

IFT3355 : Infographie
Hiver 2013
Travail Pratique #3 : 15%

Disponible : Jeudi 28 mars
Remise : Jeudi 18 avril au début de la démo
équipes : Deux étudiant(e)s
Questions : dift3355@iro.umontreal.ca

1 Description

Ce travail porte sur l'animation et la modélisation. Dans une première étape, vous avez à compléter la dernière étape de la simulation de l'écoulement d'un liquide qui consiste à déplacer chacune des particules du liquide à chaque pas de temps. Dans une deuxième étape, il vous faudra utiliser l'information fournie par la simulation pour reconstruire la surface du liquide en utilisant le concept des surfaces implicites et de l'algorithme *Marching Tetrahedra* (voir [Shirley et al. 2009]).

1.1 Animation du liquide

La simulation de l'écoulement du liquide se fait par la méthode *Smoothed Particle Hydrodynamics* (SPH). Cette technique divise la masse totale du liquide en plus petites masses, appelées *particules*, qui sont animées de telle sorte qu'elles respectent une équation différentielle. Sans l'explicitier, cette équation nous permet simplement de quantifier les différentes forces qui agissent sur chaque particule, telles que la force de pression et la force de viscosité, en tenant compte de ses voisins. La force de pression a tendance à éloigner les particules les unes des autres tandis que la viscosité a plutôt tendance à uniformiser la vitesse de chaque particule. Le calcul de ces forces se fait à chaque pas de temps. Ce calcul vous est fourni.

On note les différentes propriétés d'une particule i de la façon suivante :

- \vec{p}_i : position,
- \vec{v}_i : vitesse,
- \vec{a}_i : accélération,
- \vec{F}_i : forces totales,
- m_i : masse,
- ρ_i : densité.

Ce que vous avez à implémenter est le déplacement de chaque particule en tenant compte des forces précédemment calculées. Il vous faut utiliser la méthode d'Euler semi-explicite pour intégrer le système défini par la deuxième loi de Newton :

$$\begin{aligned}
\mathbf{F}_i &= m_i \mathbf{a}_i \\
\mathbf{a}_i &= \frac{d\mathbf{v}_i}{dt} \\
\mathbf{v}_i &= \frac{d\mathbf{p}_i}{dt}.
\end{aligned}
\tag{1}$$

Le calcul des forces vous donne directement l'accélération de la particule. La méthode d'Euler semi-explicite consiste à calculer l'intégrale d'une fonction à l'aide d'une estimation linéaire à un temps t par un pas de temps Δt . Concrètement, la méthode se décrit comme ceci :

$$\begin{aligned}
\mathbf{v}_i^{(t+\Delta t)} &= \mathbf{v}_i^{(t)} + \Delta t \cdot \mathbf{a}_i^{(t)} \\
\mathbf{p}_i^{(t+\Delta t)} &= \mathbf{p}_i^{(t)} + \Delta t \cdot \mathbf{v}_i^{(t+\Delta t)}.
\end{aligned}
\tag{2}$$

Pour éviter que les particules traversent les parois du domaine de simulation, il vous faut vérifier s'il y a intersection entre le conteneur et le segment défini par les points $\mathbf{p}_i^{(t)}$ et $\mathbf{p}_i^{(t+\Delta t)}$. S'il y a intersection, vous devez projeter la position et la vitesse de la particule au plan du point d'intersection et lui faire continuer son parcours le long de sa nouvelle direction tel que démontré par la figure 1. Cette opération est appliquée itérativement jusqu'à ce qu'il n'y ait plus d'intersection et que la particule ait atteint sa destination finale. Ici, l'important est seulement que la position de la particule ne dépasse pas les parois, sans considérer le volume de cette particule.

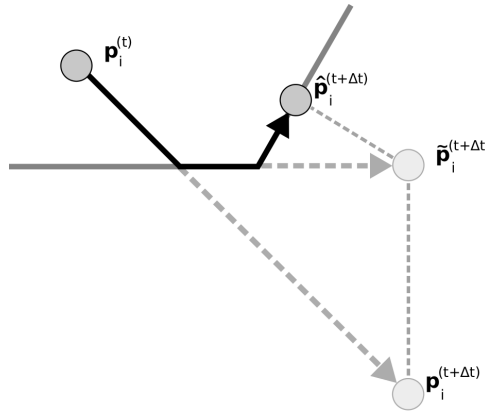


FIGURE 1 – Déviation d'une particule frappant une paroi. Au point de contact, la trajectoire est modifiée afin que la vitesse soit parallèle à la paroi. $\mathbf{p}_i^{(t)}$ est la position initiale, $\mathbf{p}_i^{(t+\Delta t)}$ est la position finale et $\tilde{\mathbf{p}}_i^{(t+\Delta t)}$ et $\hat{\mathbf{p}}_i^{(t+\Delta t)}$ sont les positions successivement corrigées.

La recherche des particules voisines pour le calcul des forces se fait à l'aide d'une grille régulière implémentée dans la classe `Grid`. Chaque particule possède une propriété `CellIndex` qui indique dans quelle cellule de la grille elle se trouve. Lors du déplacement d'une particule, vous devez vérifier si la particule appartient encore à la même cellule. Si elle a changé, vous devez la mettre à jour. Ces tests doivent utiliser les méthodes de la classe `Grid`.

1.2 Construction de l'interface air-liquide

Jusqu'à ce point, vous n'avez qu'un ensemble de particules. Il vous faut maintenant construire une surface représentant le plus fidèlement possible l'interface air-liquide. Une technique permettant ce genre de construction s'appelle le *Marching Tetrahedra*. À partir d'une surface implicite définie par l'équation $f(\mathbf{r}) = 0$, elle vous retourne un maillage approximant l'interface.

1.2.1 Surface implicite

Il faut premièrement définir cette fonction implicite. La fonction f est construite de telle sorte qu'on sache qu'on est à l'intérieur du liquide lorsque $f(\mathbf{p}) > 0$ et à l'extérieur lorsque $f(\mathbf{p}) < 0$. Une fonction qui satisfait à cette contrainte pourrait utiliser la densité. En calculant la densité à un point \mathbf{p} , on peut la comparer avec la densité du liquide au repos et quantifier cette différence. En utilisant la formulation SPH, la densité $\rho(\mathbf{p})$ à un point \mathbf{p} quelconque se calcule de la façon suivante :

$$\rho(\mathbf{p}) = \sum_i m_i W\left(\left(\mathbf{p} - \mathbf{p}_i\right)^2, h\right) \quad (3)$$

où la somme est effectuée pour toutes les particules à l'intérieur d'une sphère de rayon h et W est la fonction servant à pondérer les valeurs de chaque particule du voisinage. Cette fonction s'appelle le *noyau* et se trouve dans le code fourni.

En notant ρ_0 la densité du liquide au repos, on peut estimer que $\rho(\mathbf{p})/\rho_0 \approx 1$ à l'intérieur du liquide. Cette valeur aura tendance à chuter en s'approchant de l'interface car le nombre de particules entrant dans le calcul diminuