4.5.4.

4.5.1 Introduction

Le filtre de Kalman est un estimateur récursif qui prédit l'état d'un système dynamique à partir d'un ensemble de mesures perturbées. Notamment il permet de fournir des informations précises concernant la position et la vitesse d'un objet à partir d'une séquence d'observations légèrement erronées. Cela signifie que la nouvelle position est estimée à partir de l'état précédent du système et des mesures actuelles de **capteurs**. Donc ce filtre nécessite :

- une fonction qui prédit l'état du système, dans le cas présent l'accélération et/ou la vitesse et/ou la position, mais avec une incertitude ou variable aléatoire ;
- un capteur qui analyse une zone d'**observation** prédéfinie et qui corrige la **prédiction** de l'état du système.

Ce filtre est très utilisé dans le domaine du traitement du signal, mais aussi en météorologie, en navigation ou en finance. Ci-dessous figure une application du filtre de Kalman : soit un véhicule qui se déplace à une certaine vitesse. Connaissant sa position de départ, il est possible de prédire les positions futures. Mais une fonction de prédiction ne peut prendre en compte tous les petits phénomènes qui peuvent altérer la vitesse et la position du véhicule. De plus, la résolution numérique de la fonction engendre également des erreurs. Des mesures de positionnements relevées par le biais de capteurs permettent de corriger les résultats plus ou moins erronés de la fonction

de prédiction et ce, par des procédés statistiques.

Les méthodes d'intégration explicites sont instables si la contrainte sur le pas d'intégration temporelle n'est pas respectée. La nouvelle position/vitesse **prédite** est alors à une certaine distance de la positon/vitesse correcte. Je définis cette distance comme un bruit, le filtre de Kalman devrait pouvoir corriger cette prédiction via un modèle de capteur adéquat.

L'état du filtre est représenté par 2 variables :
$\hat{\mathbf{x}}_{t|t}$, l'estimation de l'état à l'instant t ;
$\mathbf{P}_{t|t}$, La matrice de covariance de l'erreur (une mesure de la précision de l'état estimé).

Le filtre de Kalman a deux phases distinctes, dénommée *prédiction* et *mise à jour*. La phase de prédiction utilise l'état estimé de l'instant précédent pour produire une estimation de l'état courant. Dans l'étape de mise à jour, les observations de l'instant courant sont utilisées pour corriger l'état prédit dans le but d'obtenir une estimation plus précise.

La phase de prédiction est définie comme suit :
$\hat{\mathbf{x}}_{t|t-1} = \mathbf{F}_t\hat{\mathbf{x}}_{t-1|t-1} + \mathbf{B}_t\mathbf{u}_t$ (état prédit)
$\mathbf{P}_{t|t-1} = \mathbf{F}_t\mathbf{P}_{t-1|t-1}\mathbf{F}_t^T + \mathbf{Q}_t$ (estimation prédite de la covariance)

La phase de mise à jour est définie comme suit :
$\tilde{\mathbf{y}}_t = \mathbf{z}_t - \mathbf{H}_t\hat{\mathbf{x}}_{t|t-1}$ → innovation
$\mathbf{S}_t = \mathbf{H}_t\mathbf{P}_{t|t-1}\mathbf{H}_t^T + \mathbf{R}_t$ → covariance de l'innovation
$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}_t^T\mathbf{S}_t^{-1}$ → gain de Kalman "optimal"
$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t\tilde{\mathbf{y}}_t$ → état mis à jour
$\mathbf{P}_{t|t} = (I - \mathbf{K}_t\mathbf{H}_t)\mathbf{P}_{t|t-1}$ → covariance de la mise à jour

Plus d'informations sur ce filtre figurent dans le document de Greg Welch et Gary Bishop [112].

L'application du filtre de Kalman dans le cas d'intégration explicites de systèmes masses-ressorts nécessite :

– la connaissance de la fonction de prédiction et de la variable d'erreur aléatoire ;

– la fenêtre d'observation, c'est-à-dire le modèle de capteurs mis en oeuvre.

La fonction de prédiction est définie par le système d'EDO auquel est ajoutée une variable d'erreur aléatoire (voir la section 4.5.2). De plus, le filtre requiert la définition de la fenêtre d'observation (sous-section 4.5.3) c'est-à-dire le bruit associé au capteur[4].

4.5.2 Variable d'erreur aléatoire

Le premier paramètre du filtre de Kalman est l'erreur de la variable à estimer. Dans le cas de systèmes masses-ressorts, ces variables sont la position, la vitesse et éventuellement l'accélération. La position est facilement calculée à partir d'une vélocité valide. Toujours dans le but d'appliquer le filtrage en post-traitement, je ne me suis intéressé qu'à l'estimation de la vitesse.

Soit une intégration temporelle par une méthode explicite, avec un pas $\Delta t > \Delta t_{max}$, où Δt_{max} est le pas maximal qui assure une intégration stable. À partir d'un *bruit* (variables aléatoire), des analyses statistiques devraient permettre de préserver la convergence de la méthode explicite.

Je définis l'échelle du bruit comme suit :

$$d = \frac{\Delta t}{\Delta t_{max}} \tag{4.19}$$

La variable d est la mesure d'erreur. Lorsque $d < 1$, aucune modification n'est nécessaire puisque la méthode d'intégration est dans un état stable. En revanche, lorsque $d \geq 1$, je suppose que le filtrage pourrait corriger les erreurs d'intégrations numériques.

Pour chaque dimension du vecteur de vitesse, la variable aléatoire est définie comme suit (avec $d \geq 1$) :

$$v^t_{alea} = rand(1 - d, d - 1) \cdot v^t \tag{4.20}$$

Ainsi la vitesse qui est supposée être correcte est située dans un espace $[1-d \ldots d-$

[4] un exemple d'application sur des systèmes dynamiques est détaillé dans le document à l'adresse suivante http ://en.wikipedia.org/wiki/Kalman_filter#Example

1] défini autour de la vitesse qui est supposée être erronée.

4.5.3 Fenêtre d'observation

La distance entre l'état du système souhaité et l'état du système prédit est délimitée dans une zone dans laquelle des analyses statistiques vont pouvoir être mises en place.

Dans les systèmes électroniques et mécaniques, les capteurs ne sont pas parfait pour diverses raisons. Par exemple un radar satellite permet de situer un véhicule à 3 mètres près. C'est exactement le même procédé dans le cas de l'intégration numérique de systèmes masses-ressorts. Cette zone de flou, appelée fenêtre d'observation, est la zone visible par le capteur.

La mesure d (voir l'équation 4.19) va définir la taille de la fenêtre d'observation :

- si $d-1 \simeq 0$, alors l'erreur est petite, par conséquent la fenêtre d'observation doit également être petite. Un léger filtrage devrait corriger la mauvaise prédiction ;
- si $d-1 >> 0$, alors l'erreur est importante, par conséquent la fenêtre d'observation doit également être importante, impliquant une correction considérable de la part du filtre.

Pour chaque particule du système masses-ressorts, un demi-ovoïde centré sur l'ancienne position représente la zone d'observation. La taille de l'ovoïde est définie ainsi :

$$ovoide_i = d \cdot \mathbf{v_i^t} \tag{4.21}$$

où $\mathbf{v_i^t}$ est la vitesse (en 3D) de la i-ème particule à l'instant t. La figure 4.15 schématise cette zone d'observation, uniquement en deux dimensions pour des raisons de clarté.

4.5.4 Résultats et conclusions

Ce filtrage a été appliqué sur différents modèles masses-ressorts, des tissus, des cubes et des poissons. Malheureusement, aucun résultat encourageant n'a été obtenu. L'introduction d'une variable aléatoire dans le vecteur vitesse engendre des pertur-

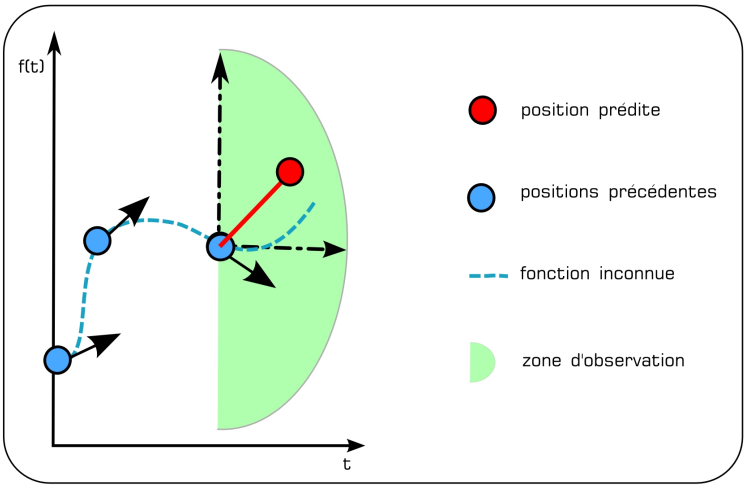

FIG. 4.15: Le bruit du capteur définie dans la zone d'observation (verte) sert à corriger une prédiction erronée (rouge).

bations sur les sommets du maillage. Ces perturbations sont visibles et gênantes. De plus, elles ont tendance à rendre le système instable. Peu de temps a été consacré à l'élaboration de cette méthode. La variable aléatoire, ainsi que la fenêtre d'observation sont peut être mal définies. Par ailleurs, ce filtre ne fonctionne qu'avec des systèmes linéaires. Le filtre de Kalman *étendu* gère les systèmes non-linéaires et est par conséquent plus adapté à notre application. Il est important de noter que ce dernier nécessite la résolution numérique d'un système linéaire. Le filtrage en post-traitement vise à stabiliser les méthodes explicites à moindre coût. Si ce filtrage dispose d'une complexité équivalente à celle d'une formulation implicite, un tel post-traitement n'est alors plus très judicieux.

4.6 Conclusion

Ce chapitre aborde la stabilisation des méthodes d'intégration temporelle explicites. Trois méthodes sont présentées : la méthode basée sur un θ-schéma et la méthode visant la stabilisation par filtrage de Kalman n'ont pas révélé pas de résultats concluants, la stabilisation par filtrage exponentiel a fait l'objet en revanche d'une

publication.

La méthode basée sur un θ-schéma consiste à décomposer les forces en termes linéaire et non-linéaire, chacun résolus respectivement implicitement et explicitement. En fonction d'un certain critère, θ, le terme explicite ou implicite est favorisé pour garantir une intégration stable tout en conservant des calculs rapides. Favoriser ponctuellement ces forces décomposées n'est pas judicieux car elles ne sont pas indépendantes. Ainsi faisant, la répulsion ou bien l'attraction domine l'autre respectivement et l'objet est agrandi ou rétréci. Cette méthode ne fonctionne donc que pour une seule valeur de $\theta = 0.5$.

La formulation implicite donne lieu à un système linéaire. La résolution numérique de ce système engendre un filtrage, qui en fait est une propagation des forces. Cette propagation est partiellement responsable de la stabilité inconditionnelle de l'approche implicite. Je propose donc d'appliquer ce procédé de propagation de forces sur les méthodes explicites pour améliorer leur stabilité. Les résultats montrent que le pas d'intégration temporelle peut être agrandie de 25%, réduisant ainsi les temps de calculs de l'ordre de 20%.

Suite à ces résultats encourageant, j'ai décidé d'utiliser un filtrage plus puissant, comme par exemple celui de Kalman qui est utilisé dans des applications connexes. Les résultats ne sont pour l'instant pas concluants car l'introduction de la variable aléatoire engendre des oscillations des sommets du maillage. Malheureusement et fautes du peu de temps consacré, ce travail n'a pu être mené à terme.

Le travail sur le filtrage consiste à considérer un signal non modifiable en entrée, sur lequel un algorithme tente de corriger les éventuelles erreurs. Dans un souci de temps de calculs, le filtrage doit être le plus rapide possible.

Bien que montrant des résultats intéressant, le signal en entrée, c'est-à-dire les données générées par les méthodes d'intégrations, **est** modifiable. Je pense qu'il est cependant plus judicieux de générer un signal, sans erreurs, plutôt que de corriger des erreurs à posteriori. Je suggère donc d'utiliser des méthodes d'intégration inconditionnellement stables, c'est-à-dire des méthodes implicites, avec des systèmes rigides.

Il serait cependant intéressant de constater qu'un filtrage pourrait stabiliser sans conditions les méthodes d'intégration explicites. L'intérêt est cependant limité si le

coût du filtrage est onéreux en temps de calculs.

Chapitre Cinq

Structure accélératrice pour la détermination du voisinage

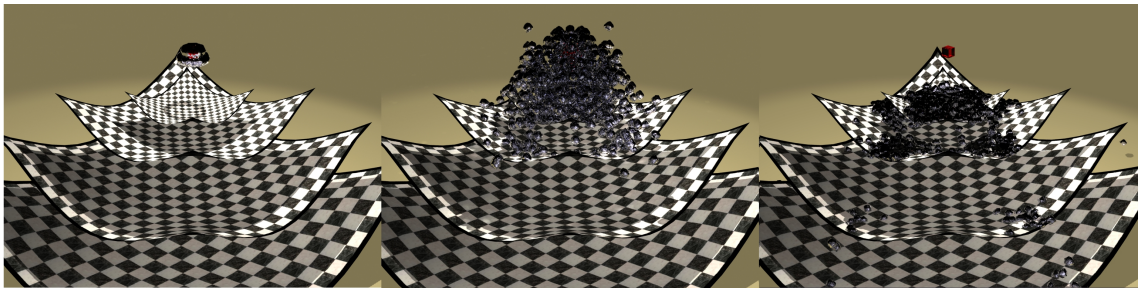

Fig. 5.1: La détermination du proche voisinage, la détection de collisions et le processus d'extraction de surface sont optimisés à l'aide du modèle de subdivision spatiale proposé.

Les structures Lagrangiennes sans maillage ont l'avantage de pouvoir traiter les changements de topologie sans difficultés. De plus, leur formulation Lagrangienne offre les avantages de la parallélisation, de la définition des bords et de la simplification de certains calculs. Ces structures profitent donc des avantages Eulériens et Lagrangiens. Ce profit est au dépend de la détermination du voisinage, processus coûteux, qui est indispensable pour calculer les changements de positions et de surface. Plusieurs modèles ont vu le jour dont la référence est une table de hachage unique qui propose une récupération du voisinage efficace via une subdivision dynamique de l'espace. La complexité d'un algorithme naïf qui détermine le voisinage est en $\Theta(n^2)$, où n est le nombre d'éléments. Puisque les informations de voisinage sont constamment sollicitées, une réduction de cette complexité s'impose. De plus, des calculs complexes sont effectués ultérieurement à partir de ces voisins. Pour éviter d'effectuer des calculs

complexes inutiles, l'ensemble des proches voisins doit être nécessaire et suffisant. La détermination efficace et en temps constant de cet ensemble suffisant et nécessaire avec des structures dynamiques qui se déplacent dans un espace non borné n'est finalement pas si simple.

La détection de collisions est un procédé qui nécessite également, pour une optimisation très recommandée, la connaissance du proche voisinage. Généralement un autre modèle de subdivision dynamique de l'espace est mis en oeuvre, comme le bucket-tree ou encore le kd-tree.

Par ailleurs, pour les structures Lagrangiennes sans maillage, l'extraction ou le déplacement de la surface requiert encore une fois la connaissance du proche voisinage. Généralement, le modèle de table de hachage préétablie est réutilisé.

L'animation et l'interaction d'une structure Lagrangienne sans maillage, telle que les *Smoothed Particles Hydrodynamics* (SPH) (voir la section 9.1), et d'une structure Lagrangienne avec maillage, telle que les systèmes masses-ressorts, nécessitent un modèle de subdivision spatiale pour le calcul de la dynamique du fluide et un autre modèle de subdivision spatiale pour la détection de collisions et éventuellement un troisième pour l'extraction de la surface. Je me suis intéressé à l'élaboration d'une structure de subdivision spatiale unifiée pour déterminer efficacement le voisinage d'un point, que ce soit un triangle ou bien une particule. Je propose un modèle basé sur les tables de hachage en ajoutant la notion de hiérarchie. Cette hiérarchie, qui en fonction de la taille des objets, a pour but d'optimiser l'insertion et la récupération d'éléments de tailles variés. Ce modèle doit évidemment être dynamique, c'est-à-dire qu'il doit pouvoir prendre en compte le déplacement d'un ou de tous les éléments à moindre coût et ce, à chaque pas d'intégration temporelle. C'est pour cette raison que les structures de type arbre octal statique sont exclues.

Ce chapitre détaille la table de hachage hiérarchique, ou modèle unifié de subdivision spatiale dynamique, qui est utilisé pour le calcul du voisinage des particules de fluides et pour la phase de détection de collision dite d'élagage approximatif, ou de *broad phase*. En comparaison avec le modèle de référence utilisé pour le calcul du proche voisinage avec un fluide Lagrangien (hachage simple), la structure proposée ci-dessous non seulement ne nécessite aucune paramétrisation manuelle, mais en plus

est en moyenne trois fois plus rapide pour des animations équivalentes. Cette structure de subdivision spatiale est entre autres utilisée pour détecter le voisinage des particules lors de la construction de la surface du fluide (voir le chapitre 7).

La section 5.1 introduit la problématique et les principaux travaux du domaine sont survolés dans la section 5.2. La structure accélératrice proposée est détaillée en section 5.3, qui est divisée en trois points :

1. la section 5.3.1 aborde la classification des éléments ;
2. la section 5.3.2 propose de réduire les conflits de hachage à l'aide de la connaissance de la hiérarchie ;
3. la section 5.3.3 explique l'insertion des triangles et des particules.

5.1 Introduction

Le modèle SPH considère un fluide comme un ensemble de particules. Le calcul de la dynamique de ce fluide nécessite la connaissance du voisinage de chaque particule. Les tables de hachage spatiales et dynamiques, comme celle présentées par Brian Mirtich [72] et Matthias Teschner *et al.* [98] sont les plus utilisées pour répondre à ce besoin, car elles sont efficaces en terme d'insertion et de récupération de données. Dans le cadre des SPH, la fonction de hachage convertit les coordonnées spatiales d'une particules en une clé. Cette clé est un indice d'un tableau unidimensionnel, ou table, dans laquelle est enregistrée la particule. L'objectif est d'établir une table de hachage qui permet de stocker les particules voisines dans la même cellule et d'éviter à tout prix d'insérer dans cette cellule des éléments éloignés. De plus, cette fonction ne doit pas borner l'espace pour que le fluide puisse ainsi se déplacer librement, mais doit prendre en compte les limitations en ressources mémoire. La fonction de hachage divisent implicitement l'espace uniformément. Ce modèle peut entre autres servir de structure accélératrice pour détecter une interaction avec l'environnement. Cependant, à cause de la subdivision uniforme de l'espace, cette structure doit être judicieusement utilisée. En prenant pour exemple une particule d'un fluide qui interagit avec ses particules voisines dans un rayon de longueur r, la taille optimale des cellules de la division

spatiale uniforme est par conséquent de r (voir les travaux de Müller *et al.* [75, 76] pour plus d'informations). Ce fluide se déversant sur un plan de dimension $b \gg r$, le processus d'insertion de ce plan, situé dans un espace tridimensionnel, dans la table de hachage a une complexité proportionnelle à $(b/r)^3$. Il est clair qu'en fonction de la configuration de la scène tridimensionnelle, l'insertion peut devenir le processus de la simulation le plus onéreux en terme de temps de calculs. Pour cette raison, deux structures de subdivisions spatiales sont couramment utilisées, une pour détecter le proche voisinage au sein du fluide, l'autre pour accélérer la détection de collisions.

La structure proposée détermine grossièrement les collisions possibles entre plusieurs objets. De cette phase dite d'élagage approximatif, ou de *broad phase*, résulte un trop grand nombre de candidats, dont certains ne sont pas susceptibles d'interpénétrer un autre objet (ou eux même). Puisque le but est d'appliquer un algorithme de réaction de collisions, souvent coûteux en temps de calculs, sur uniquement les candidats potentiels, une deuxième phase, dite d'élagage précis ou de *narrow phase*, est mis en oeuvre. Elle est abordée dans le chapitre 6. Cette table de hachage hiérarchique peut également optimiser le processus d'extraction de la surface, qui est détaillé dans le chapitre 7.

5.2 État de l'art

Cet état de l'art s'intéresse précisément aux modèles d'optimisations utilisés pour déterminer les proches voisins dans les simulations de dynamique des fluides et d'interactions fluide-structure.

Le modèle le plus simpliste est la grille uniforme. Cette structure est bien adaptée aux simulations dites Eulériennes. Nick Foster et Dimitris Metaxas [34] ont été les premiers dans le domaine de la synthèse d'images à résoudre les équations de Navier-Stokes dans un espace tridimensionnel. Pour ce faire ils ont utilisé simplement une grille uniforme. Il en est de même pour Jos Stam [92], Nick Foster et Ron Fedkiw [33] et autres extensions de ces modèles qui se sont principalement intéressés respectivement à la stabilité de la simulation et au rendu de la surface. Pour accélérer les calculs de la dynamique des fluides, qui dans une grille ont une complexité de l'ordre de $\Theta(x^3)$

où x est la longueur moyenne d'une arête du volume de l'espace de simulation, Frank Losasso *et al.* [62] ont mis en oeuvre une subdivision de l'espace à l'aide d'un arbre octal (octree).

Modéliser les interactions fluide-structure consiste à projeter l'objet dans la grille. L'interaction est reproduite en ajustant, en fonction de différents attributs, les paramètres de pression et de vitesse dans les cellules de la grille occupées par un objet (voir les travaux de Nick Foster et Ron Fedkiw [33] et de Mark Carlson *et al.* [17]). Suivant ce principe les objets doivent être de dimension plus importante que la dimension d'une cellule.

À chaque itération, Bryan Feldman *et al.* [30] remaillent le domaine de simulation en utilisant un modèle de subdivision à base de tétraèdres. Le voisinage est ensuite directement obtenu grâce au maillage.

Du côté des méthodes dites Lagrangiennes, les grilles uniformes sont mises en oeuvre principalement pour des simulations fonctionnant entièrement sur les cartes graphiques, telles celles de Takahiro Harada *et al.* [44]. Sinon, la structure la plus employée pour la recherche des proches voisins avec les méthodes Lagrangiennes est la table de hachage. Brian Mirtich [72] a utilisé une table de hachage hiérarchique pour la détection de collision, mis en oeuvre dans le logiciel ICOLLIDE. Matthias Teschner *et al.* [98] ont optimisé le fonctionnement des tables de hachage afin que leurs performances soient optimales pour des applications graphiques. Ces travaux ont été appliqué dans le cadre de simulation de fluide par Matthias Müller *et al.* [75, 76], Kevin Steele *et al.* [94] et Simon Clavet *et al.* [20][1].

Un autre façon de diviser l'espace pour restreindre au maximum le voisinage est d'utiliser des kd-trees. Cette technique a été utilisé par Neeharika Adabala et Swami Manohar [1] pour simuler du feu. Les auteurs l'utilisent entre autres pour accélérer les calculs de visualisation. La construction d'un kd-tree est assez coûteuse en terme de temps de calcul, mais par la suite la récupération d'éléments est très efficace. Richard Keiser [55] affirme que les tables de hachage sont plus performantes pour

[1] il n'est pas exclu d'utiliser cette structure accélératrice pour optimiser les simulations Eulériennes, comme démontré par David Cline *et al.* à l'adresse suivante : http ://poseidon.cs.byu.edu/∼cline/fluidFlowForTheRestOfUs.pdf

un nombre de particules restreint, comme par exemple pour le nombre requis pour des applications fonctionnant en temps réel, mais lorsque le nombre de particules est très important, comme par exemple pour les besoins cinématographiques, les kd-trees sont plus performants, car le temps de construction de l'arbre est alors négligeable comparé aux autres calculs. Les kd-trees ont été mis en oeuvre dans les travaux de Bart Adams *et al.* [3].

5.3 Détermination du voisinage par hachage hiérarchique

Cette section propose un modèle d'optimisation à mettre au service du calcul des proches voisins. La structure de données utilisée est composée de plusieurs tables de hachage, chacune correspondant à une division de l'espace prédéfinie appelée classe (voir la sous-section 5.3.1). Il existe une relation mère-filles entre les différentes cellules des tables, d'où le terme hiérarchique (voir la section 5.3.2).

Chaque table de hachage est décrite selon le modèle présenté dans les travaux de Matthias Teschner *et al.* [98]. L'extension hiérarchique proposée dans ce chapitre est inspirée des recherches de Brian Mirtich [72].

Une cellule est un élément de la table. Un voxel[2]d'une position et d'une taille prédéfinie dans l'espace est injectivement associé à cette cellule. Je définis la dimension des classes, donc des voxels comme une puissance de 2. Dans un espace tridimensionnel, un voxel est alors divisé en 8 sous-voxels associés à une classe de dimension directement inférieure. Un élément contenu dans un voxel associé à la $k-2$ classe (fille) est également contenu dans un voxel associé à la $k-1$ classe (mère).

[2]un voxel est un *volume élément*, qui est l'extension 3D de *picture element*, ou pixel. Dans ce chapitre le terme voxel fait référence à un volume cubique dans l'espace à ne pas confondre avec une cellule qui fait référence à une case d'une table.

Définition 5.1

$\forall t \in T, \exists v \in V$
- $t \in v^{k-2} \longrightarrow t \in v^{k-1}$
- $dim(t) \leq dim\left(C^{k-2}\right) \longrightarrow dim(t) < dim\left(C^{k-1}\right)$

où T est l'ensemble des triangles, v^k est un voxel associé à la classe d'ordre k, V est l'ensemble des voxels, C^k est la classe d'ordre k et dim est une fonction qui retourne la taille d'un élément (voir l'équation 5.1). Ce schéma peut être considéré comme une hiérarchie de type mère-filles, exactement comme pour une structure de type arbre octal. Mais contrairement à ces arbres, il n'est pas nécessaire de conserver toute la hiérarchie : uniquement les cellules contenant un élément sont considérées, ce qui rend possible la définition d'une grand-mère et d'une petite-fille sans avoir besoin de définir la mère et les soeurs. Lorsqu'il y a de grandes différences de dimensions entre les classes, par exemple $2^{-2}, 2^4, 2^{15}$, seulement 3 classes sont à considérer et non pas 18, ce qui permet d'économiser les ressources en mémoire.

5.3.1 Classification des cellules pour les triangles

Les scènes tridimensionnelles sont généralement constituées de millions de triangles de taille variable. L'insertion d'un triangle dans une table de hachage nécessite la projection de ce triangle dans la grille de voxels représentée implicitement par la table. Pour que le coût de cette insertion soit minimisé, alors la projection doit chevaucher le minimum de voxels :

Définition 5.2

$\forall t \in T, \exists c \in C, \exists v \in V | t \in v^k \longrightarrow dim(t) < dim(c^k), dim(c^{k-1}) \leq dim(t)$

La fonction *dim* prend en paramètre un élément et retourne la taille correspondant à la plus proche et supérieure puissance de 2. Elle est définie comme suit :

$$dim(t) = \left\lceil \frac{ln(max(t))}{ln(2)} \right\rceil \tag{5.1}$$

où $max(t)$ est une fonction qui retourne la plus grande distance géométrique d'un triangle, c'est-à-dire la plus grande arête de la boite englobante alignée aux axes (AABB). Ce calcul est effectué pour chaque élément de la scène au début de la simulation dans une phase de pré-traitement. Suite à ce calcul résulte une liste contenant les dimensions des éléments. En retirant les doublons, cette liste est de taille k. Il y a alors k classes nécessaires pour ranger les éléments de la scène tridimensionnelle, donc k tables de hachage sont créés. Chaque table de hachage peut contenir m éléments, où m est un nombre premier proche et supérieur au nombre d'éléments n, m est la taille optimale (voir les travaux de Matthias Teschner *et al.* [98]).

Cette structure est conçue pour gérer les déformations d'objets, c'est-à-dire pour traiter les changements de taille des éléments pendant la simulation. La classe correspondant aux dimensions de chaque triangle est déterminée à chaque pas d'intégration temporelle Δt. Chaque triangle est ensuite projeté dans l'espace des voxels associé à sa classe à l'instant t. La figure 5.2 illustre cette insertion qui est détaillée dans la section 5.3.3.

Pour satisfaire la définition 5.2, la dimension d'un triangle ne doit pas excéder la dimension de la plus grande des classes, c'est-à-dire C^{k-1} :

$$\forall t \in T | dim(t) < dim(C^{k-1})$$

Ceci ne peut être garanti car il n'existe aucune contrainte sur les déformations des triangles. Une solution est d'ajouter dynamiquement une nouvelle classe de dimension supérieure, ce qui correspond à ajouter une table de hachage supplémentaire de taille m. L'allocation dynamique est un processus coûteux. Les triangles, bien que déformables, subissent rarement des transformations excessives qui doublent ou plus leur dimension. D'après ces constations, une sentinelle a été ajoutée, c'est à dire une table de hachage supplémentaire (donc $k+1$ table de hachage au total) de dimension $dim(C^k) = 2 \times dim(C^{k-1})$. Dans toutes les expérimentations, cet ajout a évité un bon nombre d'allocations dynamiques.

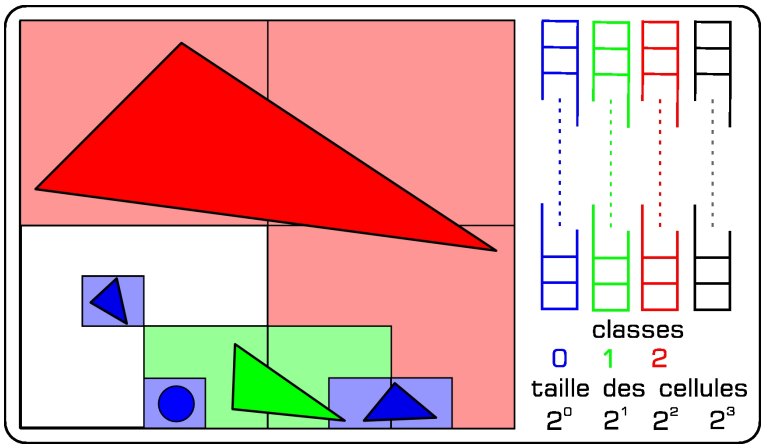

FIG. 5.2: *À gauche :* les voxels recouverts par les triangles projetés sont stockés en tant que cellules dans les tables de hachage de couleurs correspondantes. *À droite :* Chacune des 4 tables représente une classe (avec la sentinelle).

5.3.2 Réduction des conflits de hachage grâce à la hiérarchie

La fonction de hachage est une fonction non injective et non surjective. C'est-à-dire qu'à une cellule d'une table peut correspondre plusieurs voxels et donc, grâce à la non injectivité de la fonction, les tables de hachage ne bornent pas l'espace de simulation. Cependant cette relation engendre un problème : les conflits de hachage. Un conflit de hachage est issu de la définition même de la non injectivité : le stockage de plusieurs voxels dans la même cellule est nommé conflit ou collision de hachage. Ce genre de situation survient même lorsqu'il existe des cellules vides. Quel est le problème lié à cette coexistence ? Le but de cette structure accélératrice est de déterminer, en temps minimisé, le proche voisinage d'un élément quelconque, c'est-à-dire un triangle, une particule, etc.. Le proche voisinage d'un élément est normalement contenu dans sa propre cellule qui correspond dans l'espace aux éléments dans le même voxel. Bien que vrai, il existe cependant des intrus, c'est-à-dire des éléments situés en dehors du proche voisinage mais présent dans cette cellule à cause des conflits de hachage. Cette intrusion accroît bien évidemment les traitements effectués ultérieurement sur l'ensemble des éléments présents dans la liste. Malgré tout, les tables de hachage font parties des structures les plus efficaces pour déterminer le proche voisinage.

Il existe des travaux qui évite ce genre de conflits, en allouant un espace nécessaire

et suffisant, c'est-à-dire minimal, pour que la fonction de hachage soit injective, voir bijective. Sylvain Lefebvre et Hugues Hoppe [60] ont "haché parfaitement" des objets indéformables en utilisant deux tables de hachages minimales et ce, sans conflit d'où le terme "parfait". Leur méthode construit les tables dans une phase de précalcul pour chaque maillage. Ce précalcul, bien que rapide, est malheureusement trop important pour des applications interactives s'il doit être effectué à chaque pas d'intégration temporelle, ce qui est le cas pour les objets déformables. Même si le but premier de cette thèse n'est pas d'obtenir des algorithmes qui fonctionnent en temps interactifs, il est important de minimiser au maximum les temps de calculs.

La méthode proposée ci-dessous vise à réduire les conflits de hachage. Le principe est d'utiliser des informations précédemment calculées pour éviter un conflit. Ce procédé ne nécessite aucun traitement supplémentaire, simplement la sauvegarde d'un élément en cours de calcul.

Une fonction de hachage prend en paramètre une position $\mathbf{p}$ dans l'espace 3D et retourne une clé de hachage, c'est-à-dire un nombre entier naturel. Cette clé est calculée à l'aide de l'ordre de la classe et de la taille de la table de hachage associée :

$$\tilde{h}(\mathbf{p}) = \left(\alpha \left\lfloor \frac{p_x}{s} \right\rfloor + \beta \left\lfloor \frac{p_y}{s} \right\rfloor + \gamma \left\lfloor \frac{p_z}{s} \right\rfloor\right) mod\ m \tag{5.2}$$

où α, β, γ sont des grands nombres premier (par exemple 73856093, 19349663, 83492791, voir aussi les travaux de Matthias Teschner *et al.* [98] et de Kevin Steele *et al.* [94]), s, m sont respectivement la dimension de la classe et la taille de la table de hachage, p_x, p_y, p_z sont les coordonnées spatiales du point $\mathbf{p}$.

Le terme de requête est défini dans le restant de ce chapitre comme toute insertion, récupération ou questionnement d'éléments dans la structure de données du modèle d'optimisation.

Une requête appelle la fonction de hachage pour chaque classe visitée par ordre décroissant en commençant par la classe du plus haut rang k jusqu'à la classe de rang c qui correspond à la dimension de l'élément faisant l'objet de la requête. Suivant ce procédé $k - c$ clés sont calculées. La définition 5.1 formule une relation mère-filles entre les voxels associés aux classes. Les éléments situés dans une cellule avec une clé

de hachage $\tilde{h}^c$ sont selon la définition également situés dans la cellule "mère" avec la clé $\tilde{h}^{c+1}$. À cause d'un conflit de hachage, un élément issu d'une cellule avec la clé $\tilde{g}^{c+1}$ est inséré dans la cellule avec la clé $\tilde{h}^c$. Pour éviter ce conflit, les clés issues des cellules mère, c'est-à-dire $\tilde{h}^{c+1} \neq \tilde{g}^{c+1}$, sont utilisées pour décaler les cellules filles qui vont stocker les éléments, c'est-à-dire $\tilde{h}^{c+1} + \tilde{h}^c \neq \tilde{g}^{c+1} + \tilde{h}^c$. Ce procédé est illustré par la formule suivante :

$$h^c(\mathbf{p}) = \left(\sum_{i=k}^{i=c} \tilde{h}^i(\mathbf{p}) \right) \ mod\ m \tag{5.3}$$

Chaque élément est ensuite inséré dans la classe c à l'aide de la clé h^c. Cette méthode nécessite de multiples appels à l'équation 5.3, au plus $k+1$ appels. Cependant cette surcharge évite de futurs calculs inutiles liés à un proche voisinage erroné. Il est important de noter que rien ne garantit la disponibilité de la cellule demandée à la suite du décalage. Si la cellule est indisponible, alors il y aura un conflit de hachage. Toutefois, les expérimentations ont montré qu'avec cette extension les calculs sont un peu plus rapides. Cette accélération est cependant difficile à quantifier car elle est très dépendante de la constitution de la scène.

Ce procédé est catégorisé en tant que méthode à double hachage, qui vise à réduire les collisions de hachage. Des explications supplémentaires figurent dans les travaux de George Lueker et de Mariko Molodowitch [65]. Une autre façon simple mais efficace de réduire la probabilité de conflit est d'agrandir la table de hachage, ce qui nécessite bien sûr plus de ressources mémoire mais allège les temps de calculs. Cependant, l'accès à des éléments en mémoire est beaucoup plus rapides lorsque ceux-ci sont contigus. Suite à une requête, Müller *et al.* [75] ont observé une accélération de facteur dix en copiant le résultat en mémoire (contigu) par rapport à un accès par pointeur dans la structure (non contigu).

5.3.3 Insertion dans la table de hachage

Le processus d'insertion dans la table de hachage est effectué en trois étapes :

1. sélection de la classe et de la table souhaitée ;

2. détermination de la clé ;
3. insertion dans la liste d'éléments contenus dans une cellule, qui peut nécessiter une allocation mémoire.

Uniquement deux types d'éléments sont considérés dans le reste de ce chapitre, c'est à dire les triangles en sous-section 5.3.3.1 et les particules/sommets en sous-section 5.3.3.2. Dorénavant le terme élément fera référence à l'ensemble des triangles et des particules/sommets. Étendre cette insertion à d'autres éléments tels que des arêtes ou des tétraèdres est triviale : il suffit de calculer la boite englobante de l'objet, cette boite est ensuite projetée dans la grille implicitement représentée par la table.

La méthode d'insertion des éléments, qui fait office d'élagage approximatif, est dépendante de la distance parcourue par un élément pendant un pas d'intégration temporelle. Afin de prendre en compte l'ensemble des contacts possibles entre différents éléments, il est nécessaire de vérifier l'existence d'obstacles tout au long de ce parcours. Ce procédé est appelé la détection de collisions continue. Sans ce procédé, la détection de collision est robuste si et seulement si la distance parcourue par un élément ne dépasse pas un certain seuil prédéterminée, comme dans les travaux appliqués aux interactions fluide-structure de Nobuhiro Kondoh *et al.* [58] et de Takahiro Harada *et al.* [44].

5.3.3.1 Triangles

L'insertion des triangles est effectuée comme suit :

1. la longueur l de la plus grande arête de la boite englobante du triangle est calculée ;
2. la dimension d du triangle est déterminée à l'aide de l et de l'équation 5.1 ;
3. cette dimension d permet de sélectionner la classe optimale c et par conséquent la table de hachage associée pour ranger le triangle. Cette classe optimale est trouvée lorsque qu'une classe $c-1$, en partant de k, est de dimension $dim(c-1) \leq d$. La classe optimale pour le triangle est la classe c avec la clé correspondante h^c ;

4. la clé de hachage est calculée à l'aide de l'équation 5.3 ;

5. le triangle est finalement inséré dans la liste de la cellule pointée par clé.

Le paragraphe suivant détaille les quelques optimisations mises en oeuvre pour accélérer le traitement des points précédents.

Pour le point numéro 3, il est plus judicieux de mémoriser les dimensions des classes plutôt que de les recalculer à chaque requête. Cela ajoute une liste de k nombre entiers, mais évite un nombre d'appels conséquent à l'équation 5.1 et donc à la fonction $ln(x)$. Cette équation est donc uniquement utilisée lors de la phase de précalcul. En ce qui concerne le point numéro 5, j'ai ajouté à chaque cellule un marqueur, c'est à dire un booléen, qui indique la présence ou non d'une fille. Ce booléen, mis à jour pendant l'insertion des triangles, est utilisé par la suite lors des requêtes des sommets/particules. Cela permet d'éviter la vérification dans les $k - c$ tables de l'existence d'un élément si, par exemple, il est indiqué grâce au booléen que la table mère k n'a pas de filles. Ce booléen est mis à jour efficacement lorsque qu'un triangle parcourt de la plus grande à la plus petite les $k - c$ tables. Si le booléen n'est pas à jour, pour chaque table parcourue ce bit indiquant la présence d'une fille de la cellule visitée est activé. Si la cellule choisie de la table associée à la classe $c - 1$ est vide, alors le booléen de la cellule associé à la classe c est désactivé. En faisant l'analogie avec un arbre, cela correspond à avoir atteint une feuille.

En ce qui concerne l'insertion et donc le point numéro 6, une cellule est définie comme vide si elle ne contient aucun élément ou si le poinçon de temps (*timestamp*) n'est plus valide. Le poinçon de temps est une optimisation issue des travaux de Matthias Teschner *et al.* [98] qui évite un nombre conséquent d'allocations mémoire. Si un élément existe dans la liste mais est périmé, c'est-à-dire étiqueté avec un poinçon de temps dépassé, alors cet élément est écrasé par le nouveau lors d'une insertion. Bien sûr lorsque le poinçon de temps est à jour et que l'élément n'est pas dans la liste, cette insertion nécessite une allocation. Cependant l'optimisation du poinçon de temps réduit considérablement ces allocations qui font chuter les performances. Dans les premières secondes de simulation, généralement les listes dans les cellules nécessitent des allocations lors des requêtes d'insertions. Après ces quelques secondes,

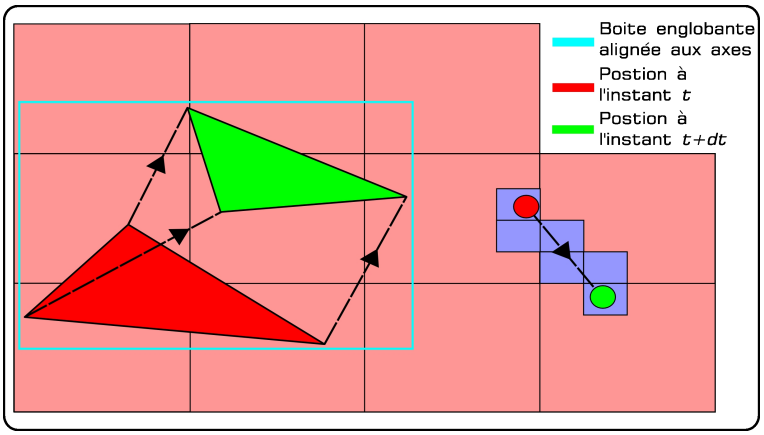

FIG. 5.3: La boite englobante du prisme est utilisée pour détecter toutes les collisions potentielles.

les insertions se font rares et donc les performances sont quasi-optimales.

Dans le cas où la dimension de la plus grande classe (en incluant la sentinelle) est plus petite que la dimension du triangle l, c'est-à-dire $dim(C^k) < l$, alors la projection du triangle dans la grille de voxels ne sera pas optimale. Cette surcharge est proportionnelle à $\left(l/dim(C^k)\right)^3$.

De la définition 5.2 se déduit qu'un triangle statique défini dans un espace tridimensionnel ne peut recouvrir qu'au maximum 8 cellules. Mais le chemin parcourut par ce triangle pendant un pas d'intégration temporelle peut en recouvrir d'avantages. un déplacement linéaire du triangle est assumé pendant un pas d'intégration temporelle. Le volume résultant de cette extrusion est un prisme (schématisé en 2D dans la figure 5.3). La boite englobante alignée aux axes est calculée pour ce prisme et chaque cellule recouverte par cette boite englobante est soumise à une insertion. En pratique dans tous nos tests les triangles ont recouvert entre une et dix-huit cellules. Ce nombre n'est pas excessif et est expliqué par le fait que les objets déformables, souvent assez rigides, contraignent le pas d'intégration temporelle a être petit. Dictés par ce petit pas, les déplacements des triangles sont par conséquent modérés.

Les éléments sans épaisseur, tels que les triangles, ou les arêtes d'un triangle sont susceptibles d'être soumis à des erreurs numériques. Il n'est pas rare qu'une arête soit sur une bordure d'un voxel. Lors du calcul du positionnement de l'arête par

rapport à la cellule, une erreur numérique peut identifier cette arête comme étant à droite de la bordure. Une particule, qui est aussi sur la bordure de ce même voxel, pourra être identifiée à gauche de la bordure, toujours à cause d'un arrondi numérique. Puisque que ces deux éléments ne sont pas situés dans le même voxel, ils ne sont pas soumis ultérieurement à une vérification de collision. Ce genre de situation n'est pas rare, notamment lorsque les triangles sont alignés sur les axes ; les particules passent alors au travers de l'arête adjacente aux deux triangles. Pour faire face à ce problème, les boites englobantes des triangles sont élargies de ϵ. Cette précaution évite les "recouvrement erronés" dûs aux erreurs numériques.

Les triangles sont insérés dans les tables en premier lieu. En second lieu les particules et sommets sont traités, c'est-à-dire soit insérés dans les tables ou soit simplement vérifiés pour d'éventuelles collisions. Il n'y a pas réellement d'ordre dans le cadre d'interaction fluide-structure, les triangles et les particules doivent être insérés pour pouvoir par la suite accéder à tous les éléments triés.

5.3.3.2 Particules et sommets

Les particules et les sommets sont deux entités qui ont des caractéristiques équivalentes : ce sont des positions qui véhiculent des informations telles que la couleur, la vitesse, la pression, etc.. Cependant un sommet fait référence à un sommet d'un triangle auquel sont connectées au moins deux arêtes et une face.

Les sommets et les particules peuvent être traités de manière identique. Dans le but d'optimiser certains calculs, ces entités sont différenciées : Les particules de fluides SPH, faisant parties des méthodes Lagrangiennes sans maillage, nécessitent la connaissance de leur proche voisinage pour calculer leur accélération, d'après les équations de Navier-Stokes (voir section 2.1.2). En effet, cette accélération, qui est utilisée pour calculer la vitesse et la prochaine position des particules, est déterminée par la vitesse des particules voisines ou force de viscosité, par l'accumulation des particules qui engendre une force appelée pression et par des forces extérieures comme la gravité ou la tension de surface.

La connaissance du voisinage est donc indispensable pour ces particules. Pour

optimiser cette recherche, l'insertion des particules dans les tables de hachage réduit la complexité du problème de recherche du proche voisinage de $\Theta(n^2)$ à $\Theta(v \times n)$, où v est le nombre de voisins.

Les objets faisant partis des modèle Lagrangiens avec maillage, tels que les systèmes masses-ressorts, ont connaissance à tout moment des sommets, arêtes et faces adjacents. L'utilisation d'une structure accélératrice pour déterminer leur proche voisinage géométrique, pour par exemple calculer l'influence d'un ressort sur ses deux extrémités, est dénuée de sens puisque ce voisinage géométrique est intrinsèque au modèle Lagrangien. Les contacts triangles-triangles peuvent être réduits aux contacts sommets-triangles. Bien qu'un algorithme robuste de collision doit prendre en compte les interactions arêtes-arêtes, ce système permet de reproduire des comportements simples mais satisfaisant pour mon cadre de travail. J'ai donc choisi de soumettre les sommets des objets à une vérification de collisions, sans insertion puisque que les sommets n'ont pas besoin d'informations de voisinage. Bien évidemment, éviter des insertions permet de réduire des allocations mémoires et donc de diminuer les temps de calculs. Ces interactions triangles-triangles sont donc obtenues à faible coût.

Après cette vérification pour les sommets et insertion pour les particules, les traitements ultérieurs sont identiques. Cette structure accélératrice est donc générique pour tout type d'objet disposant de sommets et/ou de particules.

Uniquement les particules voisines situées dans un rayon r ont une influence sur une particule donnée. Ce rayon est un paramètre de la méthode SPH. La dimension optimale de la classe est donc $C^p = dim(r)$. Si cette classe n'existe pas dans la liste des classes, alors elle est ajoutée ainsi que la table de hachage associée. Les simulations de dynamique des fluides réalistes requièrent la connaissance du voisinage dans un petit rayon. Par conséquent, la classe C^p est généralement la plus petite des classes. Pour des raisons de bon fonctionnement avec la structure hiérarchique, expliquées dans les paragraphes suivants, cette classe C^p **doit** être la plus petite classe.

L'insertion des particules est effectuée tout comme les triangles, c'est-à-dire en utilisant le schéma de double hachage présenté en section 5.3.2.

Pour chacune des classes, une clé de hachage est calculée. Une cellule pointée par cette clé qui n'est pas vide, c'est à dire qui contient des triangles rangés dans la

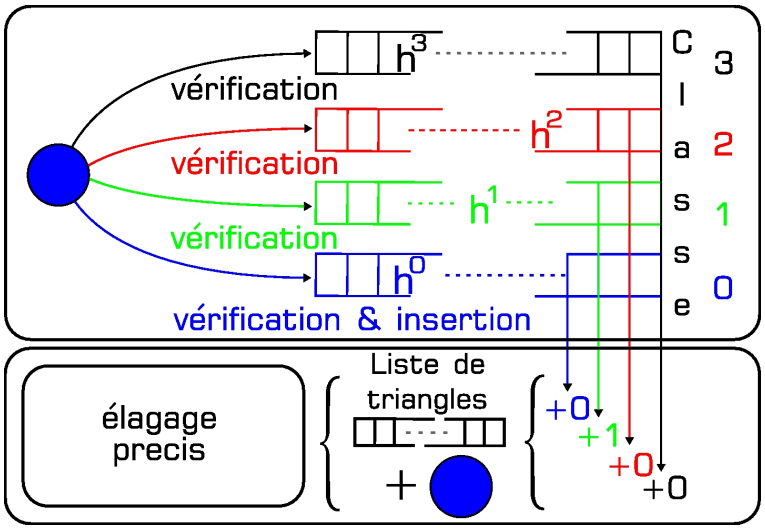

FIG. 5.4: *Haut :* insertion d'une particule par ordre décroissant de grandeur de classe, issue de la scène illustrée en figure 5.2. *Bas :* Les triangles situés dans les cellules parcourues par la particule sont étiquetés et injectés dans la phase d'élagage précis.

.

classe associée, indique que la particule et les triangles sont proches et éventuellement en contact. Alors pour chaque élément de la liste une collision est balisée. Elle sera traitée ultérieurement par la phase d'élagage précis. Lorsque la classe C^p est atteinte, la particule est insérée dans sa table associée.

Si la classe C^p n'est pas la plus petite des classes, puisque la vérification de collisions est effectuée par ordre décroissant en partant de la plus grande des classes, alors les classes d'ordres inférieures à C^p ne seront pas traitées et des collisions seront omises. La simple solution d'examiner les classes inférieures engendre des résultats incohérents, puisque la définition 5.1 n'est pas respectée. Le décalage effectué par la fonction de double hachage échoue car des particules situées dans des cellules filles de clés différentes se retrouvent alors dans de mauvaises cellules mères à cause du décalage. Le problème est dans ce cas inversé. Donc, d'après les définitions, C^p **doit** être la plus petite des classes. La figure 5.4 schématise l'insertion d'une particule de l'exemple illustré dans la figure 5.2.

Pour détecter toutes les collisions possibles d'une particule en déplacement pendant un pas d'intégration temporelle, tout le chemin parcouru par cette particule

doit être soumis à une vérification. Ceci correspond à tracer une ligne dans un espace discrétisé, implicitement représenté par la table de hachage associée à la classe C^p. Pour ce faire l'algorithme de Bresenham 3D (détaillé en 2D dans l'ouvrage de James Foley *et al.* [51]) est utilisé. Ce procédé est illustré à droite dans la figure 5.3.

5.4 Résultats et comparaisons

Tous les résultats présentés ci-après ont été réalisés sur un ordinateur cadencé à 2 GHz et disposant du système d'exploitation Linux.

Dans cette section figure des comparaisons entre le modèle hiérarchique proposé dans ce chapitre et le modèle dit de référence de Matthias Teschner *et al.* [98]. L'utilisation des kd-trees pour la recherche du proche voisinage dans les simulations de fluide est récente. Les travaux de Richard Keiser [55] et de Bart Adams [3] ont été effectués en parallèle des nôtres. Faute de temps, des comparaisons entre les kd-trees et les tables de hachage hiérarchiques ne figurent pas dans cette thèse.

La première scène de test, illustrée dans la figure 5.5, est composée d'environ $1k$ triangles, d'approximativement $1k$ sommets et $1k$ particules de fluide.

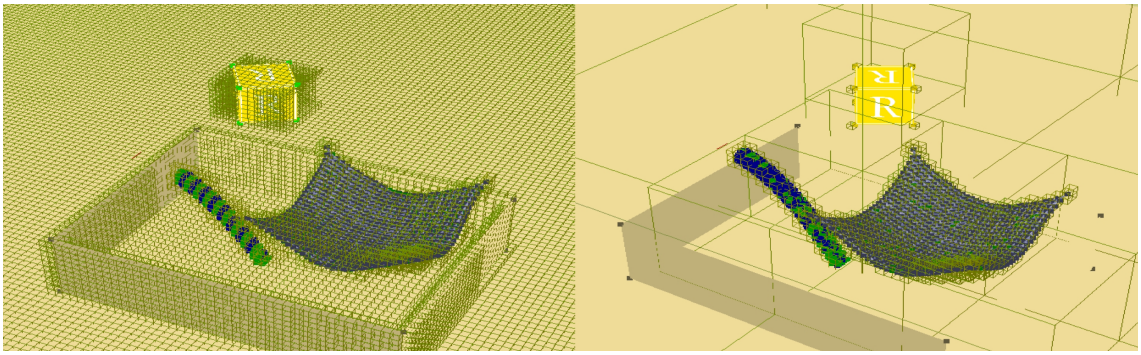

FIG. 5.5: Les éléments d'une scène sont projetés dans une grille représentée *À gauche :* par une table de hachage unique *À droite :* par le modèle de hachage hiérarchique.

Le modèle présenté par David Baraff et Andrew Witkin [6] a été utilisé pour simuler le tissu. La structure du drap est constituée de masses et de ressorts qui est intégrée dans le temps à l'aide d'une méthode implicite (voir la sous-section 2.3.2).

L'extension du modèle des SPH présentée par Matthias Müller *et al.* [75] a été employée pour simuler la dynamique du fluide.

Ci-dessous sont présentés des temps de comparaisons des structures accélératrices qui incluent les temps de création, de précalculs et de requêtes, c'est-à-dire insertions, vérifications et détermination du voisinage pour la détection de collisions. L'ensemble de ces temps est désigné en tant que C&R (Création et Requête).

Le modèle de référence nécessite une paramétrisation manuelle, contrairement au modèle hiérarchique proposé qui se paramétre automatiquement. La dimension de la cellule optimale, c'est-à-dire qui propose les temps de calculs les plus courts, est donc obtenue expérimentalement. Pour ce faire elle est testée de 2^{-3} à 2^{3} (qui fait office de premier rang dans le tableau de comparaisons présenté en figure 5.6).

Le temps passé pour C&R le modèle de référence est long lorsque :

1. les boites englobantes des triangles sont projetées dans des cellules minuscules, ce qui engendre des parcours volumiques coûteux et beaucoup d'insertions ;
2. Le proche voisinage est de plus en plus important lorsque les cellules sont de plus en plus grandes. En fonction de cette taille, ce voisinage peut être dans le pire des cas étendu à tous les éléments présents dans la scène.

Dans la première scène de test, la dimension optimale obtenue manuellement est de 2^{-1}. Cette taille correspond à la dimension des triangles du tissu qui représentent 99% des polygones de la scène. La dimension de la cellule a une influence majeure sur les temps calculs de la dynamique du fluide, visible dans la ligne SPH des tableaux de comparaisons. Ces temps comprennent la récupération des particules voisines et les calculs des forces des particules. Plus la dimension des cellules est importante, plus le nombre de voisins sera conséquent, augmentant ainsi le temps passé à calculer les forces. Le rayon du noyau des SPH est gardé constant pendant les tests.

En ce qui concerne le modèle hiérarchique, la dimension des cellules est obtenue par un procédé automatique et n'est pas modifiable manuellement. Seule la dimension de la classe C_p des particules a été modifiée, dans la limite de 2^{-1}, dimension des plus petits triangles, afin de respecter la définition 5.1. Les deux dernières lignes des tableaux contiennent le temps total consommé par les deux structure accélératrices.

Les cases grisées indiquent les meilleurs temps et révèlent que le modèle hiérarchique est plus de trois fois plus rapide que le modèle de référence.

La deuxième scène de test est illustrée dans la figure 5.1. Cette scène est intéressante car les assiettes colorées comme des damiers sont constituées du même nombre de triangles. En partant du haut, chaque assiette est doublée en taille. Par conséquent les triangles sont uniformément répartis dans les différentes classes de la structure accélératrice hiérarchique. Cette scène est composée d'environ $2k$ triangles, d'approximativement $1k$ sommets et $1k$ particules de fluides. Les temps de comparaisons indiquent que la méthode hiérarchique est trois fois plus efficaces que le modèle de référence.

Il est important de noter que ces temps sont dépendants du pas d'intégration temporelle et de la taille des tables de hachage : plus le pas d'intégration est important, plus la distance parcourue par les éléments est grande, augmentant ainsi les temps d'insertions. Également, la taille de la table de hachage a une influence importante sur le nombre de conflits de hachage : plus celle-ci est petite, plus y il aura de conflits engendrant ainsi des temps de calculs plus longs. Inversement, plus la taille de la table est grande, moins il y aura de conflits et les calculs seront alors plus optimaux.

5.5 Conclusion

Ce chapitre détaille une structure accélératrice basée sur une table de hachage hiérarchique qui a pour but d'optimiser :

- la recherche du proche voisinage pour les particules de fluide ;
- la détection de collision ;
- l'extraction de surface.

Les comparaisons par rapport au modèle de référence qui utilise une unique table de hachage montrent que le modèle proposé est trois fois plus efficaces. Il est cependant $k + 1$ fois plus gourmand en terme de ressources mémoires.

Cependant, ce modèle peut être amélioré dans des travaux futurs. Par exemple, il serait de rigueur d'analyser le schéma de double hachage. Bien que difficile car les deux fonctions sont dépendantes, cela permettrait de quantifier la réduction du nombre de

size (2^x)	-3	-2	-1	0	1	2	3
Uni hash	1,45	0,37	0,15	0,13	0,22	0,7	5,8
Hiera. hash	0,1	0,11	0,15				
SPH	0,05	0,16	0,67	1,55	3,7	13	21
Uni + SPH	1,5	0,53	0,82	1,68	3,92	13,7	26,8
Hiera. + SPH	0,15	0,27	0,82				

size (2^x)	-3	-2	-1	0	1	2	3
Uni hash	7	1,33	0,33	0,15	0,11	3	30
Hiera. hash	0,09	0,09	0,09	0.1			
SPH	0,04	0,03	0,06	0,2	0,83	2,22	4,15
Uni + SPH	7,04	1,36	0,39	0,35	0,94	5,22	34,15
Hiera. + SPH	0,13	0,12	0,15	0,3			

FIG. 5.6: Tableau de comparaison des temps de calculs entre le modèle proposé et celui de référence. Les meilleurs temps des deux modèles, surlignés en gris, révèlent que la structure hiérarchique est trois fois plus rapide. Des captures d'écran des scènes expérimentées figure dans les images 5.5 et 5.1 pour les tableaux situés en haut et en bas respectivement.

conflits de hachage, et éventuellement de l'améliorer. Ce schéma de hachage peut être étendu pour réduire d'avantage ces conflits dans le but d'obtenir une réduction totale, c'est à dire un schéma de hachage parfait et dynamique en temps raisonnables, voir interactifs. Malheureusement, Les tables de hachage sont gourmandes en ressources mémoire. C'est souvent le compromis en informatique entre occupation mémoire et rapidité des traitements. La structure hiérarchique proposée mérite d'être comparé aux kd-trees, récemment utilisés dans le contexte de la recherche du proche voisinage et qui sont plus performants que le modèle de référence pour des scènes complexes. Entre autres, pour tester l'efficacité de ce modèle, il faudrait effectuer d'avantages de comparaisons avec des scènes simples et complexes. Par ailleurs, cette approche est surement adaptée pour mettre en oeuvre des fluides Lagrangiens multi-résolutions.

Chapitre Six

Interactions fluide/membranes

Dans ce chapitre je considère un fluide de type Lagrangien sans maillage, c'est-à-dire un ensemble de particules et une membrane (objet sans épaisseur) de type Lagrangienne avec maillage, c'est-à-dire un ensemble de triangles. Je réduis l'interaction entre ces deux prétendants, qui est détaillée dans le restant de ce chapitre, à une résolution de contacts particules-triangles. Beaucoup de travaux se sont intéressés, en synthèse d'images et dans d'autres disciplines d'ailleurs, à la détection de collisions et à la résolution des contacts. Un aperçu de ces techniques mises en oeuvre en informatique graphique figure dans l'état de l'art de Matthias Teschner *et al.* [99].

6.1 Introduction

La procédure de traitement des interactions se déroule généralement en trois étapes :

1. la première étape consiste à déterminer grossièrement les paires possibles de collisions entre des éléments de la scène, qui sont pour la plupart du temps en pratique des triangles, des arêtes et des sommets/particules. Cette phase, dite de *broad phase* ou d'élagage approximatif, est abordée dans le chapitre 5. J'ai cependant décidé de compléter cet élagage dans ce chapitre, qui relève plus du détails d'interactions. Par exemple, suite à des erreurs numériques, les arêtes, ou bords des triangles, doivent être traitées avec précautions. La sous-section 6.2.3 propose une méthode qui gère ces erreurs numériques ;
2. la deuxième étape consiste à ne sélectionner, avec précisions, que les éléments

qui sont en contact. Cette phase, dite de *narrow phase* ou d'élagage précis, est abordée en section 6.2 ;

3. la troisième étape consiste à appliquer un algorithme de résolution de contacts qui doit tenir compte des lois de la physique dans le but d'obtenir un scénario réaliste. Dans le cadre de fluides Lagrangiens, la friction simultanée de plusieurs particules sur un même triangle est un cas classique. Un algorithme de résolution d'impacts multiples est présenté dans la sous-section 6.2.2. Un modèle d'échange de moments est proposé dans la sous-section 6.3.

La déformation de membranes, qui sont des objets sans épaisseur, ajoute une difficulté à la détection de collision. Je propose en section 6.4 un modèle d'épaississement de surface par une carte de hauteur qui vient pallier cette difficulté[1]. Pour permettre une paramétrisation des collisions plus intuitive, la sous-section 6.4.2 présente un modèle basé sur des cartes qui contiennent différents attributs physiques. Grâce à ce procédé, les frictions perpendiculaire et tangentielle peuvent être hétérogènes sur une même surface, offrant ainsi à l'utilisateur une gestion intuitive de scénarii d'interactions.

6.2 Détermination temporelle des impacts

La sous-section 6.2.1 aborde la détermination temporelle du contact entre une particule et un triangle et en sous-section 6.2.2 entre plusieurs particules et un triangle simultanément. Une attention particulière est portée en sous-section 6.2.3 sur les arêtes qui, à cause d'erreurs numériques, laissent traverser des particules entre deux triangles.

6.2.1 Détermination de pénétration particule/plan

Le modèle présenté dans cette section pour déterminer si une particule entre en contact avec un triangle est relativement simple. Le déplacement d'une particule

[1] en fait le problème est repoussé à de plus grandes distances de déplacement des objets à chaque itération.

pendant un pas d'intégration temporelle, ainsi que celui d'un triangle déformable, sont supposés être linéaires.

L'algorithme d'élagage approximatif retourne une liste de candidats potentiels pour un contact. Pour chaque paire d'éléments triangle/particule, une évaluation de distance algébrique, par un simple produit scalaire, est faite entre :

- (a) la position précédente de la particule et un plan, défini par le triangle avant son déplacement ;
- (b) la position précédente de la particule et un plan, défini par le triangle après son déplacement ;
- (c) la nouvelle position de la particule et un plan, défini par le triangle après son déplacement.

En comparant les signes de ces trois évaluations, une collision est marquée entre ces deux éléments si :

$$(signe(a) \geq 0 \ \& \ signe(b) \leq 0) \mid (signe(b) \geq 0 \ \& \ signe(c) < 0) \tag{6.1}$$

Deux exemples sont schématisés dans la figure 6.1. Ce procédé simpliste permet de limiter le nombre d'appels à la fonction de résolution d'équations cubiques, présentée dans la section suivante, qui détermine précisément le moment de l'impact, mais qui est assez coûteuse en terme de temps de calculs.

6.2.2 Impacts multiples

La gestion des impacts multiples présentée dans cette section est une extension de la méthode basée-impulsion proposée par Simon Clavet *et al.* [20]. Cette extension a pour but de prendre en compte les membranes déformables dans le cadre d'interactions avec des particules.

Le moment de l'impact, ou *Time Of Impact* (TOI), entre une particule et un triangle en déplacement est calculé de la manière suivante[2] : soit $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$ les positions respectives des trois sommets du triangle et de la particule. Le vecteur

[2] d'après les travaux de Xavier Provot [84], de Robert Bridson *et al.* [12] et de Kenny Erleben *et al.* [27].

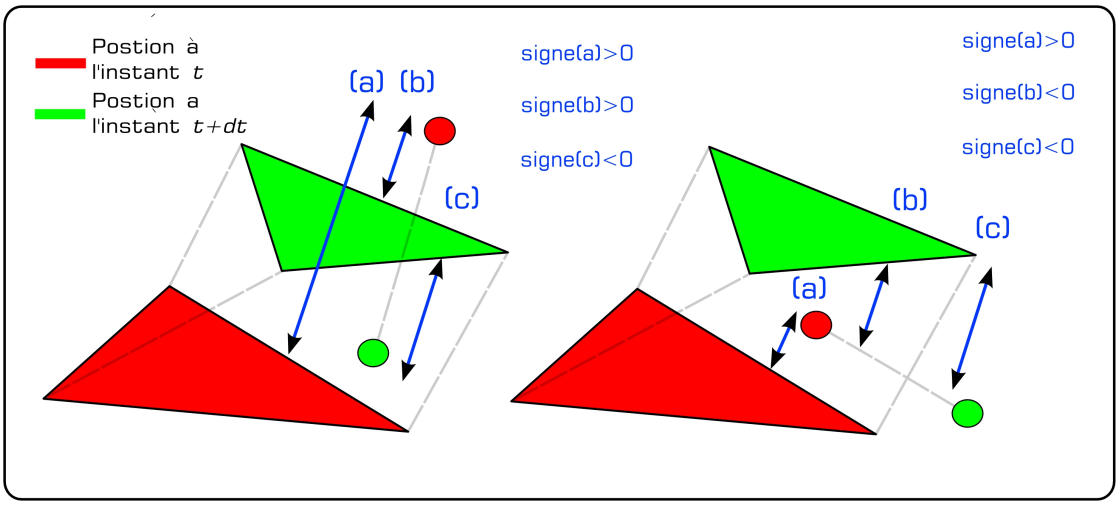

FIG. 6.1: Deux cas possibles de contacts entre un triangle et une particule pendant un pas d'intégration temporelle.

normal $\mathbf{n}$ au plan dans lequel est situé le triangle est calculé par le produit vectoriel suivant :

$$\mathbf{n} = (\mathbf{x}_2 - \mathbf{x}_1) \times (\mathbf{x}_3 - \mathbf{x}_1) \tag{6.2}$$

En prenant au hasard le point $\mathbf{x}_1$ qui appartient au plan, la distance d par rapport à l'origine est :

$$d = \mathbf{n} \cdot \mathbf{x}_1 \tag{6.3}$$

La particule est en contact avec le plan si la distance les séparant est nulle :

$$(\mathbf{n} \cdot \mathbf{x}_4) - d = 0 \tag{6.4}$$

En substituant et en considérant que les positions sont dépendantes du temps :

$$(\mathbf{x}_2^{t+\Delta t} - \mathbf{x}_1^{t+\Delta t}) \times (\mathbf{x}_3^{t+\Delta t} - \mathbf{x}_1^{t+\Delta t}) \cdot (\mathbf{x}_4^{t+\Delta t} - \mathbf{x}_1^{t+\Delta t}) = 0 \tag{6.5}$$

En posant :

$$\mathbf{x}^{t+\Delta t} = \mathbf{x}^t + \Delta t \mathbf{v}^t \tag{6.6}$$

$$\mathbf{x}_{\alpha,\beta} = \mathbf{x}_\alpha - \mathbf{x}_\beta \tag{6.7}$$

$$\mathbf{v}_{\alpha,\beta} = \mathbf{v}_\alpha - \mathbf{v}_\beta \tag{6.8}$$

où $\mathbf{v}$ représente la vitesse au point $\mathbf{x}$ et en substituant encore une fois l'équation devient :

$$(\mathbf{x}^t_{2,1} + \Delta t \mathbf{v}^t_{2,1}) \times (\mathbf{x}^t_{3,1} + \Delta t \mathbf{v}^t_{3,1}) \cdot (\mathbf{x}^t_{4,1} + \Delta t \mathbf{v}^t_{4,1}) = 0 \tag{6.9}$$

Une solution analytique de cette équation cubique existe. L'objectif est de trouver le plus petit temps $0 \leq t < \Delta t$, c'est-à-dire l'intersection la plus proche dans l'ensemble des trois racines solutions. Comme précisé par Robert Bridson *et al.* [12], une collision peut être dissimulée à cause d'une erreur d'arrondi lorsque $t = \Delta t$. Il est dans ce cas recommandé d'effectuer un test supplémentaire.

Ce TOI définit une collision entre la particule et le plan en mouvement, non pas le triangle. Pour déterminer si la particule intersecte le triangle, ces deux éléments sont déplacés dans un premier temps selon le TOI, puis, dans un deuxième temps, une évaluation barycentrique indique si la particule est située à l'intérieur du triangle.

C'est seulement à cette étape qu'une collision est marquée. Cependant, lorsque des particules reposent sur un triangle, le TOI $\simeq 0$ et donc la simulation est "gelée". Il est alors important d'utiliser un pas temporelle adaptatif. Entre autres, afin de n'avoir qu'un seul contact par itération, le TOI **doit** prendre le plus petit temps calculé pour l'ensemble des particules/sommets. Lorsque des éléments sont continuellement en contact, comme par exemple un drap sur une table ou un liquide dans un verre, la simulation est ralentie à cause de ce procédé de détection de collision.

Pour palier ce problème, je propose de conserver un pas de temps constant en approximant les collisions. La méthode fonctionne de la manière suivante : les triangles sont dans un premier temps déplacés selon leur modèle dynamique, sans prendre en compte les collisions. Ils sont "gelés" à cette position jusqu'à la prochaine itération.

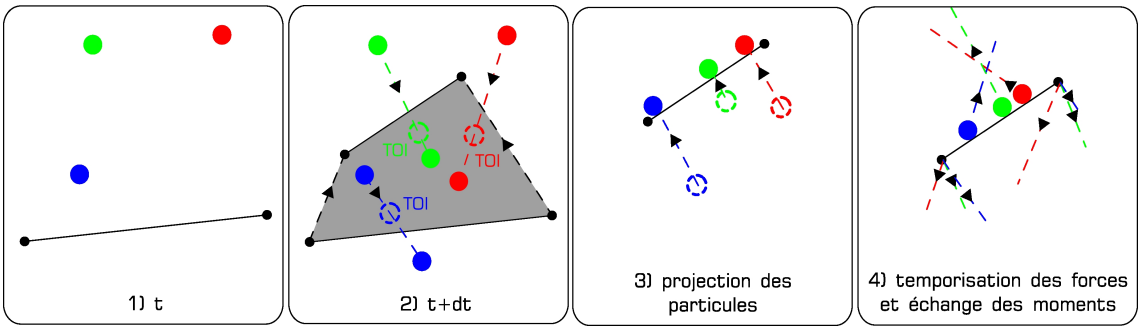

FIG. 6.2: L'approximation de la collision et la définition de tampons de forces par sommet permet de gérer les impacts multiples, mais aussi une interaction bidirectionnelle entre des particules/sommets et des triangles.

Dans un deuxième temps, les particules qui entrent en contact avec le plan sont déplacées à la position de l'impact, en utilisant le TOI, puis elles sont projetés sur la surface des triangles qu'elles viennent de pénétrer. Cette projection est effectuée à l'aide du vecteur normal du triangle. Un modèle de collision élastique permet de calculer simultanément les nouvelles vitesses de la particule et des sommets des triangles (voir la section 6.3). Les particules se voient attribuer cette nouvelle vitesse qui sera intégrée lors de la prochaine itération. En revanche, pour les triangles, cette vitesse est transformée puis accumulée dans un tampon de forces par sommet. Lorsque cette étape de collision est terminée, ce tampon de force est ajouté aux forces de chaque sommet qui seront intégrées temporellement par la suite. Ainsi faisant, plusieurs particules pourront interagir avec un même triangle et ce, durant un pas d'intégration. La figure 6.2 schématise le procédé d'interaction bidirectionnelle et la gestion des impacts multiples de cette méthode.

6.2.3 Gestion des arêtes/bordures

Une particule est définie à l'intérieur d'un triangle si ses coordonnées barycentriques, w_0 et w_1, vérifient les propriétés suivantes :

$$w_0 \geq 0, w_1 \geq 0, w_0 + w_1 \leq 1 \tag{6.10}$$

Une particule qui entre en contact avec une arête partagée entre deux triangles

ne peut être située que dans l'un ou l'autre triangle puisqu'une arête est, par nature, sans volume ni surface. Il arrive que des arrondis numériques fassent échouer les tests barycentriques, par exemple la coordonnée du premier triangle $w_0^{t1} = -10^{-10}$ et que la somme des deux coordonnées barycentriques du second soit tout juste supérieure à 1, i.e $w_0^{t2} + w_1^{t2} = 1 + 10^{-10}$. Je propose trivialement dans ce cas d'élargir les arêtes d'une valeur $\epsilon \simeq 0$. En pratique les tests barycentriques deviennent alors :

$$w_0 \geq -\epsilon, w_1 \geq -\epsilon, w_0 + w_1 \leq 1 + \epsilon \tag{6.11}$$

Cependant, cette élargissement pose à priori un problème. Quand une particule entre en contact avec une arête partagée entre deux triangles, lequel des deux est utilisé pour échanger les moments ? Le TOI pour les deux polygones est exactement le même, il faut donc faire un choix. La solution la plus judicieuse est d'échanger les moments avec les deux triangles. Cette échange est réalisé à l'aide d'un vecteur normal à la surface. Puisque deux triangles sont dans ce cas précis en contact avec une particule, le vecteur normal est par conséquent la moyenne des vecteurs normaux des deux polygones respectifs. Si un seul de ces vecteurs est privilégié, les particules seront déplacées à des positions erronées et elles passeront au travers des objets. Les collisions ne seront donc pas *bullet-proof*, schématisé dans la figure 6.3.

Ce procédé semble trivial, pourtant l'élaboration d'une détection et réaction de collision robuste est difficile même avec des exemples simples. Mon appareillage de test était constitué de deux maillages très simples, une boite est un sorte de diamant (voir la figure 6.4). L'objectif de mes expérimentations fut de laisser une particule rebondir contre les triangles, sans aucune friction, avec un pas d'intégration variable et grand, pendant une heure par exemple.

6.3 Échange des moments

Pour l'échange des moments, j'ai utilisé un modèle de "collision élastique" entre une particule $\mathbf{p}_1$ et un point $\mathbf{p}_2$ appartenant au triangle. Les nouvelles vitesses de ces deux points, respectivement $\mathbf{w_1}$ et $\mathbf{w_2}$, sont obtenues en calculant la solution du

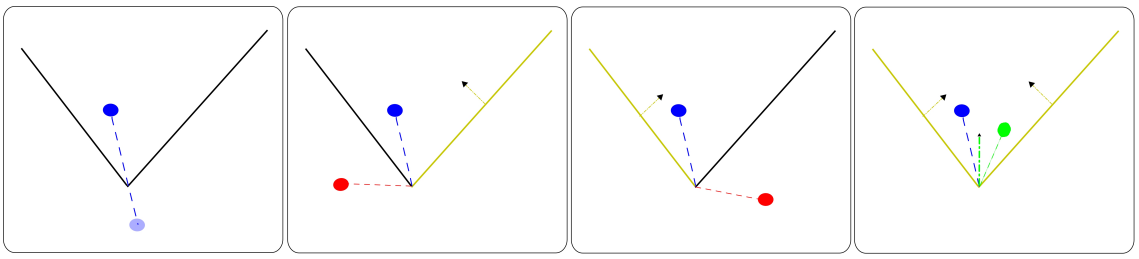

FIG. 6.3: *de gauche à droite :* une particule entre en collision avec deux triangles, exactement sur l'arête qu'ils partagent. Privilégier le vecteur normal de l'un ou de l'autre des triangles engendrera des fuites. La solution consiste à faire la moyenne de ces vecteurs.

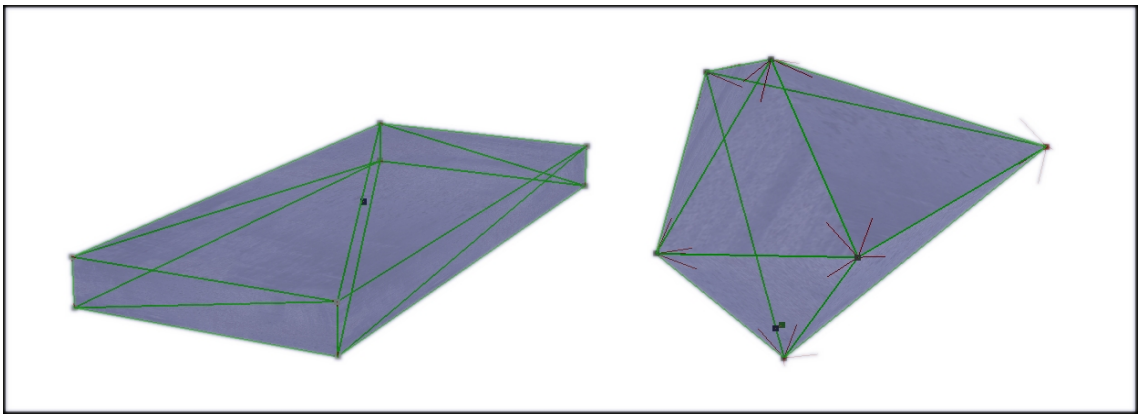

FIG. 6.4: Mon appareillage de test de collisions.

système d'équations suivant :

$$\begin{cases} m_1\mathbf{v_1} + m_2\mathbf{v_2} = m_1\mathbf{w_1} + m_2\mathbf{w_2} \\ m_1\mathbf{v_1^2} + m_2\mathbf{v_2^2} = m_1\mathbf{w_1^2} + m_2\mathbf{w_2^2} \end{cases} \tag{6.12}$$

où $m_1, \mathbf{v_1}$ (respectivement $m_2, \mathbf{v_2}$) sont la masse et la vitesse du point $\mathbf{p}_1$ (respectivement $\mathbf{p}_2$).

La vitesse $\mathbf{v_2}$ est calculée par une interpolation barycentrique des vitesses des sommets du triangle. La nouvelle vitesse $\mathbf{w_2}$ est dans un premier temps transformée en force puis dans un deuxième temps ajoutée dans le tampon de forces des trois sommets à l'aide d'une autre interpolation barycentrique. J'ai ajouté de la friction tangentielle et perpendiculaire, respectivement $f_t, f_n \in [0 \dots 1]$, à la nouvelle vitesse $\mathbf{w_1}$:

$$\mathbf{w_1'} = (1 - f_t)(\tilde{\mathbf{n}} - \mathbf{w_1}) + f_n\tilde{\mathbf{n}} \tag{6.13}$$

où $\tilde{\mathbf{n}} = (\mathbf{w_1} \cdot \mathbf{n})\mathbf{n}$, $\mathbf{n}$ est le vecteur normal du triangle. Les figures 6.5 7.6, 7.7, 7.8 et 7.9 montrent des captures d'écran d'animations obtenues par cette méthode.

6.4 Épaississement des membranes

La détection de collisions avec une membrane, à cause de son épaisseur "epsilonesque", doit être élaborée avec soins. En pratique, le déplacement des objets à chaque itération est minimisé à cause d'un pas d'intégration temporelle souvent petit. Au lieu de mettre en place des systèmes de gestion de collisions complexes pour les objets fins, il suffit de les épaissir et, en garantissant qu'un sommet ou une particule ne puisse parcourir une distance supérieure à cette épaississement par pas d'intégration temporelle, les traitements particuliers pour les membranes ne sont alors plus nécessaires. La sous-section 6.4.1 présente un modèle d'épaississement des membranes, à l'aide de cartes de hauteurs, pour détecter les collisions à moindre coût. Cette carte de hauteurs sert entre autres à définir des d'objets disposant d'un maillage détaillé qui s'affranchissent de calculs complexes pour détecter les collisions. La plupart des logi-

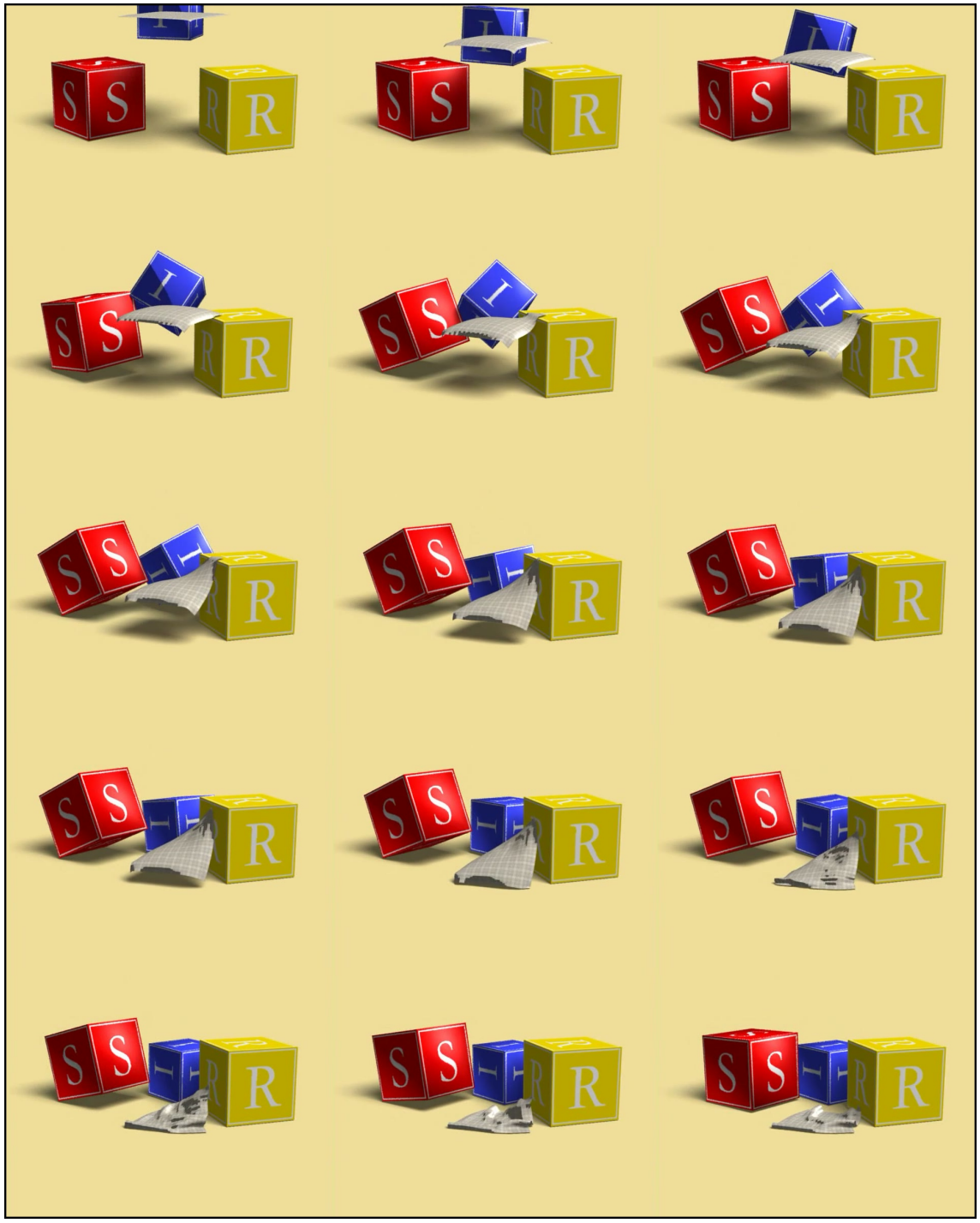

FIG. 6.5: Un drap et des boites qui entrent en interaction illustre le modèle de collision proposé dans ce chapitre.

ciels présents sur le marché de l'animation proposent de paramétrer, par objet ou par matériaux, les coefficients physiques de friction. Ces attributs sont donc homogènes pour tout le maillage. La sous-section 6.4.2 présente un simple modèle de définition de coefficients hétérogènes par le biais de cartes. De ce fait, des scénarii intéressants sont intuitivement élaborés par l'utilisateur. Ceci permet, notamment, de modéliser plus de matériaux.

6.4.1 Définition par une carte de hauteur

Une carte de hauteur est un tableau bidimensionnel qui contient des informations de hauteur. L'avantage d'un tel modèle est que des maillages haute définition sont facilement manipulés grâce à ces cartes, et notamment sur du matériel graphique dédié. L'inconvénient est qu'une seule hauteur est définissable. Des approches comme celle de Jonathan Shade *et al.* [89] et de Koichi Onoue et Tomoyuki Nishita [57] proposent des modèles de cartes à plusieurs hauteurs.

Les cartes sont utilisées comme des textures : elles nécessitent un triangle, que je nomme polygone porteur de la surface. La hauteur, en un point quelconque de ce polygone, est obtenue par un simple accès à la carte avec les coordonnées de "texture" u et v. Une particule qui entre dans le volume du polygone porteur "extrudé" balise une collision si la distance séparant le polygone porteur de la particule est inférieure à la hauteur en ce point.

Cette hauteur est trouvée en projetant dans un premier temps la position de la particule sur la surface du triangle, à l'aide de son vecteur normal, puis dans un deuxième temps en calculant les coordonnées u et v de ce point. Je souligne que ce procédé a ses limites et qu'il est plus judicieux de mettre en oeuvre une méthode comme celle présentée par Fábio Policarpo *et al.* [81], qui recherche le bon impact dans une carte de hauteur.

Le calcul des collisions nécessite la connaissance du vecteur normal à la surface. Puisque la surface planaire du triangle est modifiée par la carte de hauteur, il est impératif de calculer le vecteur normal en tout point de cette surface pour obtenir une résolution de contacts réaliste. Pour ce faire, j'ai décidé, dans une phase de pré-

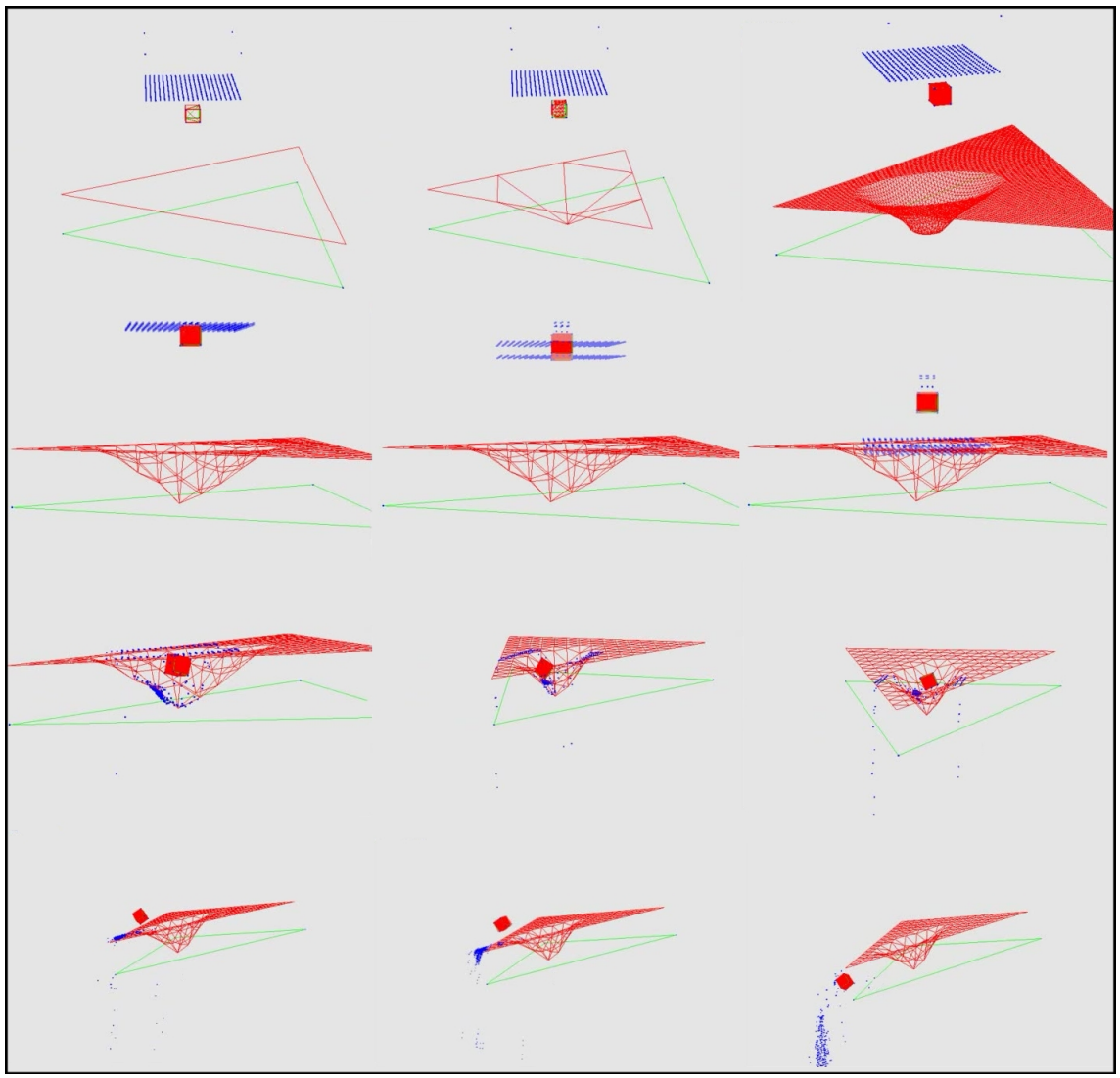

FIG. 6.6: Le triangle vert est le polygone porteur. La maillage rouge représente la triangulation de la carte de hauteur. Cette triangulation est bien sûr adaptative, comme montré dans les trois premières figures du haut. L'image en haut à droite représente le maillage à sa résolution maximale : tout les tests de collisions sont effectués sur ce maillage à cette résolution, indépendamment de l'affichage. Puisque les tests de collisions sont réduits à un simple accès de coordonnées de textures, la coût de requête est exactement le même pour une carte simple ou complexe.

traitement, de calculer tout les vecteurs normaux et de les enregistrer dans une carte des normales, en transformant les coordonnées spatiales de l'espace x, y, z à l'espace r, g, b. Lors des tests de collisions, ces vecteurs sont récupérés à l'aide des coordonnées u et v, tout comme pour la carte de hauteur. Puisque les triangles porteurs sont déformables et se déplacent en fonction du temps, le vecteur normal en un point de collision est défini comme étant la somme du vecteur normal du triangle porteur et du vecteur normal de la carte des normales (avec une normalisation bien évidemment). La figure 6.6 montre des particules et un cube[3] qui entrent en collision avec un maillage défini par une carte de hauteur.

6.4.2 Définition de la carte des forces

L'utilisation de cartes de hauteurs est simple, intuitive et efficace. Elle est employée dans de nombreuses applications, comme par exemple la modélisation de terrains, et de ce fait le modèle présenté n'est pas nouveau. Dans la continuité de l'utilisation de cartes pour définir des paramètres, comme la hauteur ou un vecteur normal, je propose d'utiliser deux autres cartes, une représentant les coefficients de frictions perpendiculaires en sous-section 6.4.2.1 et l'autre représentant les coefficients de frictions tangentielles en sous-section 6.4.2.1. Le but est de fournir à l'utilisateur une paramétrisation intuitive des paramètres de friction en les dessinant par exemple avec son logiciel favori. Les coefficients sont donc hétérogènes par surface et ainsi faisant, offrent d'avantages de possibilités d'élaboration de scénarii d'animations.

6.4.2.1 Forces perpendiculaires

Une balle de tennis ne rebondira pas de la même façon sur de la terre battue que sur de l'herbe, car la friction (perpendiculaire) est différente. Ce comportement est reproduit par la variation des paramètres de frictions présents dans l'équation 6.13. Si le coefficient de friction perpendiculaire à la surface $f_n = 1$, alors le rebond de la balle de tennis sera maximal. En revanche, si $f_n = 0$, la balle restera collée au sol.

[3] ce cube est "extrudé" par une carte de hauteur uniforme.

Ce coefficient est unidimensionnel, tout comme la hauteur et la friction tangentielle. En pratique, j'ai choisi d'enregistrer ces trois paramètres dans une seule image au format (r, g, b), la hauteur est la composante rouge, la friction perpendiculaire est la composante verte et enfin la friction tangentielle est la composante bleue. Chacun de ces paramètres est représenté par un octet, i.e. ils sont compris dans l'interval $[0 \dots 255]$. J'ai choisi de définir qu'une valeur de $f_n = 0$, est un trou, c'est-à-dire que la balle passe au travers de la surface. Un exemple de passoire est illustré dans la figure 6.7.

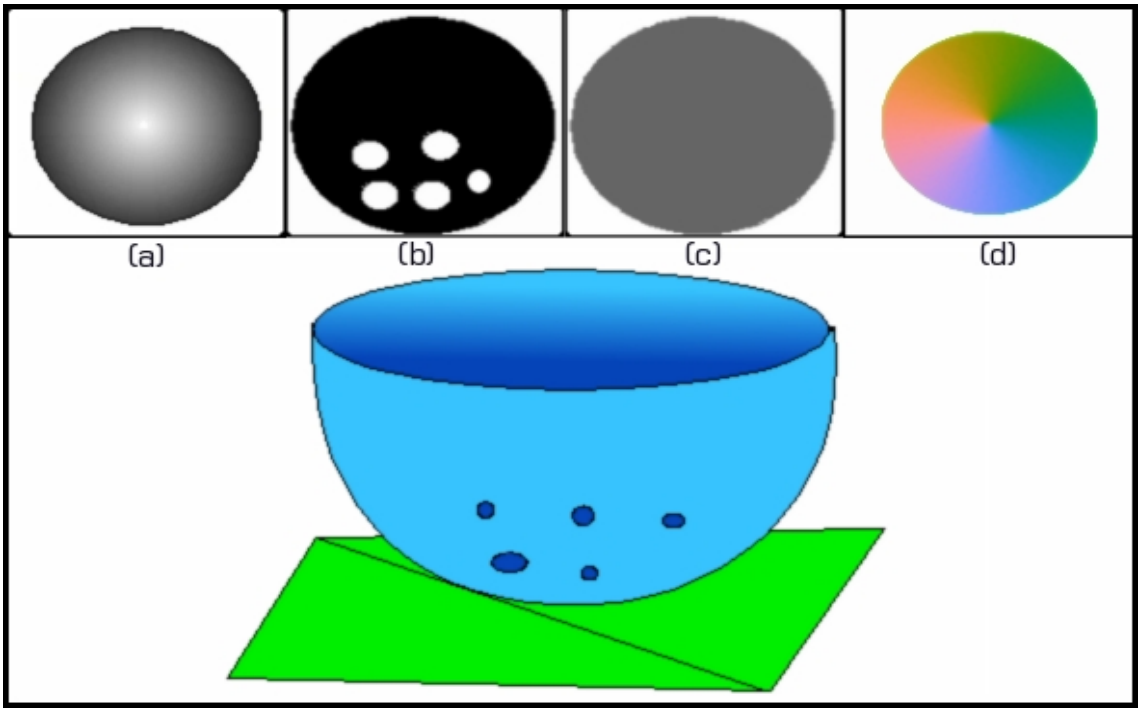

FIG. 6.7: La carte composée comprend respectivement (a) les hauteurs, (b) les coefficients de friction perpendiculaire et (c) tangentielle. La carte des normales (d) est générée dans une phase de précalculs. Le résultat de ces cartes appliquées sur le polygone porteur vert est la passoire bleue.

6.4.2.2 Forces tangentielle

L'absence de friction tangentielle est responsable du glissement d'un patin sur de la glace. Dans un tel cas cette friction est quasiment nulle. À l'inverse, la friction est

très forte sur de la moquette par exemple. De même que pour la friction perpendiculaire, la friction tangentielle est enregistrée dans une carte pour avoir des coefficients hétérogènes sur la surface. Ainsi, par exemple dans le cadre de fluide, l'utilisateur pourra dessiner dans cette carte au hasard les lettres "SIGGRAPH", en leur attribuant un coefficient de friction élevé, le reste ayant une friction quasiment nulle. De l'eau se déverse sur le sol sur lequel est plaquée cette carte de friction, ce qui a pour effet qu'une partie de ce liquide sera retenue et formera le mot "SIGGRAPH" et ce, grâce à une friction tangentielle hétérogène.

6.5 Conclusion

Dans ce chapitre sont présentés un modèle simple d'élagage approximatif, un modèle de résolution d'impacts multiples avec un pas d'intégration temporelle relativement grand, une gestion des arêtes, une méthode d'échange de moments et une définition de paramètres de friction hétérogènes par le biais de cartes. Le but de toutes ces méthodes et de pouvoir définir un algorithme de résolution de contacts robuste avec des membranes déformables et des particules de fluides sans trop de dépendances au pas d'intégration temporelle. Cependant dans cette étude, certainement trop ambitieuse, un élément important a été négligé : la différence de pression au niveau des contacts entre un fluide et une membrane engendre une accumulation des particules en un point donné. L'interaction fluide-structure ne doit pas être réduite à l'interaction particules/triangles. Le modèle de Takahiro Harada *et al.* [44] propose une solution pour conserver la pression. De plus, la projection des particules sur les surfaces des triangles à l'aide du vecteur normal a ses limites : les particules de fluides dans certains cas, car mal projetées, passent au travers des membranes. Je préconise donc d'améliorer cette projection.

Chapitre Sept

Extraction de surface autour des membranes

Dans les simulations de fluides modélisés par des structures Lagrangiennes sans maillage, c'est-à-dire des particules, la surface est rarement explicite. Elle est souvent définie par une isosurface construite à partir et autour des particules. La visualisation de ce fluide consiste à trianguler cette isosurface, qui est en quelque sorte une enveloppe autour du fluide. Puisque elle est définie autour, donc à une certaine distance des particules, dans le cadre d'interactions fluide-structure, cette enveloppe passe temporellement au travers des membranes[1]. Ce chapitre présente une méthode qui empêche cet artefact visuel.

7.1 Introduction

La visualisation de systèmes de particules est un sujet qui sollicite beaucoup d'intérêt dans les images de synthèse. Une technique qui a fait ses preuves durant les deux dernières décennies consiste à trianguler une isosurface. Cette triangulation est réalisée, par exemple, à l'aide de l'algorithme proposé par William Lorensen et Harvey Cline [61] qui s'intitule le *Marching Cube Algorithm* (MCA). Le MCA et ses dérivées, comme le *marching tetrahedras*, sont très employés aussi bien pour des applications fonctionnant en temps interactifs que pour de la visualisation *hors ligne*.

Une isosurface est un ensemble de valeurs de potentiels qui matérialise la surface

[1] une membrane est définie comme un objet sans épaisseur.

d'un objet. Une particule représente un certain volume de visualisation[2] qui est défini par une fonction scalaire, ou fonction implicite, dont la position de la particule est le centre. L'ensemble de ces fonctions implicites, évaluées dans un espace discrétisé, représente l'isosurface. L'isosurface et la surface triangulée, ou surface extraite, sont situées à une certaine distance d de l'ensemble des particules, distance qui est dépendante de la fonction scalaire, c'est-à-dire du volume représenté par la particule mais aussi de la résolution de la grille utilisée pour la triangulation. Lorsque des particules de fluides sont en contact avec une membrane, séparées par une distance $\epsilon < d$, l'isosurface et par conséquent la surface extraite vont passer temporellement au travers de cette membrane. Trois solutions vont permettre d'empêcher cet artefact visuel :

- la première consiste à garantir que la distance de contact qui sépare les particules de la membrane soit toujours supérieure à la distance séparant l'ensemble des particules de leur surface, c'est-à-dire $\epsilon > d$. Il faut cependant définir ϵ avec précautions car s'il est trop grand, la surface du fluide semblera voler au dessus de la membrane ;
- contraindre la fonction scalaire pour qu'elle intègre les contraintes spatiales engendrées par les contacts ;
- projeter le long de la membrane les triangles de la surface extraite qui passe au travers de l'objet.

Dans ce chapitre je propose de corriger les portions de surface erronées, c'est-à-dire qui passent au travers de la membrane, en les projetant le long l'objet en contact. Par ailleurs, je propose d'utiliser la structure de subdivision spatiale présentée dans le chapitre 5 pour localiser les triangles qui constituent la membrane, en section 7.2, mais aussi pour accélérer les calculs du processus d'extraction de surface, en section 7.3.

[2] qui n'est pas nécessairement en corrélation avec le volume spatial du fluide que représente la particule.

7.2 Détermination spatiale de la surface

La création de l'isosurface repose sur une fonction scalaire qui nécessite la connaissance de particules de fluide dans un proche voisinage. Puisque le MCA fait appel intensément à cette fonction, son coût doit être minimisé. La structure de subdivision spatiale présentée dans le chapitre 5 permet de connaître rapidement le proche voisinage de particules et de triangles. Je propose donc de la mettre en oeuvre pour faciliter les calculs de l'isosurface. Pour ce faire, la cellule de hachage est déterminée à partir des coordonnées spatiales de l'échantillon et le proche voisinage est défini par les particules situées dans cette cellule et dans les 26 cellules connexes (en 3D). Dans le cas où la liste des particules voisines est vide, la fonction scalaire n'a pas besoin d'être évaluée. En revanche, lorsque cette liste de voisins contient des éléments, elle est passée en paramètre de la fonction scalaire suivante (en reprenant le modèle de Simon Clavet *et al.* [20]) :

$$\phi(\mathbf{x}) = \sum_{j \in voisins} \left(1 - \left(\frac{r}{h}\right)^2\right)^{\frac{1}{2}} \tag{7.1}$$

où $r = |\mathbf{x} - \mathbf{x_j}|$, avec $\mathbf{x}$ la position de l'échantillon et $\mathbf{x_j}$ la position de la particule voisine j. Cette fonction définie un scalaire qui est proportionnel à la distance séparant l'échantillon de la particule. La fonction tend vers 1 quand la particule se rapproche de l'échantillon et inversement tend vers zéro quand lorsque la particule est à une distance h de l'échantillon.

La surface visuellement réaliste d'un fluide doit être lisse et détaillée. Une triangulation détaillée de l'isosurface requiert une grille finement discrétisée, ce qui n'est le cas même avec les petits voxels de la table hiérarchique. Par conséquent une grille supplémentaire est mise en oeuvre en plus de la table de hachage et, de part la différence de taille, un voxel de la table de hachage contient donc plusieurs voxels de la grille de l'isosurface, ou échantillons. Cette simple constation permet de mettre en oeuvre une autre optimisation, c'est-à-dire que les échantillons situés dans le même voxel de la table ont un voisinage identique, avec ou sans collisions de hachage[3] et donc dans

[3]voir la section 5.3.2 pour plus de détails.

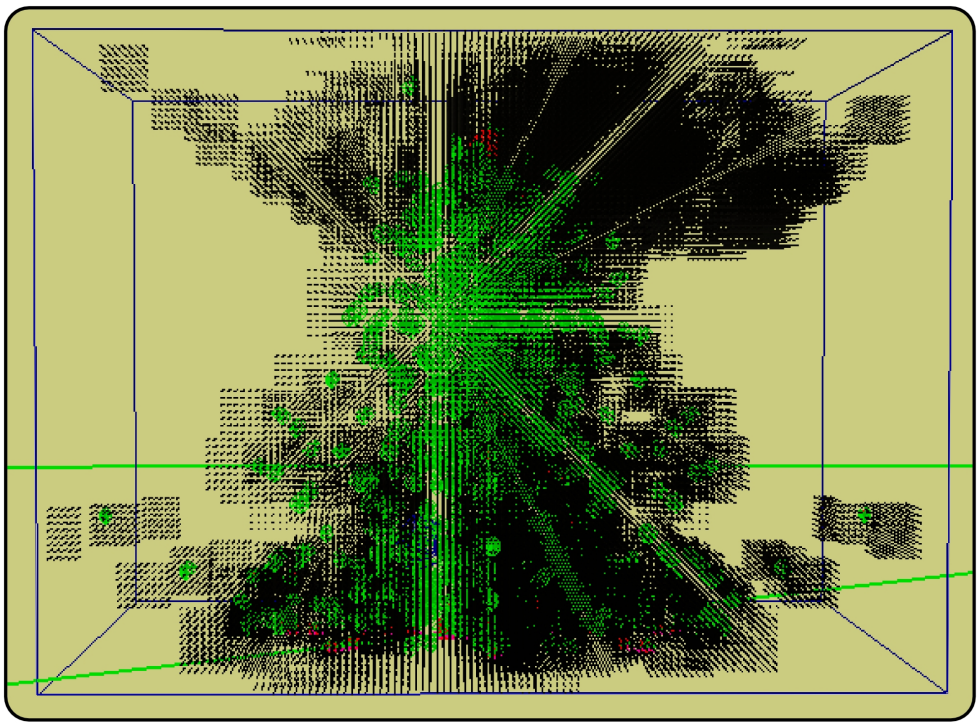

FIG. 7.1: La fonction scalaire (la surface verte) est évaluée (points noirs) à partir des particules de fluide voisines. Beaucoup de calculs inutiles sont évités à l'intérieur de la boite englobante (en bleue) grâce à l'utilisation de la structure proposée dans le chapitre 5 qui fait office de modèle d'optimisation. Des cubes de points noirs représentent des voxels de la table de hachage et, tous les points noirs ou échantillons à l'intérieur du même cube ont un voisinage identique.

ce cas précis, le voisinage précédemment déterminé est réutilisé. À titre d'exemple, la figure 7.1 illustre cette optimisation qui montre clairement que ce système permet d'éviter des appels à la fonction scalaire.

7.3 Correction de la surface erronée

Une particule de fluide est considérée comme étant en contact avec un objet lorsque la distance les séparant est inférieure à ϵ. La fonction scalaire de l'équation 7.1 définie une sphère autour de la particule de rayon h. Si $h > \epsilon$, ce qui est par ailleurs souvent le cas en pratique, une partie de la sphère sera située de l'autre coté de la bordure de l'objet. Avec des objets volumiques, ceci n'est pas gênant puisque la surface qui

pénètre à l'intérieur du volume n'est pas visible. En revanche, la surface qui traverse une membrane, puisque sans épaisseur pour cacher le volume qui a franchit la surface, engendre des animations visuellement non réalistes. Pour pallier ce problème, je propose un algorithme, dit de visibilité, qui projette sur le bord de la membrane les sommets des triangles de la surface extraite qui ont traversé cette membrane. Ce procédé est similaire à la méthode mis en oeuvre par Eran Guendelman *et al.* [43], où un algorithme de lancer de rayon est utilisé à partir de points de références.

Pour le bon fonctionnement de cette procédure, le MCA doit subir une modification : à chaque évaluation de la fonction scalaire, la position de la particule la plus proche de l'échantillon est enregistrée. Ce paramètre est le seul coût additionnel en ressources mémoire pour le bon déroulement de l'algorithme de visibilité. Le calcul de ce point le plus proche est effectué **dans** la fonction scalaire qui évalue de toutes façons les distances particules-échantillons. Ainsi faisant, il n'y a pas de coût de traitements additionnels. Ce point, nommée point de référence, va servir dans un deuxième temps pour le calcul de la projection si nécessaire. Il est normalement du bon coté de la membrane, sauf s'il y a un souci de collisions (voir le chapitre 6), mais le dans ce cas, le problème de la surface qui passe au travers de la membrane n'est pas dû à une erreur de visualisation. L'algorithme 3 récapitule le MCA modifié.

Algorithme 3 L'algorithme modifié des *marching cubes*

```
1:  pour tout échantillons i sur les axes x, y, z faire
2:     n = voisinage(i) ;
3:     position p ;
4:     i.scalaire = fluide.fonction_scalaire(n, p) ;
5:     i.point_ref = p ;
6:     i.position = position(x, y, z) ;
7:  fin pour
8:  pour tout v ∈ voxels faire
9:     si v ∈ bordure alors
10:       visibilité(v) ;
11:       trianguler(v) ;
12:    fin si
13: fin pour
```

Dans un premier temps, la procédure du MCA se déroule donc avec cette légère

modification, éventuellement en prenant en compte l'optimisation proposée dans la section 7.2. Dans un deuxième temps, tout les voxels en bordure de la surface sont traitées par l'algorithme de visibilité. Ces voxels sont définis en tant que bordure si au moins un des sommets est à une valeur α du champ scalaire et qu'au moins un des sommets à une valeur supérieure à α. Cette valeur α est un seuil souvent fixé à zéro. C'est par ailleurs le même test que celui utilisé par le MCA pour déterminer si un voxel doit être triangulé. Dans un troisième temps, la liste des triangles situés entre le point de référence et l'échantillon est récupérée à l'aide de la structure accélératrice, par le même procédé de traçage de ligne utilisé en section 5.3.3.2. Le triangle le plus proche du point de référence est sélectionné, puis dans un quatrième temps, l'algorithme de visibilité compare le positionnement de l'échantillon avec son point de référence par rapport à ce triangle. Si les deux points sont du même côté du triangle, alors la surface extraite est par défaut bien positionnée. Le cas échéant, la position de l'échantillon est projetée en utilisant une méthode de lancer de rayon sur le triangle, son attribut d'isosurface est mis à zéro pour éviter toute éventuelle triangulation ultérieure de voxels partageant ce point. Dans un dernier temps, la suite de la procédure du MCA est exécutée sans aucune modification. Ce procédé est résumé dans l'algorithme 4, schématisé en figure 7.2 et une capture d'écran en figure 7.3 illustre ces propos.

Algorithme 4 L'algorithme de visibilité

Nécessite: *voxel*

1: **pour tout** $s \in voxel.sommets$ **faire**
2: $triangles$ = détermine_triangles($s.position, s.point\_ref$) ;
3: t = proche_triangle($s.point\_ref$, triangles) ;
4: **si** côté($s.position$, t) $\neq$ côté($s.point\_ref$, t) **alors**
5: projection($s.position$, t) ;
6: $s.scalaire = 0$;
7: **fin si**
8: **fin pour**

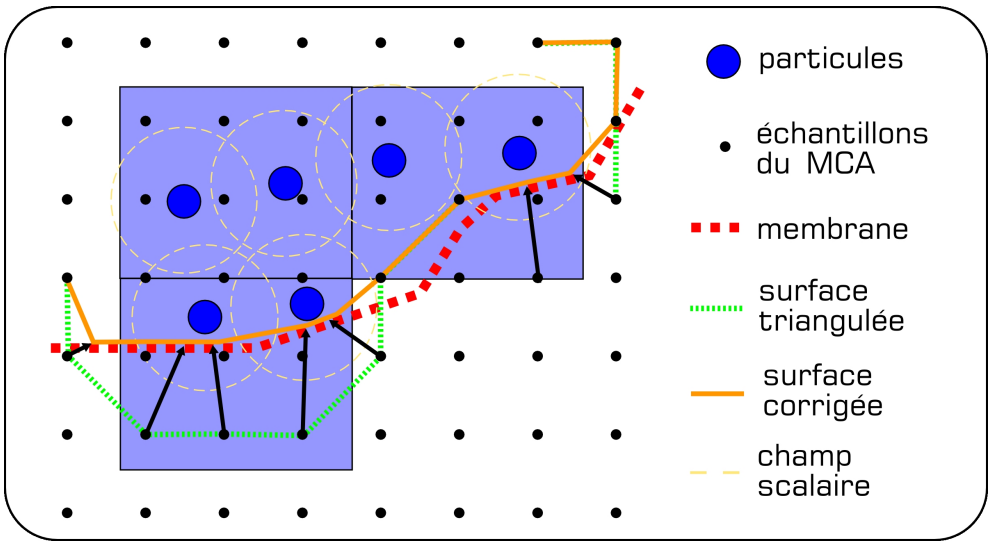

FIG. 7.2: La surface extraite (verte) qui traverse la membrane (rouge) est corrigée par l'algorithme de visibilité (orange).

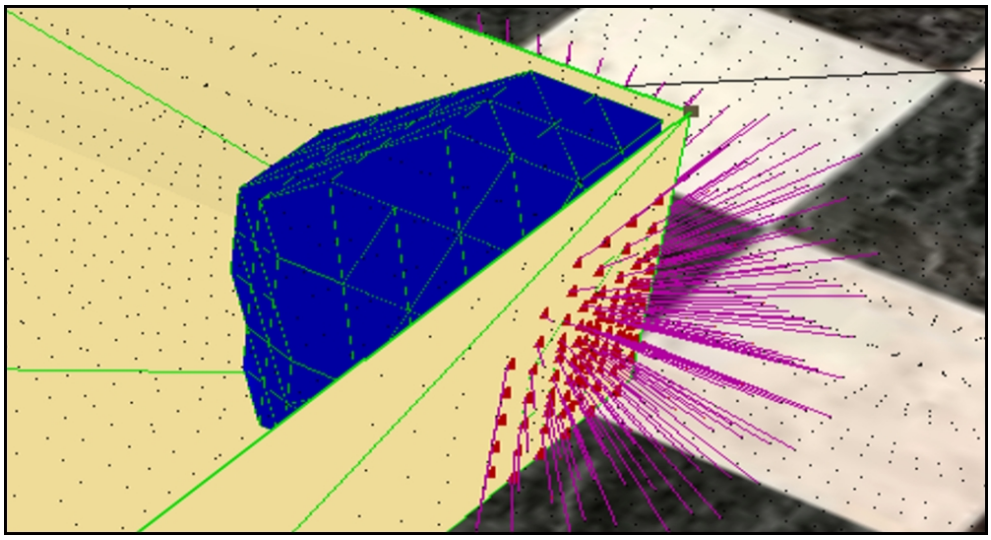

FIG. 7.3: Une particule est située dans le coin d'un boite aux bordures sans épaisseur. La sphère bleue représente l'isosurface grossièrement triangulée autour de la particule. La projection des sommets des triangles de la surface extraite est schématisée par les lignes violettes.

7.4 Résultats

Des captures d'écran d'animations avec la procédure de visibilité désactivée et activée figurent respectivement dans les images 7.4 et 7.5. Dans cette expérimentation, une particule, soumise à la force de gravité, entre en contact avec le sol. La sphère bleue représente l'isosurface grossièrement triangulée autour de la particule. Lorsque l'algorithme de visibilité est désactivé, la sphère passe au travers du sol. Inversement, lorsque l'algorithme de visibilité est activé, une partie de la sphère est projetée sur le sol, ce qui est plus réaliste. Il est important de noter que ces exemples sont très schématiques : en pratique pour les fluides, le rayon définissant l'isosurface est petit, peu de voxels par sphère subissent des projections.

Des exemples d'applications sur des fluides figurent dans les images 7.6, 7.7, 7.8 et 7.9. Dans ces animations, un liquide visqueux interagit avec une membrane. Des captures d'écran provenant d'une caméra située derrière l'objet montre l'efficacité de l'algorithme de visibilité, qui empêche la surface extraite de traverser la membrane.

7.5 Conclusion

Ce chapitre aborde le processus d'extraction de surface de fluide à proximité de membranes. Deux méthodes sont proposées :

- La première est une application directe de la structure accélératrice proposée dans le chapitre 5, elle consiste à optimiser les appels à la fonction d'évaluation du champ scalaire. L'utilisation de cette structure accélère le processus d'extraction de surface. Cette accélération est très dépendante de la taille de la table de hachage : plus la taille de la table hiérarchique est grande, plus les temps de calculs sont écourtés, occupant en contre partie d'avantages de ressources mémoires. Pour bien évaluer ce modèle, il serait intéressant d'établir des comparaisons de temps de calculs avec d'autres structures accélératrices utilisées pour la visualisation, comme les kd-trees par exemple qui, faute de temps, ne figurent pas dans cette thèse. Par ailleurs, il serait plus judicieux de réduire le nombre des requêtes de voisinage en suivant la surface du fluide plutôt que

d'échantillonner le volume de la boite englobante ;

- La deuxième méthode propose un algorithme qui empêche les triangles, issus de l'extraction de surface, de passer au travers des objets en contact avec les particules. Malheureusement, cette méthode ne conserve pas le volume. Pour pallier ce problème, il serait intéressant de faire la somme des valeurs scalaires des échantillons corrigés, puis de la distribuer aux échantillons restants. Le choix de la cette distribution est bien sûr crucial, pour des gouttes d'eau une distribution uniforme n'est pas forcément le modèle le plus judicieux.

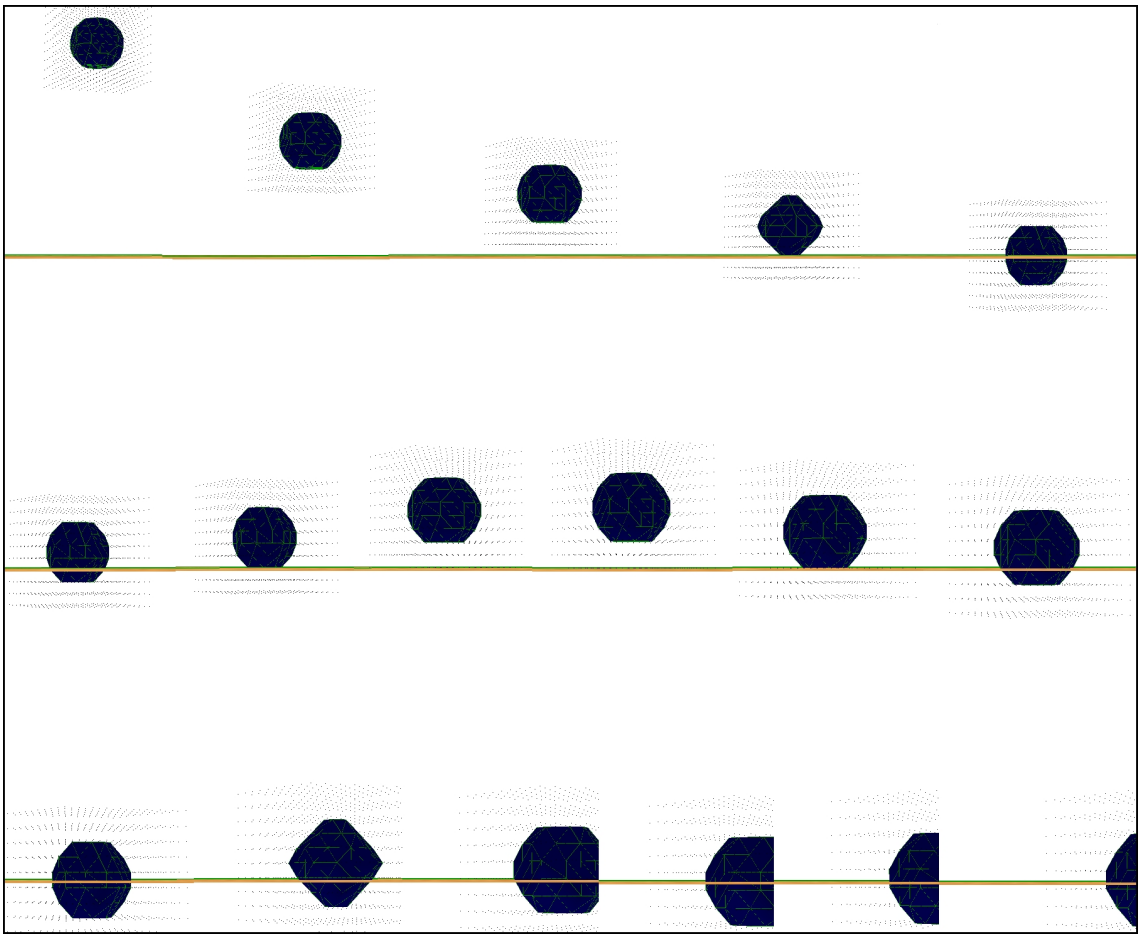

FIG. 7.4: Une particule, soumise à la force de gravité, entre en interaction avec le sol (orange). La sphère bleue représente l'isosurface grossièrement triangulée autour de cette particule. L'algorithme de visibilité n'est pas activé et par conséquent, les triangles de la sphère passent au travers du sol.

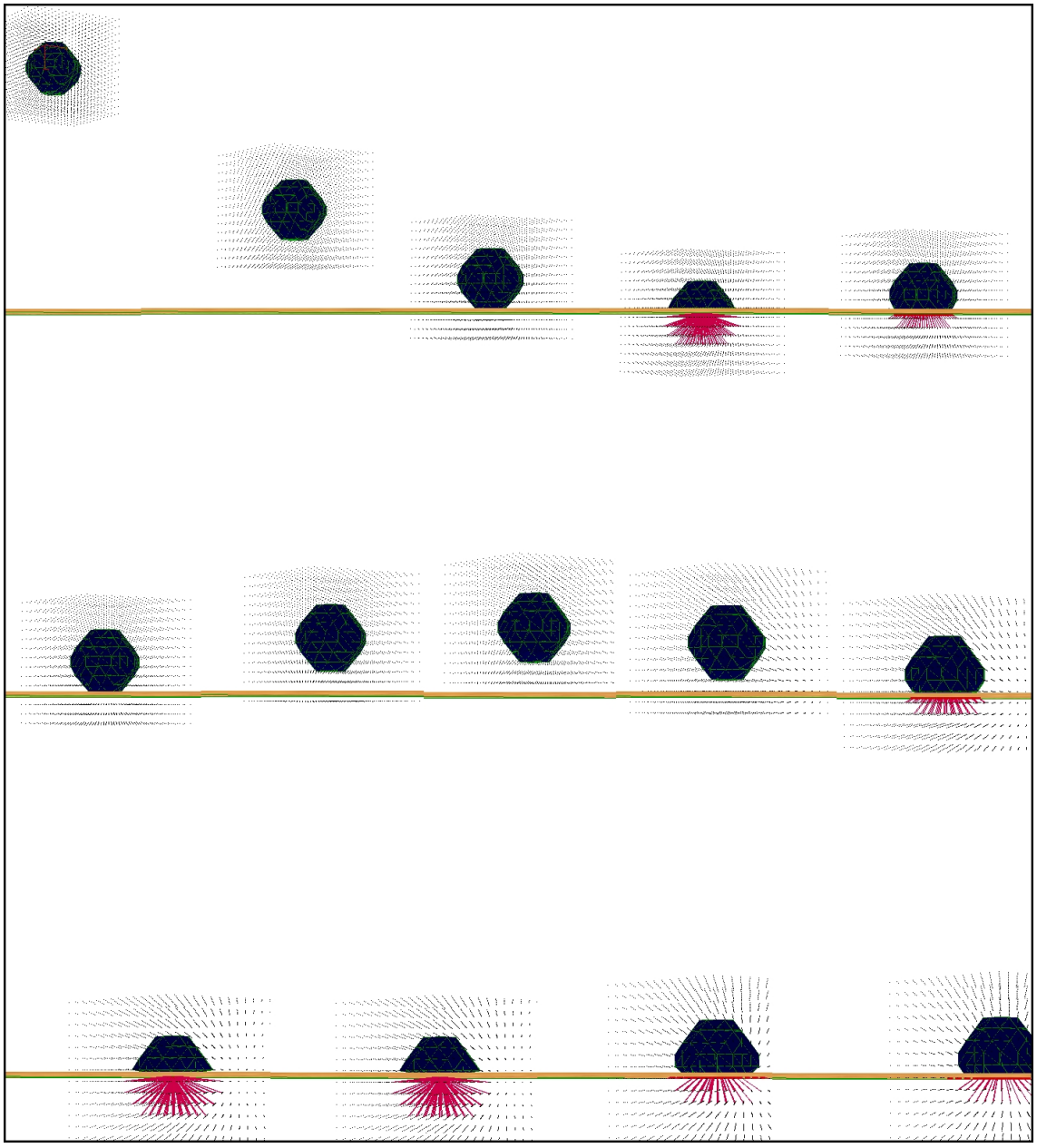

FIG. 7.5: Une particule, soumise à la force de gravité, entre en interaction avec le sol (orange). La sphère bleue représente l'isosurface grossièrement triangulée autour de cette particule. L'algorithme de visibilité est activé. Les lignes violettes indiquent les projections effectuées pour corriger la surface.

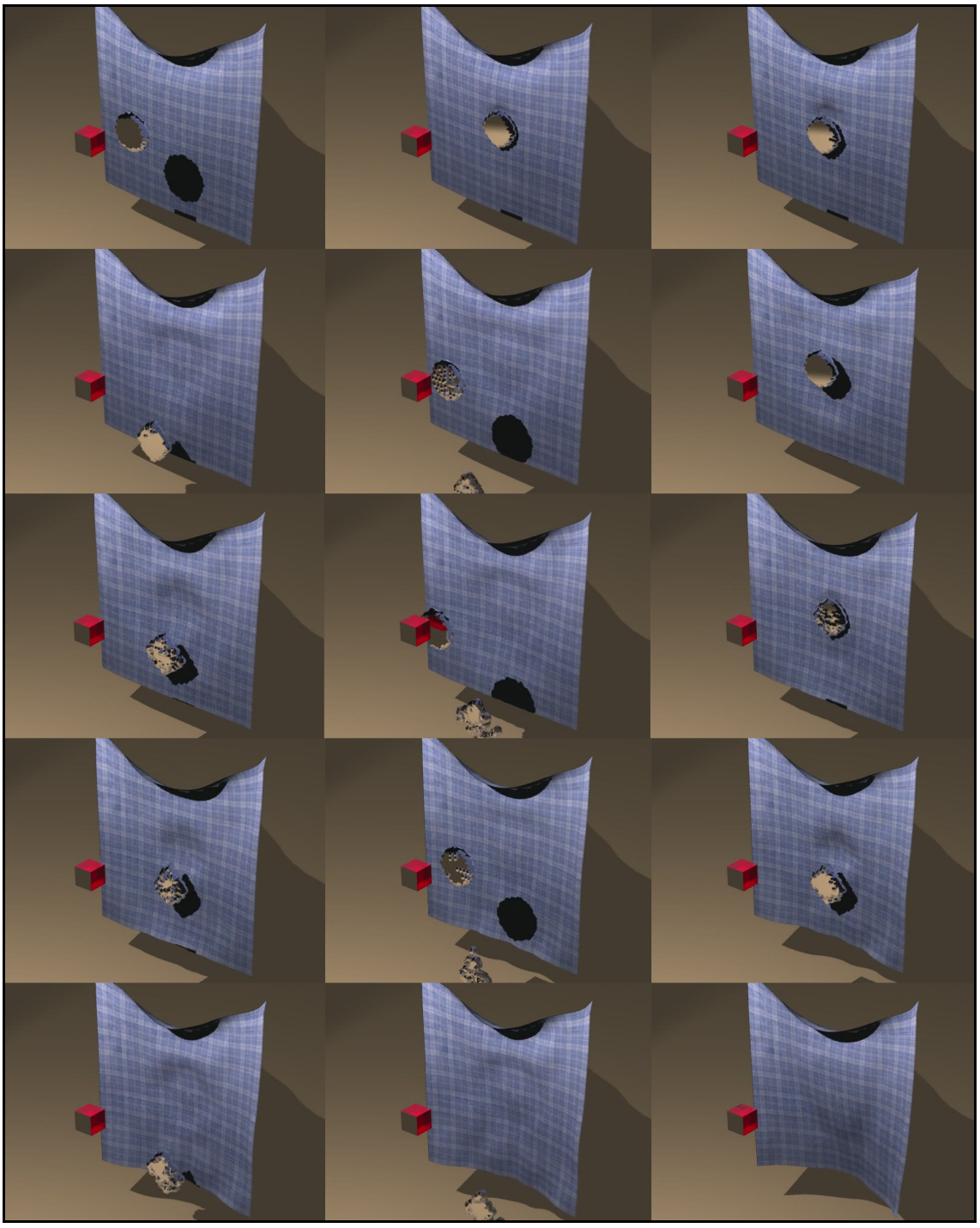

FIG. 7.6: Des “tranches” de fluides interagissent avec un tissu sans épaisseur, vu de devant

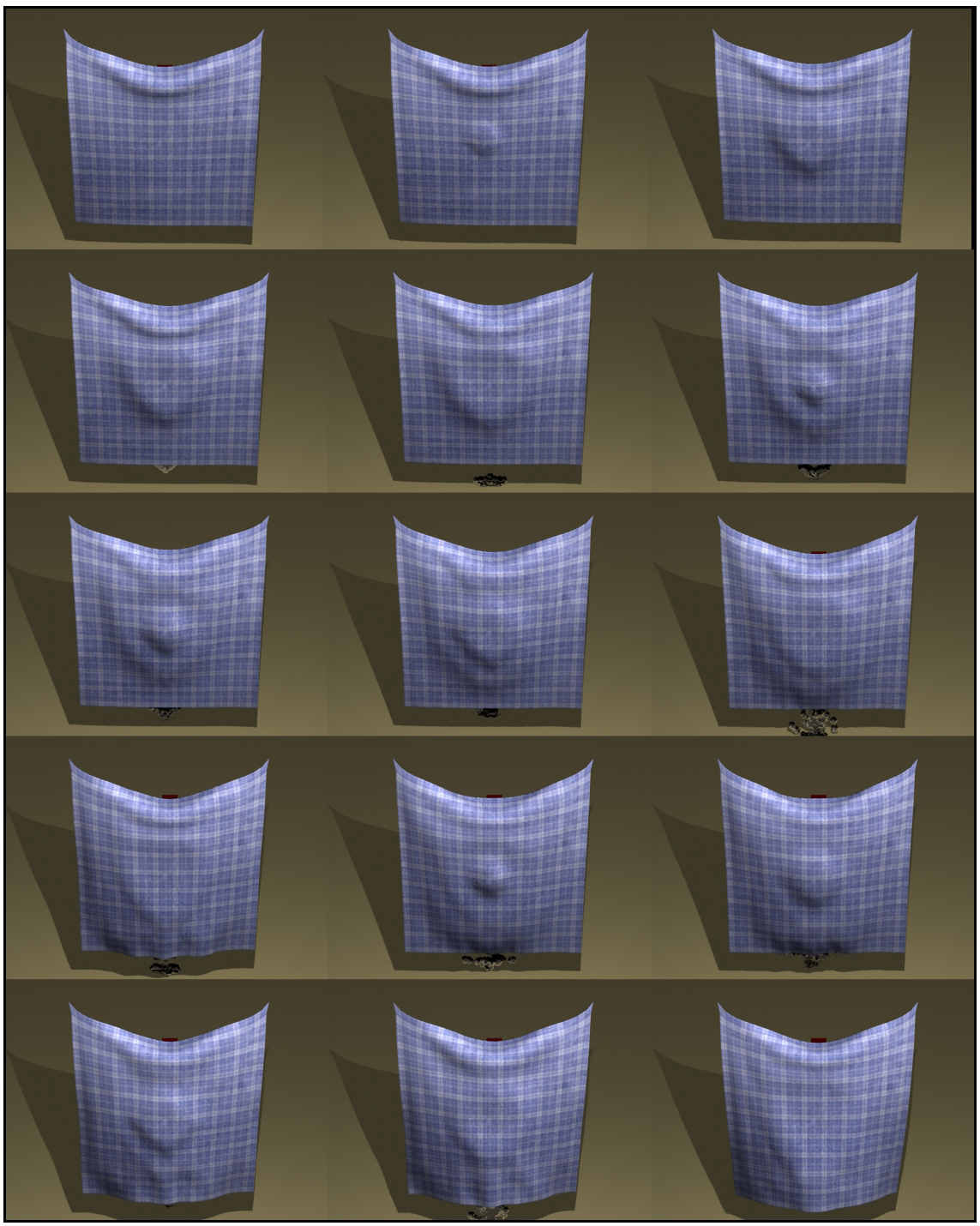

FIG. 7.7: Des “tranches” de fluides interagissent avec un tissu sans épaisseur, vu de derrière. La surface extraite autour des particules de fluides ne traverse pas la membrane, grâce à l’algorithme de visibilité.

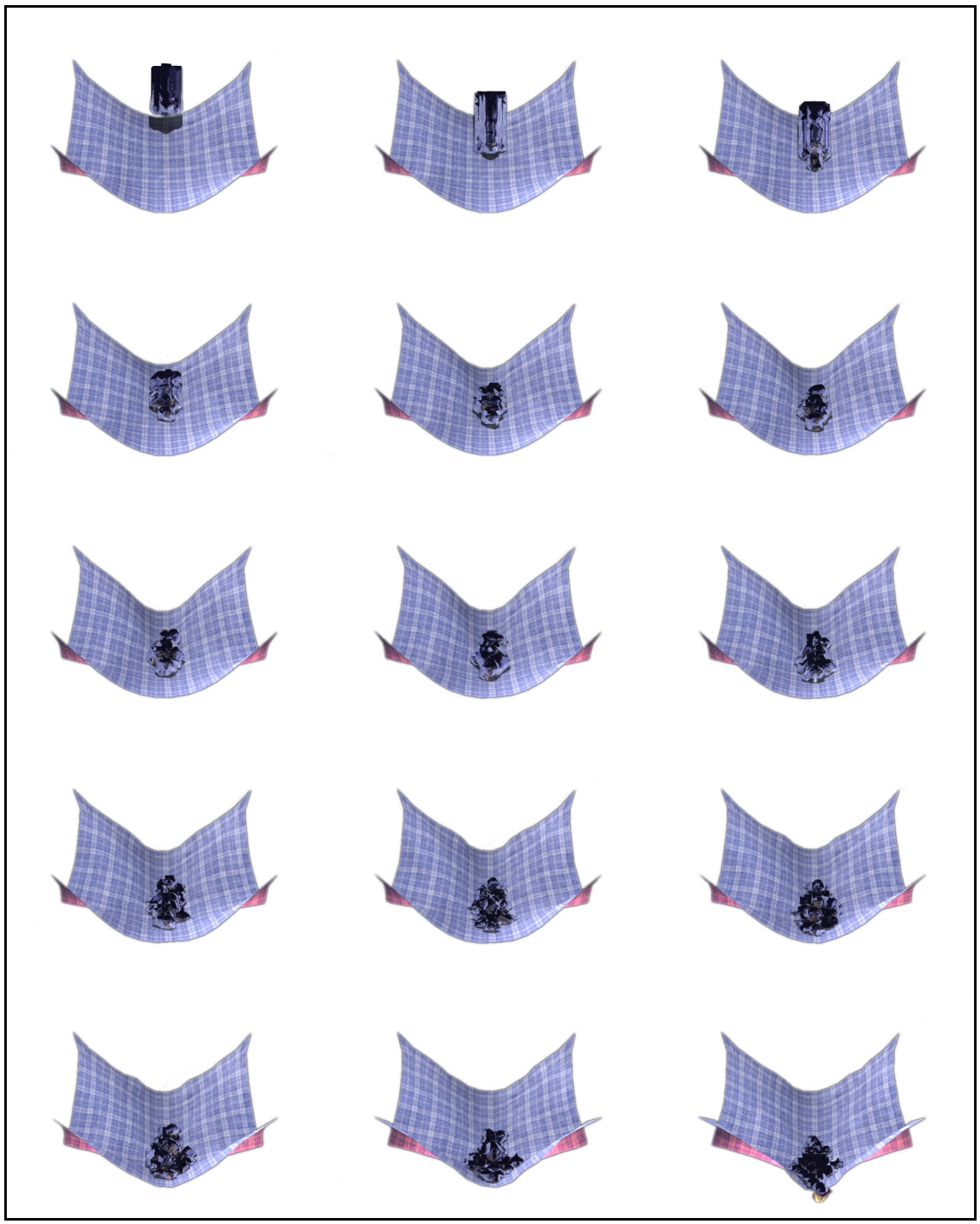

FIG. 7.8: Un liquide est versé sur un mouchoir attaché à ses quatre extrémités, vu de devant

.

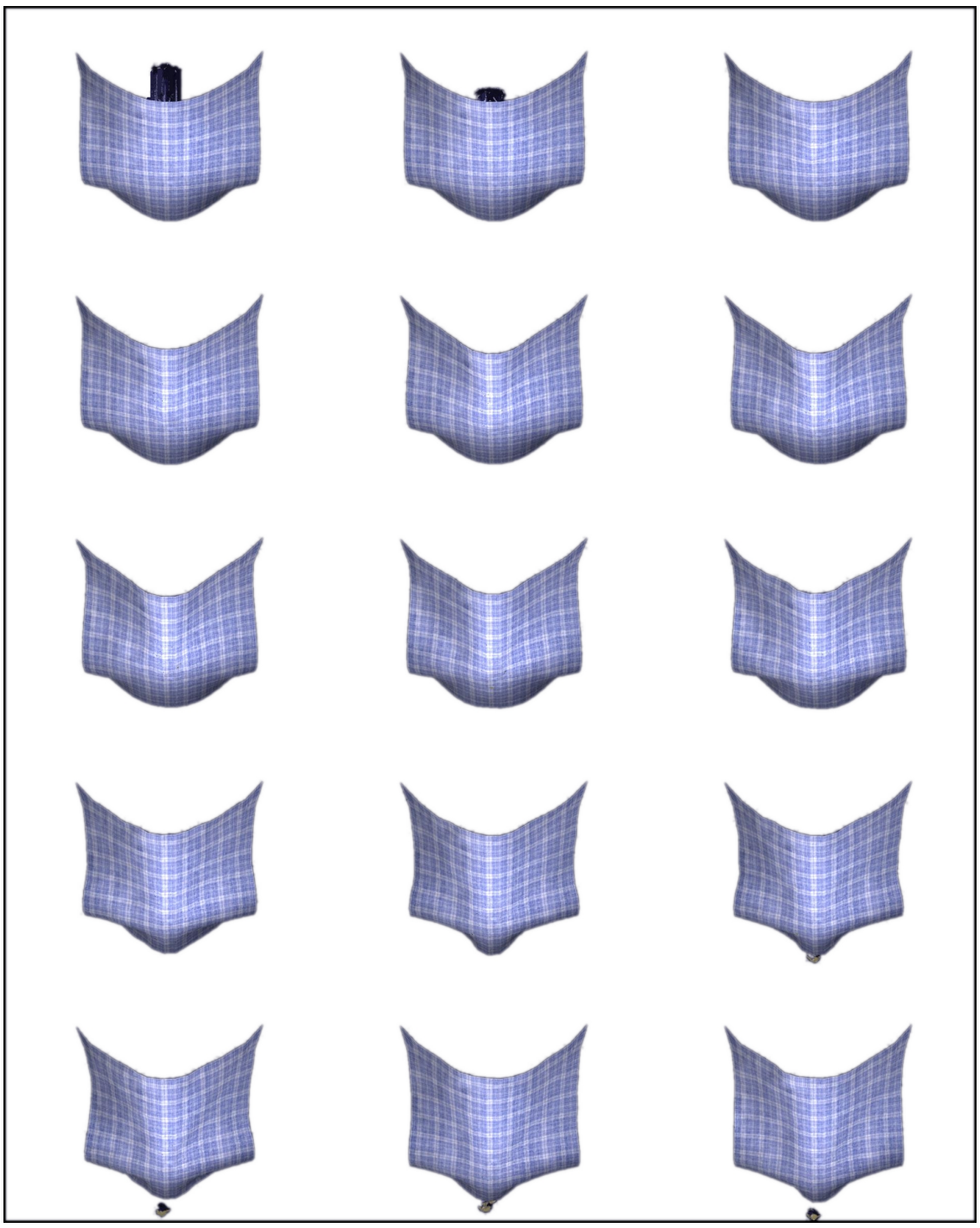

FIG. 7.9: Un liquide est versé sur un mouchoir attaché à ses quatre extrémités, vu de derrière. La surface extraite autour des particules de fluides ne traverse pas la membrane, grâce à l'algorithme de visibilité.

Chapitre Huit

Conclusion

L'introduction de la physique dans les images de synthèse pour dicter le mouvement d'objets permet d'assister la création d'animations complexes, pour des artistes par exemple dans un cadre cinématographique, mais aussi d'élaborer des mondes virtuels autonomes, comme dans les jeux vidéo et les simulateurs en tous genres.

Grâce à la puissance grandissante des ordinateurs, des phénomènes de plus en plus sophistiqués sont reproduits, aussi bien dans un cadre temps-réel que dans des simulations hors ligne. Les équations qui régissent les mouvements sont généralement complexes et leur résolution engendre de longs temps de calculs. Ces temps de calculs ont un enjeu économique important et, pour les réduire, la communauté en synthèse d'images s'autorise à simplifier les modèles tant que le résultat visuel n'est pas trop altéré. Ce compromis entre le visuellement réaliste et les temps de calculs est un sujet capital, spécialement dans les applications interactives.

Deux aspects se distinguent par conséquent dans les animations basées-physique, l'introduction de nouveaux phénomènes et l'optimisation des modèles existants.

Je me suis intéressé à ces deux aspects. J'ai cherché, dans un premier temps, à reproduire l'interaction entre un fluide et une membrane déformable ce qui, à l'époque, était un phénomène qui n'avait pas été explicitement abordé à ma connaissance, en informatique graphique. Pour ce faire, un fluide Lagrangien a été couplé avec une membrane Lagrangienne. J'ai choisi d'utiliser le même modèle géométrique afin de simplifier, dès le départ, les calculs d'interactions. Puisque la réaction de collisions d'un fluide avec des membranes engendre des calculs complexes, j'ai approximé cette réaction de contact en considérant un mouvement linéaire pendant la totalité du pas d'intégration, mais aussi en utilisant des tampons de forces par sommet des triangles

des membranes pour traiter les impacts multiples. Bien qu'à priori l'interaction soit indépendante du pas d'intégration, lorsque celui-ci est trop grand, le mouvement est erroné, dû à l'accumulation des approximations. De plus dans le modèle proposé, la pression du fluide n'est pas prise en compte ce qui engendre une concentration importante des particules le long de la membrane. J'envisage donc dans des travaux futurs de corriger ces problèmes et également d'améliorer la projection des particules de fluides sur la surface de la membrane.

Un fluide Lagrangien n'a pas de surface explicite et donc ne peut être visualisé directement. Pour le visualiser, j'ai dans un deuxième temps utilisé l'algorithme des *marching cubes*, qui est un procédé renommé, pour trianguler la surface implicite définie autour des particules en bordure. Cependant, des artefacts visuels temporaires apparaissent lorsque le fluide est en contact avec la membrane, c'est-à-dire que des triangles traversent cette membrane. Pour pallier ce problème, j'ai présenté un algorithme dit de visibilité qui projette les portions de surface erronées le long de la membrane. Ce procédé a cependant l'inconvénient de ne pas conserver le volume de la surface implicite et je propose dans des travaux futurs de distribuer la quantité de volume projeté au volume du fluide.

Je me suis également intéressé à l'aspect d'optimisation des modèles existants. Dans un troisième temps, j'ai cherché à stabiliser les méthodes d'intégration numériques explicites, qui ont l'avantage de la simplicité. Pour ce faire, je propose un filtrage exponentiel des vitesses qui permet d'augmenter le pas d'intégration temporelle et par conséquent de réduire les temps de calculs. Entre outre, j'ai essayé de stabiliser les méthodes d'intégration en utilisant un θ-schéma et un filtrage de Kalman, mais je n'ai pas obtenu de résultats concluants avec ces deux dernières méthodes. Cependant, le filtrage de Kalman permettrait certainement de stabiliser sous une condition moins forte les méthodes d'intégration explicite et j'envisage d'approfondir cette étude.

Toujours en terme d'optimisation, j'ai proposé dans un quatrième temps, une structure de subdivision spatiale dynamique pour accélérer la détection du voisinage d'une particule et d'un triangle. Cette structure permet de réduire les coûts de détermination du voisinage d'une particule pour le calcul de la dynamique du fluide, pour la détection de collisions et pour l'extraction de surface. Cette approche a cependant

l'inconvénient d'être gourmande en terme de ressources mémoires. Pour pallier cet inconvénient, je propose dans les travaux futurs de mettre en oeuvre une fonction de hachage "parfaite" ce qui pourrait minimiser l'occupation mémoire. J'ai également présenté une méthode simple mais efficace pour paramétrer intuitivement des coefficients hétérogènes de frictions sur une même surface, par le biais de cartes.

Plusieurs travaux ont été publiés ces trois dernières années concernant l'interaction entre un fluide et une membrane. Cependant aucun ne propose de solution robuste, indépendante du pas d'intégration temporelle avec un fluide Lagrangien. D'un point de vue général, je suis assez sceptique quand à la pérennité de l'usage de triangles pour définir la surface d'un fluide. En effet, aujourd'hui il y a généralement plus d'éléments dans une scène tridimensionnelle que de pixels à afficher, ce qui constitue l'inverse du paradigme initial en synthèse d'images. Il est par conséquent plus judicieux d'effectuer plus de traitements sur les pixels que sur les éléments afin de réduire les temps de calculs. Je préconise donc l'utilisation d'un affichage basé-point pour représenter les surfaces d'un fluide Lagrangien. Cette technique offre déjà des résultats très réalistes et en un temps plus qu'honorable.

Je suis assez sceptique quand à l'utilisation de méthodes d'intégration explicite avec des systèmes dit raides. Cependant, le filtrage représente un pas vers une stabilisation inconditionnelle, mais à quel coût celle-ci sera t-elle obtenue ?

Par contre, je suis très optimiste quand à l'utilisation de structures de subdivision spatiale hiérarchique. Puisque la détermination du voisinage est le point faible des fluides Lagrangiens et qu'ils sont en plein essor, je pense que ces structures accélératrices vont solliciter beaucoup d'attentions. Je préconise d'ailleurs dans ce cadre la mise en oeuvre d'un *neighborhood shader* avec du matériel dédié : une détermination du voisinage matérielle permettrait d'accélérer grandement l'animation et le rendu de structures Lagrangiennes sans maillage, mais également d'autres procédés tels que la détection de collisions ou encore le rendu sous-surfacique par exemple.

Chapitre Neuf

Annexes

9.1 Smoothed Particles Hydrodynamics

Les *Smoothed Particles Hydrodynamics* (SPH) est un modèle Lagrangien sans maillage qui permet de reproduire la dynamique de fluides. Cette méthode a été initialement développée pour simuler des problèmes astrophysiques tels que la fission des étoiles par Leon Lucy [64] et Robert Gingold et Joe Monaghan [38]. Une quantité physique, ainsi que sa dérivée spatiale, sont approximées par une fonction d'interpolation dans un proche voisinage. Par exemple, les forces sont calculées à l'aide de cette fonction. De plus, puisque cette méthode est un modèle Lagrangien, la conservation de la masse est assurée. Cette simplification de la résolution des équations de la dynamique des fluides permet notamment d'utiliser les SPH en temps interactifs.

En prenant une fonction continue, $A(\mathbf{x})$ définie sur le domaine Ω, l'interpolation de l'intégrale $\langle A(\mathbf{x})\rangle$ est calculée par un noyau de lissage :

$$\langle A(\mathbf{x})\rangle = A * \omega = \int A(\mathbf{x}')\omega(\mathbf{r}, h)d\mathbf{x}' \tag{9.1}$$

où $\mathbf{r} = \mathbf{x} - \mathbf{x}'$, h est le noyau de lissage et $d\mathbf{x}'$ est un élément de volume différentiable. Ce noyau est normalisé, c'est-à-dire :

$$\int \omega(\mathbf{r}, h)d\mathbf{x}' = 1 \tag{9.2}$$

L'idée des SPH est de représenter une fonction continue par un échantillonage de Monte Carlo de particules. Chacune de ces particules représente une partie du volume global de la matière. D'un point de vue mathématique, ces particules sont des

points d'interpolations qui permettent d'évaluer certaines quantités. Une approximation discrète de la fonction $A(\mathbf{x})$ consiste à remplacer l'élément différentiable $d\mathbf{x}'$ par le volume V_j de la particule p_j de position $\mathbf{x}_j$:

$$\langle A(\mathbf{x})\rangle \simeq \sum_j A(\mathbf{x_j})\omega(\mathbf{x}-\mathbf{x_j},h)V_j \tag{9.3}$$

Sachant que $V_j = m_j/\rho_j$, le calcul de la densité par exemple est :

$$\begin{aligned}\rho(\mathbf{x}) &= \textstyle\sum_j \rho_j \frac{m_j}{\rho_j}\omega(\mathbf{x}-\mathbf{x_j},h)\\ &= \textstyle\sum_j m_j\omega(\mathbf{x}-\mathbf{x_j},h)\end{aligned} \tag{9.4}$$

ou encore la pression :

$$\nabla p(\mathbf{x}) = \sum_j m_j \frac{p_j}{\rho_j}\nabla\omega(\mathbf{x}-\mathbf{x}_j,h) \tag{9.5}$$

Ces équations reposent principalement sur les noyaux de lissage, qui sont en synthèse d'images de type Gaussien, B-spline ou Q-spline. Ci-dessous figure un exemple de noyau proposé par Matthias Müller *et al.* [75] :

$$\omega(\mathbf{r},h) = \frac{315}{64\pi h^9}\begin{cases}(h^2-r^2)^3 & 0<r<h\\ 0 & sinon\end{cases} \tag{9.6}$$

9.2 Systèmes masses-ressorts

Les systèmes masses-ressorts sont des systèmes de particules contraintes dans l'espace par des ressorts. Un ressort est caractérisé par des forces d'attraction et de répulsion. Ainsi, il contraint les deux particules connectées à ses extrémités à conserver une distance prédéfinie dénommée l, c'est-à-dire la longueur initiale du ressort. La notion de force induit la connaissance d'autres quantités physique telles que la position, la vitesse, l'accélération et la masse. Ces quantités sont véhiculées par les particules.

La force d'interaction entre deux particules i et j connectées par un ressort est

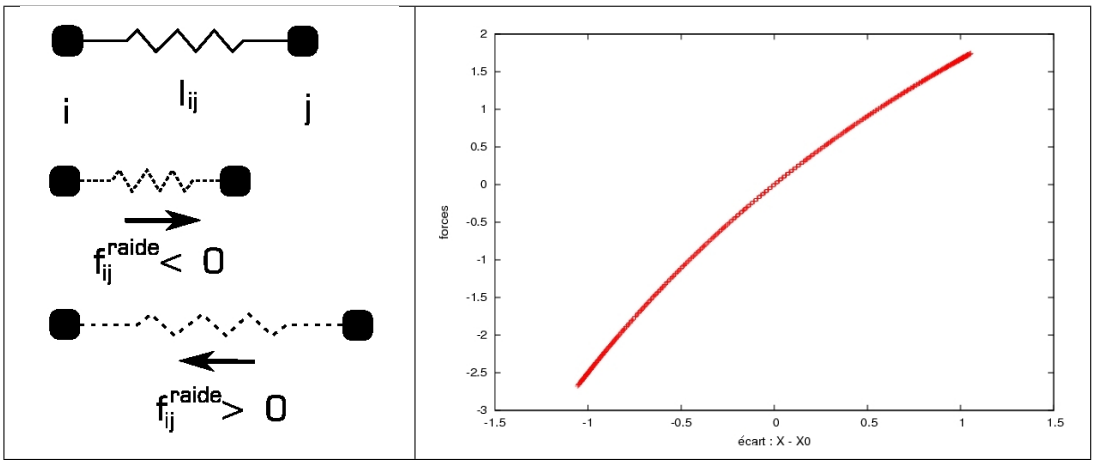

FIG. 9.1: *À gauche :* Une ressort est comprimé puis étiré. *À droite :* la distance séparant les deux particules est borné entre -1 et 1, le ressort oscille à l'infini car il n'y a pas d'amortissement.

calculée par la formule suivante :

$$\mathbf{f}_{ij}^{raide} = k\left(\|\mathbf{x}_{ji}\| - l_{ij}\right)\frac{\mathbf{x}_{ji}}{\|\mathbf{x}_{ji}\|} \tag{9.7}$$

où $\mathbf{f}_{ij}^{raide}$ est la force d'attraction ou de répulsion entre les particules i et j appliquée sur la particule i, $\mathbf{x}_{ji} = \mathbf{x}_j - \mathbf{x}_i$ avec $\mathbf{x}_j$, $\mathbf{x}_i$ les positions respectives des particules j et i et l_{ji} est la longueur initiale du ressort. Le scalaire $k \in \Re^+$ est le paramètre de raideur, coefficient multiplicateur des forces. Plus k va être important, plus les forces de répulsions et d'attractions seront grandes et donc plus les particules seront contraintes dans leurs déplacements. Le comportement de l'objet simulé aura par conséquent un aspect rigide, voir solide. Inversement, une petite valeur de k engendrera des comportements d'objets mous très élastiques. Les figures 9.1 et 9.2 illustrent ces propos.

La première loi de mouvement de Newton[1] est énoncée comme suit :

Tout corps persévère dans l'état de repos ou de mouvement uniforme en ligne droite dans lequel il se trouve, à moins que quelque force n'agisse sur lui, et ne le

[1]Principes mathématiques de la philosophie naturelle. D'après la traduction du latin en français par Émilie du Chatelet (1756).

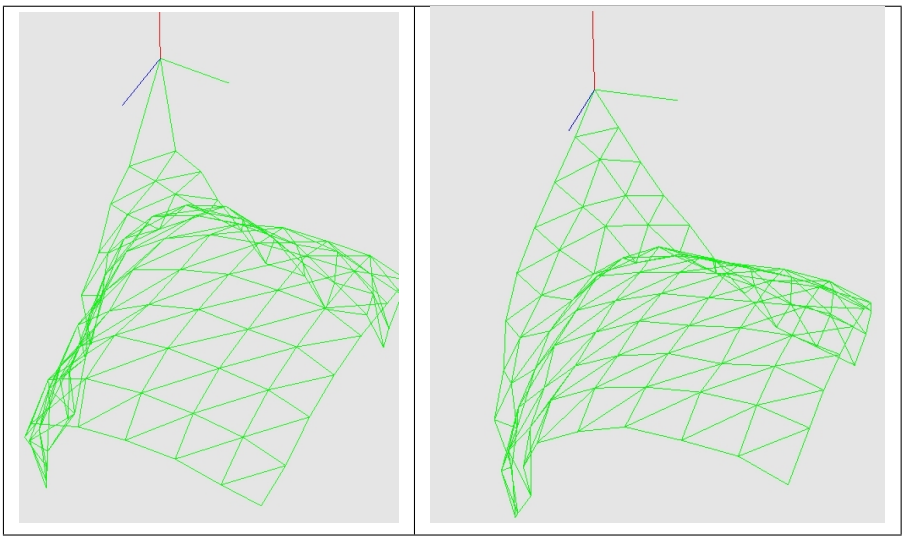

FIG. 9.2: Le coin étiré est dû à une faible rigidité dans l'image de gauche, contrairement à l'image de droite où une forte rigidité force l'objet à conserver son aspect initial.

contraigne à changer d'état.

Les forces de répulsions et d'attractions engendrées par l'équation 9.7 vont se succéder continuellement car il n'existe pas de terme de friction. Cette phase d'oscillation doit être "amortie", donc un terme d'amortissement est ajouté :

$$\mathbf{f}_{ij}^{amort} = d(\mathbf{v}_j - \mathbf{v}_i) \tag{9.8}$$

où $\mathbf{v}_j$, $\mathbf{v}_i$ sont respectivement les vitesses des particules j et i et $d \in [0 \dots 1]$ est un coefficient d'amortissement. Cette formulation n'est pas correcte car elle agit non seulement dans l'axe du ressort (dans le *in-plane*), mais également dans d'autres axes (*out-plane*) et engendre par conséquent des déplacements erronés. Malgré cela, ce modèle est très employé. Des informations supplémentaires sur ce sujet figurent dans la section 4.4.

Il existe d'autres formulations, comme celle dite de l'amortissement du point :

$$\mathbf{f}_{i}^{amort} = d \cdot \mathbf{v}_i \tag{9.9}$$

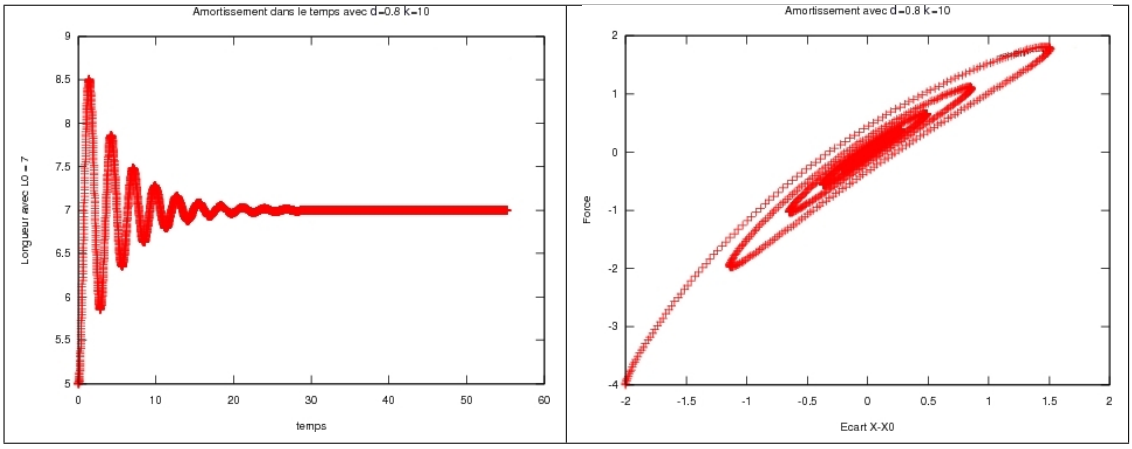

FIG. 9.3: *À gauche :* le ressort stabilise sa taille à sa longueur de repos en 30 ms. *à droite :* La force d'attraction/répulsion est nulle lorsque la distance séparant les particules est égale à la longueur de repos du ressort.

Cependant, cet amortissement engendre un ralentissement global du mouvement. Cette équation peut simuler une forte friction avec l'environnement comme par exemple un objet plongé dans de l'eau ou de l'huile. Mais ce n'est pas le modèle adapté pour amortir les oscillations d'un ressort. Le modèle le plus correct est nommé l'amortissement projeté, qui n'agit **que** dans l'axe du ressort (voir aussi le schéma 9.3) :

$$\begin{aligned} \mathbf{f}_{ij}^{damp} &= d\dot{\mathbf{r}}_{ij}\frac{\mathbf{x}_{ji}}{\|\mathbf{x}_{ji}\|} \\ \dot{\mathbf{r}}_{ij} &= \frac{\mathbf{v}_{ji}\cdot\mathbf{x}_{ji}}{\|\mathbf{x}_{ji}\|} \end{aligned} \tag{9.10}$$

avec $\mathbf{v}_{ji} = \mathbf{v}_j - \mathbf{v}_i$.

Les forces d'attractions/répulsions et d'amortissements constituent les forces dites *internes* du système auquelles s'ajoute des forces dites *externes* comme la gravité, les interactions utilisateurs, la friction avec le fluide environnant et les collisions avec les objets présents dans la scène (voir le chapitre 6 pour plus de détails).

Connaissant la somme des forces appliquées sur une particule, les méthodes d'intégrations numériques permettent d'approximer le déplacement induit : la nouvelle vitesse et la nouvelle position. Ces méthodes sont détaillées dans la section 2.3 et dans le chapitre 4.

9.3 Mise en oeuvre

Cette section discute de l'implantation des méthodes proposées dans les chapitres précédents. J'ai travaillé essentiellement avec l'éditeur de code VIM, sous le système d'exploitation Linux, en programmant avec le langage C++.

9.3.1 Greffon pour le logiciel Maya

Dans un premier temps, j'ai programmé un greffon pour le logiciel Maya. Ce choix s'est fait naturellement puisque ce logiciel possède une riche panoplie de fonctionnalités. Je n'avais qu'à programmer le strict minimum et à utiliser l'existant, ceci permettant de ne pas implanter des "détails" non valorisables dans un cadre de recherche.

Ma première tâche fut d'implanter un modeleur volumique issu du laboratoire SIR, à partir duquel je devais effectuer mes recherches, en tant que greffon pour Maya. Un résultat graphique est illustrée dans la figure 9.4. Les poissons sont générés par le modeleur volumique. Le plaquage de textures, les fausses caustiques, la caméra et les lumières sont des fonctionnalités de Maya. Le décor sous-marin a été "dessiné" à l'aide de Maya Paint Effects.

Malheureusement, le kit de développement de Maya (*Software Developpment Kit*, SDK) est une surcouche qui se situe au dessus du noyau du modeleur auquel le programmeur n'a pas accès (voir l'ouvrage de David Gould [40] pour plus de détails). Cette phase de communication supplémentaire ralentie les calculs, mais, je dois l'admettre, est efficace. J'ai cependant décidé de laisser de côté ce logiciel et d'en programmer un dédié et optimisé pour les animations basées-physique.

9.3.2 The Tomato Project

The Tomato Project[2] est un logiciel d'animation basées-physique, qui comprend approximativement 20k lignes de codes, et d'après sloccount[3], coûterait $600k et

[2]je reconnais que ce nom n'est pas très explicite, mais il vaut un *physicator*, un yapbm (*Yet Another Physical Based Modeler*) ou bien un a.out.

[3]http ://www.dwheeler.com/sloccount/

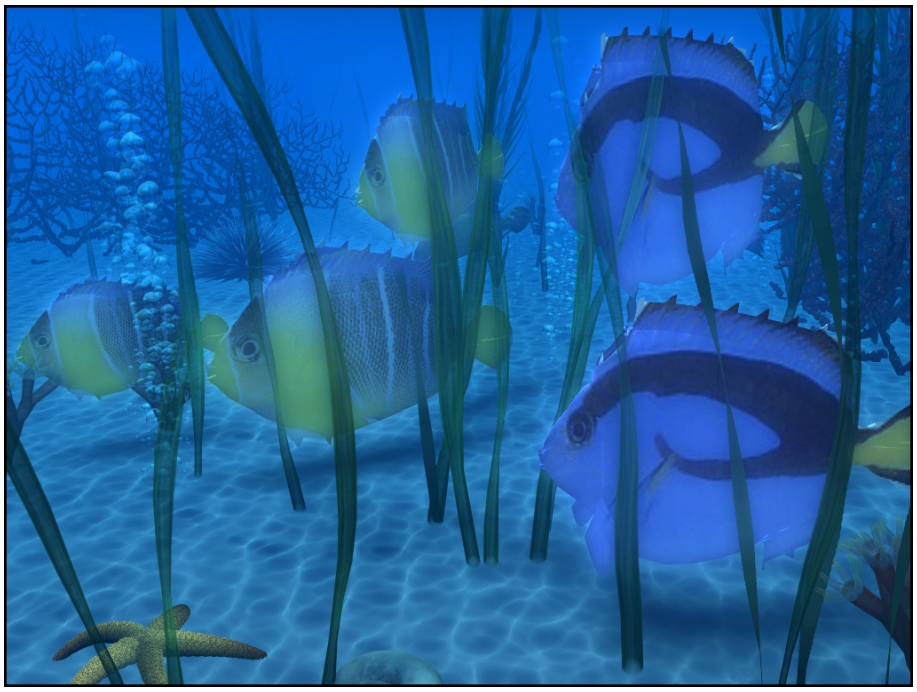

FIG. 9.4: Exemple de modélisation de poissons avec le greffon volumique intégré au logiciel Maya.

nécessiterait 4,81 personnes pendant un an pour l'élaborer... Ce logiciel est muni d'une interface graphique programmée avec la librairie QT[4], qui sert entre autres à paramétrer les différents modèles masses-ressorts, les SPH, les tables de hachage, les méthodes d'intégration numériques et l'affichage. La figure 9.5 montre une capture d'écran de cette interface.

Un analyseur lexical, Flex, et syntaxique, Bison, permettent de charger les scènes et les objets. Ce logiciel dispose d'une prévisualisation OpenGL avec des textures, des lumières et de la transparence. Par ailleurs, l'utilisateur peut interagir avec les objets en sélectionnant un sommet par le biais du *picking*, puis en le déplaçant. Ce déplacement est réalisé via des forces qui sont ajoutées au sommet, puis intégrées lors du prochain pas d'intégration temporelle.

Entre outre, pour calculer les animations en version finale, j'ai ajouté un conver-

[4]http ://fr.wikipedia.org/wiki/Qt

tisseur RIB, le format de fichier du logiciel de rendu Renderman[5]. Le logiciel Pixie[6] a été utilisé pour rendre les images/animations. Pour les fluides, j'ai ajouté la prise en compte des caustiques qui nécessitent deux étapes de rendu et par conséquent deux fichiers RIB quelque peu différents. Des résultats de rendu avec Pixie sont illustrés dans les figures 9.6 et 9.7.

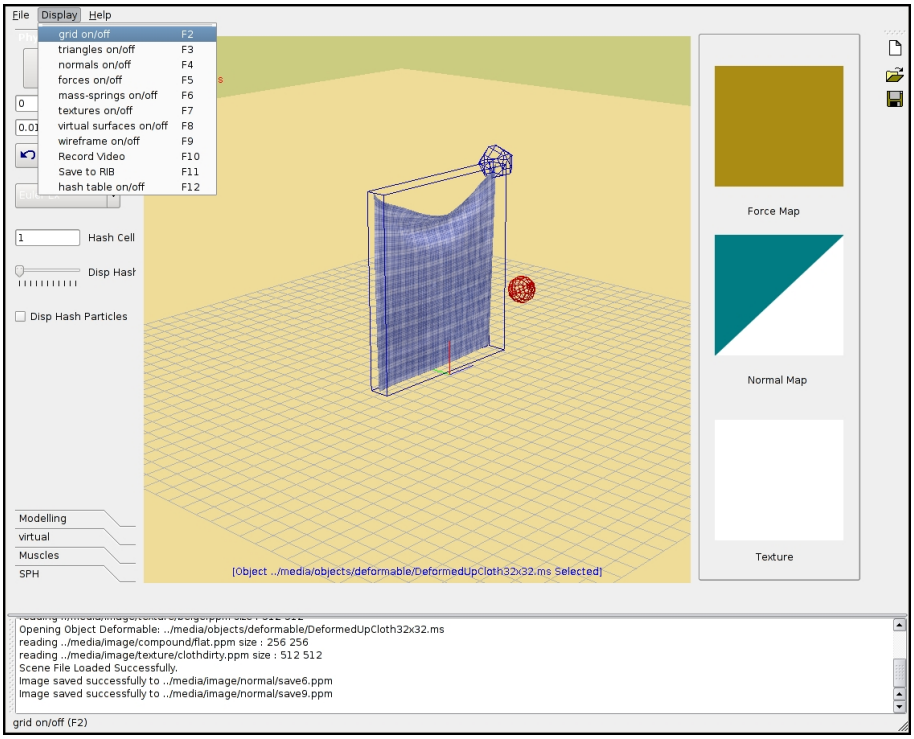

FIG. 9.5: Une capture d'écran de The Tomato Project.

[5]http ://www.renderman.org
[6]http ://pixie.sourceforge.net

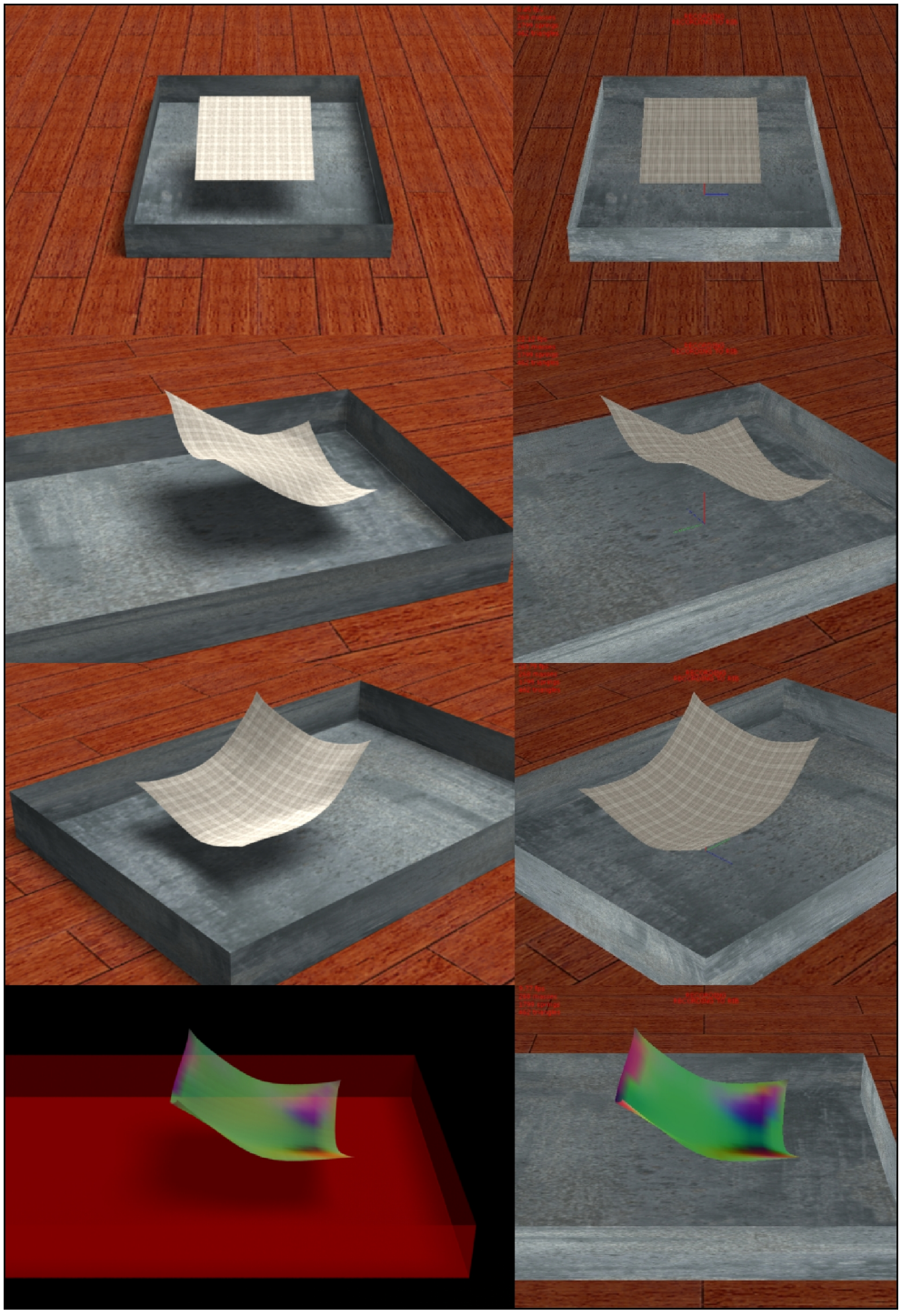

FIG. 9.6: Comparaison de la prévisualisation en temps-réel en utilisant OpenGL (droite) et du rendu *hors ligne* avec le logiciel Pixie (gauche).

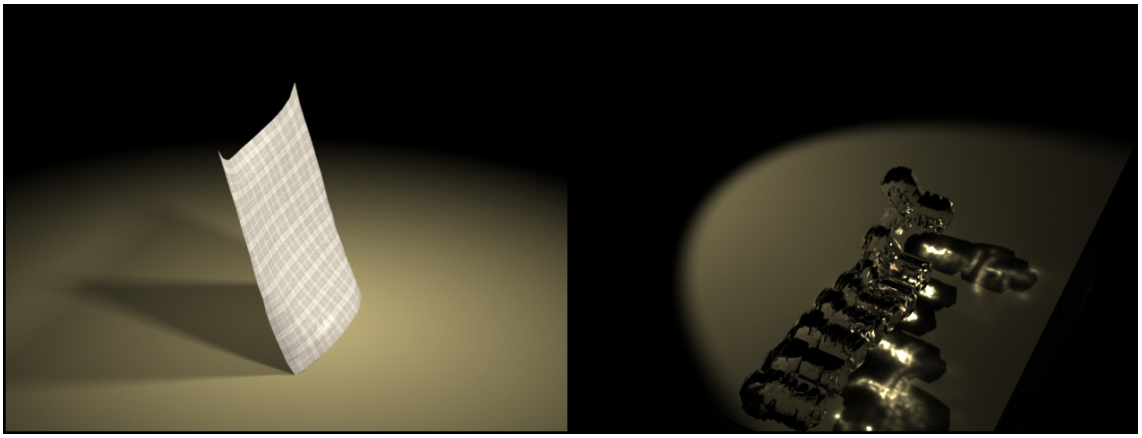

FIG. 9.7: Deux scènes ont été exportées à partir de The Tomato Projet et sont rendues avec le logiciel Pixie.

Bibliographie

[1] Adabala, N., and Manohar, S. Modeling and rendering of gaseous phenomena using particle maps. *Journal of Visualization and Computer Animation 11*, 5 (2000), 279–293.

[2] Adabala, N., and Manohar, S. Techniques for realistic visualization of fluids : A survey. *Computer Graphics Forum 21*, 1 (2002), 65–65.

[3] Adams, B., Pauly, M., Keiser, R., and Guibas, L. J. Adaptively sampled particle fluids. In *SIGGRAPH '07* (New York, NY, USA, 2007), ACM Press (In Press).

[4] Ahmad, A., Adly, S., Terraz, O., and Ghazanfarpour, D. Stability analysis of filtered mass-spring systems. In *Theory and Practice of Computer Graphics* (2007), Eurographics Association, pp. 45–52.

[5] Alexandrova, T., Terraz, O., and Ghazanfarpour, D. Interaction between water and dynamic soft bodies. In *GRAPP* (2006), pp. 384–391.

[6] Baraff, D., and Witkin, A. Large steps in cloth simulation. In *SIGGRAPH '98 : Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), ACM Press, pp. 43–54.

[7] Bargteil, A. W., Goktekin, T. G., O'Brien, J. F., and Strain, J. A. A semi-lagrangian contouring method for fluid simulation. *ACM Transactions on Graphics 25*, 1 (2006).

[8] Bart Adams, T. L., and Dutré, P. Particle splatting : Interactive rendering of particle-based simulation data. Tech. Rep. Technical Report CW 453, CGL ETH Zurich, July 2006.

[9] Boxerman, E. Speeding up cloth simulation. Master's thesis, University of British Columbia, 2003.

[10] Boxerman, E., and Ascher, U. Decomposing cloth. In *SCA '04 : Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2004), ACM Press, pp. 153–161.

[11] Brackbill, J. U., and Ruppel, H. M. Flip : A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *J. Comput. Phys. 65*, 2 (1986), 314–343.

[12] Bridson, R., Fedkiw, R., and Anderson, J. Robust treatment of collisions, contact and friction for cloth animation. In *SIGGRAPH '02 : Proceedings of the*

29th annual conference on Computer graphics and interactive techniques (New York, NY, USA, 2002), ACM Press, pp. 594–603.

[13] Bridson, R., Hourihan, J., and Noderstam, M. Curl-noise for procedural fluid flow. In *SIGGRAPH 2007 (To appear)* (New York, NY, USA, 2007), ACM Press.

[14] Bridson, R., Marino, S., and Fedkiw, R. Simulation of clothing with folds and wrinkles. In *SCA '03 : Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation* (Aire-la-Ville, Switzerland, 2003), Eurographics Association, pp. 28–36.

[15] Cabral, B., and Leedom, L. C. Imaging vector fields using line integral convolution. In *SIGGRAPH '93 : Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1993), ACM Press, pp. 263–270.

[16] Carlson, M., Mucha, P. J., R. Brooks Van Horn, I., and Turk, G. Melting and flowing. In *SCA '02 : Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2002), ACM Press, pp. 167–174.

[17] Carlson, M., Mucha, P. J., and Turk, G. Rigid fluid : animating the interplay between rigid bodies and fluid. *ACM Trans. Graph. 23*, 3 (2004), 377–384.

[18] Chen, J. X., and da Vitoria Lobo, N. Toward interactive-rate simulation of fluids with moving obstacles using navier-stokes equations. *Graph. Models Image Process. 57*, 2 (1995), 107–116.

[19] Choi, K.-J., and Ko, H.-S. Stable but responsive cloth. In *SIGGRAPH '02 : Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM Press, pp. 604–611.

[20] Clavet, S., Beaudoin, P., and Poulin, P. Particle-based viscoelastic fluid simulation. In *Symposium on Computer Animation 2005* (July 2005), pp. 219–228.

[21] Desbrun, M., and Gascuel, M.-P. Smoothed particles : A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation '96* (1996), pp. 61–76.

[22] Eberhardt, B., Etzmuss, O., and Hauth, M. Implicit-explicit schemes for fast animation with particle systems. In *Eurographics Computer Animation and Simulation Workshop 2000* (2000).

[23] Ebert, D. S., Musgrave, F. K., Peachey, D., Perlin, K., and Worley, S. *Texturing and Modeling : A Procedural Approach.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[24] Ebert, D. S., and Parent, R. E. Rendering and animation of gaseous phenomena by combining fast volume and scanline a-buffer techniques. In *SIGGRAPH '90 : Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1990), ACM Press, pp. 357–366.

[25] Enright, D., Fedkiw, R., Ferziger, J., and Mitchell, I. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys. 183*, 1 (2002), 83–116.

[26] Enright, D., Marschner, S., and Fedkiw, R. Animation and rendering of complex water surfaces. In *SIGGRAPH '02 : Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM Press, pp. 736–744.

[27] Erleben, K., Sporring, J., Henriksen, K., and Dohlman, K. *Physics-based Animation (Graphics Series)*. Charles River Media, Inc., Rockland, MA, USA, 2005.

[28] Fan, Z., Zhao, Y., Kaufman, A., and He, Y. Adapted unstructured lbm for flow simulation on curved surfaces. In *SCA '05 : Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), ACM Press, pp. 245–254.

[29] Fedkiw, R., Stam, J., and Jensen, H. W. Visual simulation of smoke. In *SIGGRAPH '01 : Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM Press, pp. 15–22.

[30] Feldman, B. E., O'Brien, J. F., and Klingner, B. M. Animating gases with hybrid meshes. *ACM Trans. Graph. 24*, 3 (2005), 904–909.

[31] Feldman, B. E., O'Brien, J. F., Klingner, B. M., and Goktekin, T. G. Fluids in deforming meshes. In *SCA '05 : Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), ACM Press, pp. 255–259.

[32] Fleisher, D. T. . J. P. . A. B. . K. Elastically deformable models. *Computer Graphics 21*, 4 (July 1987), 205–214.

[33] Foster, N., and Fedkiw, R. Practical animation of liquids. In *SIGGRAPH '01 : Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM Press, pp. 23–30.

[34] Foster, N., and Metaxas, D. Realistic animation of liquids. In *GI '96 : Proceedings of the conference on Graphics interface '96* (Toronto, Ont., Canada, Canada, 1996), Canadian Information Processing Society, pp. 204–212.

[35] Foster, N., and Metaxas, D. Modeling the motion of a hot, turbulent gas. In *SIGGRAPH '97 : Proceedings of the 24th annual conference on Com-*

puter graphics and interactive techniques (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 181–188.

[36] Fournier, A., and Reeves, W. T. A simple model of ocean waves. In *SIGGRAPH '86 : Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1986), ACM Press, pp. 75–84.

[37] Génevaux, O., Habibi, A., and Dischler, J.-M. Simulating fluid-solid interaction. In *Graphics Interface* (June 2003), CIPS, Canadian Human-Computer Commnication Society, A K Peters, pp. 31–38. ISBN 1-56881-207-8, ISSN 0713-5424.

[38] Gingold, R., and Monaghan, J. Smoothed particle hydrodynamics : theory and application to non-spherical stars. *mn 181* (1977), 375–389.

[39] Goktekin, T. G., Bargteil, A. W., and O'Brien, J. F. A method for animating viscoelastic fluids. *ACM Trans. Graph. 23*, 3 (2004), 463–468.

[40] Gould, D. *Maya Programming*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[41] Grote, M. J., and Majda, A. J. Stable time filtering of strongly unstable spatially extended systems. *Proceedings of the National Academy of Sciences of the United States of America 103*, 20 (2006), 7548–7553.

[42] Guendelman, E., Bridson, R., and Fedkiw, R. Nonconvex rigid bodies with stacking. *ACM Trans. Graph. 22*, 3 (2003), 871–878.

[43] Guendelman, E., Selle, A., Losasso, F., and Fedkiw, R. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph. 24*, 3 (2005), 973–981.

[44] Harada, T., Koshizuka, S., and Kawaguchi, Y. Real-time fluid simulation coupled with cloth. In *Theory and Practice of Computer Graphics* (2007), Eurographics Association, pp. 13–20.

[45] Harlow, F. H. The particle-in-cell method for numerical solution of problems in fluid dynamics. *In Experimental arithmetic, high-speed computations and mathematics* (1963).

[46] Harlow, F. H., and Welch, J. E. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids 8*, 12 (December 1965), 2182–2189.

[47] Hauth, M., Etzmuss, O., and Strasser, W. Analysis of numerical methods for the simulation of deformable models. *The Visual Computer 19*, 7-8 (2003), 581–600.

[48] Houston, B., Nielsen, M. B., Batty, C., Nilsson, O., and Museth, K. Hierarchical rle level set : A compact and versatile deformable surface representation. *ACM Trans. Graph. 25*, 1 (2006), 151–175.

[49] Iglesias, A. Computer graphics for water modeling and rendering : a survey. *Future Gener. Comput. Syst. 20*, 8 (2004), 1355–1374.

[50] Jakub Wejchert, D. H. Animation aerodynamics. *Computer Graphics 25*, 4 (Juillet 1991), 19–22.

[51] James D.Foley, Andries van Dam, S. K. F. J. F. H. e. R. L. *Introduction à l'infographie*, addison-wesley ed. Vuibert, 1999.

[52] Kass, M. An introduction to physically based modeling, chapter introduction to continuum dynamics for computer graphics. In *SIGGRAPH Course Notes* (1995).

[53] Kass, M., and Miller, G. Rapid, stable fluid dynamics for computer graphics. In *SIGGRAPH '90 : Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1990), ACM Press, pp. 49–57.

[54] Kačić-Alesić, Z., Nordenstam, M., and Bullock, D. A practical dynamics system. In *SCA '03 : Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, 2003), Eurographics Association, pp. 7–16.

[55] Keiser, R. *Meshless Lagrangian Methods for Physics-Based Animations of Solids and Fluids*. PhD thesis, Dissertation ETH Zurich, 2006.

[56] Klingner, B. M., Feldman, B. E., Chentanez, N., and O'Brien, J. F. Fluid animation with dynamic meshes. In *SIGGRAPH '06 : ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM Press, pp. 820–825.

[57] Koichi Onoue, T. N. An interactive deformation system for granular material. *Computer Graphics Forum 24*, 1 (2005), 51–60.

[58] Kondoh, N., Sasagawa, S., and Kunimatsu, A. Creating animations of fluids and cloth with moving characters. In *SIGGRAPH '04 : ACM SIGGRAPH 2004 Sketches* (New York, NY, USA, 2004), ACM Press, p. 136.

[59] Laramee, R. S., Hauser, H., Doleisch, H., Vrolijk, B., Post, F. H., and Weiskopf, D. The state of the art in flow visualization : Dense and texture-based techniques. *Comput. Graph. Forum 23*, 2 (2004), 203–222.

[60] Lefebvre, S., and Hoppe, H. Perfect spatial hashing. In *SIGGRAPH '06 : ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM Press, pp. 579–588.

[61] Lorensen, W. E., and Cline, H. E. Marching cubes : A high resolution 3d surface construction algorithm. In *SIGGRAPH '87 : Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), ACM Press, pp. 163–169.

[62] Losasso, F., Gibou, F., and Fedkiw, R. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. 23*, 3 (2004), 457–462.

[63] Losasso, F., Irving, G., and Guendelman, E. Melting and burning solids into liquids and gases. *IEEE Transactions on Visualization and Computer Graphics 12*, 3 (2006), 343–352. Member-Ron Fedkiw.

[64] Lucy, L. B. A numerical approach to testing the fission hypothesis. *aj 82*, 12 (December 1977), 1013–1924.

[65] Lueker, G., and Molodowitch, M. More analysis of double hashing. In *STOC '88 : Proceedings of the twentieth annual ACM symposium on Theory of computing* (New York, NY, USA, 1988), ACM Press, pp. 354–359.

[66] Maciel, A., Boulic, R., and Thalmann, D. Deformable tissue parameterized by properties of real biological tissue. In *IS4TH* (2003), pp. 74–87.

[67] Max, N., and Becker, B. Flow visualization using moving textures. In *Proceedings of the ICAS/LaRC Symposium on Visualizing Time-Varying Data* (1996), D. C. Banks, T. W. Crockett, and K. Stacy, Eds., NASA Conference Publication 3321, pp. 77–87.

[68] Max, N. L. Vectorized procedural models for natural terrain : Waves and islands in the sunset. In *SIGGRAPH '81 : Proceedings of the 8th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1981), ACM Press, pp. 317–324.

[69] Meyer, M., Debunne, G., Desbrun, M., and Barr, A. H. Interactive animation of cloth-like objects in virtual reality. *Journal of Visualization and Computer Animation 12*, 1 (2001), 1–12.

[70] Miller, G. S. P. The motion dynamics of snakes and worms. In *SIGGRAPH '88 : Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1988), ACM Press, pp. 169–173.

[71] Miller, G. S. P., and Pearce, A. Globular dynamics : A connected particle system for animating viscous fluids. *Computers & Graphics 13*, 3 (1989), 305–309.

[72] Mirtich, B. V. *Impulse-based dynamic simulation of rigid body systems.* PhD thesis, University of Berkeley, California, 1996.

[73] Monaghan, J. Smoothed particle hydrodynamics. *Annu. Rev. Astron. Astrophys. 30* (1992), 543–574.

[74] Mould, D., and Yang, Y.-H. Modeling water for computer graphics. *Computers & Graphics 21*, 6 (1997), 801–814.

[75] Müller, M., Charypar, D., and Gross, M. Particle-based fluid simulation for interactive applications. In *SCA '03 : Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, 2003), Eurographics Association, pp. 154–159.

[76] Müller, M., Schirm, S., Teschner, M., Heidelberger, B., and Gross, M. Interaction of fluids with deformable solids. *Comput. Animat. Virtual Worlds 15*, 3-4 (2004), 159–171.

[77] Müller, M., Solenthaler, B., Keiser, R., and Gross, M. Particle-based fluid-fluid interaction. In *SCA '05 : Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), ACM Press, pp. 237–244.

[78] Nealen, A., Müller, M., Keiser, R., Boxermann, E., and Carlson, M. Physically based deformable models in computer graphics. In *Proceedings of Eurographics 2005* (2005), pp. 91–94.

[79] O'Brien, J. F., and Hodgins, J. K. Dynamic simulation of splashing fluids. In *CA '95 : Proceedings of the Computer Animation* (Washington, DC, USA, 1995), IEEE Computer Society, p. 198.

[80] Perlin, K. An image synthesizer. In *SIGGRAPH '85 : Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1985), ACM Press, pp. 287–296.

[81] Policarpo, F., Oliveira, M. M., and Jo a. L. D. C. Real-time relief mapping on arbitrary polygonal surfaces. *ACM Trans. Graph. 24*, 3 (2005), 935–935.

[82] Premoze, S., Tasdizen, T., Bigler, J., Lefohn, A. E., and Whitaker, R. T. Particle-based simulation of fluids. *Comput. Graph. Forum 22*, 3 (2003), 401–410.

[83] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[84] Provot, X. Collision and self-collision handling in cloth model dedicated to design garments. In *Graphics Interface* (1997), pp. 177–189.

[85] Rasmussen, N., Enright, D., Nguyen, D., Marino, S., Sumner, N., Geiger, W., Hoon, S., and Fedkiw, R. Directable photorealistic liquids. In *SCA '04 : Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2004), ACM Press, pp. 193–202.

[86] Reeves, W. T. Particle systems - a technique for modeling a class of fuzzy objects. In *SIGGRAPH '83 : Proceedings of the 10th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1983), ACM Press, pp. 359–375.

[87] Rousseau, P., Jolivet, V., and Ghazanfarpour, D. Realistic real-time rain rendering. *Computers & Graphics 30*, 4 (2006), 507–518. special issue on Natural Phenomena Simulation.

[88] Selle, A., Rasmussen, N., and Fedkiw, R. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph. 24*, 3 (2005), 910–914.

[89] Shade, J., Gortler, S., wei He, L., and Szeliski, R. Layered depth images. In *SIGGRAPH '98 : Proceedings of the 25th annual conference on*

Computer graphics and interactive techniques (New York, NY, USA, 1998), ACM Press, pp. 231–242.

[90] Shinya, M. Theories for mass-spring simulation in computer graphics : Stability, costs and improvements. *IEICE - Trans. Inf. Syst. E88-D*, 4 (2005), 767–774.

[91] Sims, K. Particle animation and rendering using data parallel computation. In *SIGGRAPH '90 : Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1990), ACM Press, pp. 405–413.

[92] Stam, J. Stable fluids. In *SIGGRAPH '99 : Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 121–128.

[93] Stam, J., and Fiume, E. Depicting fire and other gaseous phenomena using diffusion processes. In *SIGGRAPH '95 : Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1995), ACM Press, pp. 129–136.

[94] Steele, K. L., Cline, D., Egbert, P. K., and Dinerstein, J. Modeling and rendering viscous liquids. *Journal of Visualization and Computer Animation 15*, 3-4 (2004), 183–192.

[95] Stora, D., Agliati, P.-O., Cani, M.-P., Neyret, F., and Gascuel, J.-D. Animating lava flows. In *Proceedings of the 1999 conference on Graphics interface '99* (San Francisco, CA, USA, 1999), Morgan Kaufmann Publishers Inc., pp. 203–210.

[96] Terzopoulos, D., and Fleischer, K. Modeling inelastic deformation : viscolelasticity, plasticity, fracture. In *SIGGRAPH '88 : Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1988), ACM Press, pp. 269–278.

[97] Terzopoulos, D., Platt, J., and Fleischer, K. Heating and melting deformable models. *The Journal of Visualization and Computer Animation 2*, 2 (1991), 68–73.

[98] Teschner, M., Heidelberger, B., Müller, M., Pomerantes, D., and Gross, M. H. Optimized spatial hashing for collision detection of deformable objects. In *VMV* (2003), pp. 47–54.

[99] Teschner, M., Kimmerle, S., Heidelberger, B., Zachmann, G., Raghupathi, L., Fuhrmann, A., Cani, M.-P., Faure, F., Magnenat-Thalmann, N., Strasser, W., and Volino, P. Collision detection for deformable objects. *Computer Graphics Forum 24*, 1 (March 2005), 61–81.

[100] Thon, S. *Représentation de l'eau en synthèse d'images : exemples de grandes étendues d'eau.* PhD thesis, Université de Limoges, Limoges, 2001.

[101] Thon, S., Dischler, J.-M., and Ghazanfarpour, D. Ocean waves synthesis using a spectrum-based turbulence function. In *CGI '00 : Proceedings of the International Conference on Computer Graphics* (Washington, DC, USA, 2000), IEEE Computer Society, p. 65.

[102] Thuerey, N. *Physically based Animation of Free Surface Flows with the Lattice Boltzmann Method*. PhD thesis, University of Erlangen-Nuremberg, 2007.

[103] Tonnesen, D. Modeling liquids and solids using thermal particles. In *Graphics Interface '91* (June 1991), pp. 255–262.

[104] Tu, X., and Terzopoulos, D. Artificial fishes : physics, locomotion, perception, behavior. In *SIGGRAPH '94 : Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1994), ACM Press, pp. 43–50.

[105] van Wijk, J. Flow visualization with surface particles. *IEEE Comput. Graph. Appl. 13*, 4 (1993), 18–24.

[106] Volino, P., and Magnenat-Thalmann, N. Implementing fast cloth simulation with collision response. In *CGI '00 : Proceedings of the International Conference on Computer Graphics* (Washington, DC, USA, 2000), IEEE Computer Society, p. 257.

[107] Volino, P., and Magnenat-Thalmann, N. Comparing efficiency of integration methods for cloth simulation. In *Computer Graphics International* (2001), pp. 265–274.

[108] Wang, H., Mucha, P. J., and Turk, G. Water drops on surfaces. *ACM Trans. Graph. 24*, 3 (2005), 921–929.

[109] Wei, X., Li, W., Mueller, K., and Kaufman, A. E. The lattice-boltzmann method for simulating gaseous phenomena. *IEEE Trans. Vis. Comput. Graph. 10*, 2 (2004), 164–176.

[110] Wei, X., Zhao, Y., Fan, Z., Li, W., Yoakum-Stover, S., and Kaufman, A. Blowing in the wind. In *SCA '03 : Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, 2003), Eurographics Association, pp. 75–85.

[111] Wei Li, Zhe Fan, X. W., and Kaufman, A. Gpu-based flow simulation with complex boundaries. Tech. Rep. 031105, SUNY at Stony Brook, Computer Science Department, November 2003.

[112] Welch, G., and Bishop, G. An introduction to the kalman filter. In *SIGGRAPH Courses Notes* (2001).

[113] Weyrich, T., Heinzle, S., Aila, T., Fasnacht, D. B., Oetiker, S., Botsch, M., Flaig, C., Mall, S., Rohrer, K., Felber, N., Kaeslin, H., and Gross, M. A hardware architecture for surface splatting. *ACM Transactions on Graphics (Proc. SIGGRAPH) 26*, 3 (Aug. 2007).

[114] Yaeger, L., Upson, C., and Myers, R. Combining physical and visual simulation - creation of the planet jupiter for the film '2010'. In *SIGGRAPH '86 : Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1986), ACM Press, pp. 85–93.

[115] Yuksel, C., House, D. H., and Keyser, J. Wave particles. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3 (2007), to appear.

[116] Zhu, Y., and Bridson, R. Animating sand as a fluid. *ACM Trans. Graph. 24*, 3 (2005), 965–972.

Résumé

Je présente dans cette thèse des travaux concernant la simulation d'interactions entre un fluide et des membranes déformables. Ce sujet ne manque pas d'applications puisque l'interaction d'un fluide avec des feuilles, des tissus ou même des nageoires de poisson est un scénario courant. Pourtant, peu de travaux en synthèse d'images se sont intéressés à reproduire ce phénomène.

Pour ce faire, je propose un algorithme de réaction de contacts entre un fluide Lagrangien et une membrane déformable qui est indépendant du pas d'intégration temporelle. Afin de pallier les artefacts visuels dus à des triangles, issus du processus d'extraction de surface du fluide, qui passent au travers de la membrane lors d'interactions, je présente une technique qui projette la portion de volume erronée le long de la membrane. De plus, une paramétrisation intuitive de coefficients hétérogènes de friction sur une surface est exposée.

Je présente également deux modèles d'optimisation. Le premier modèle propose de filtrer les vitesses de particules calculées par une méthode explicite afin de réduire la forte contrainte sur le pas d'intégration temporelle. Des analyses fréquentielles permettent de quantifier cette réduction. Le deuxième modèle est une structure de subdivision spatiale dynamique qui détermine efficacement le voisinage d'un point et d'un triangle. Cette structure a pour but de réduire les temps de calculs des processus de détermination du voisinage pour les fluides Lagrangiens, mais aussi ceux de la détection de collisions et enfin ceux de l'extraction de surface.

Abstract

The presented works' main focus is the interaction of liquids and thin shells, such as sheets of paper, fish fins and even clothes. Even though such interactions is an every day scenario, few research work in the computer graphics community have investigated this phenomenon.

Thereby, I propose an algorithm which resolves contacts between Lagrangian fluids and deformable thin shells. Visual artefacts may appear during the surface extraction procedure due to the proximity of the fluids and the shells. Thus, to avoid such artefacts, I propose a visibility algorithm which projects the undesired overlapping volume of liquid onto the thin shells' surface. In addition, an intuitive parametrisation model for the definition of heterogeneous friction coefficients on a surface is presented.

I also propose two optimisation methods. The first one reduces the well-known dependency of numerical stability and the timestep when using explicit schemes by filtering particles' velocities. This reduction is quantified with the use of frequency analysis. The second optimisation method is a unified dynamic spatial acceleration model, composed of a hierarchical hash table data structure, that speeds up the particle neighbourhood query and the collision broad phase. The proposed unified model is besides used to efficiently prune unnecessary computations during the surface extraction procedure.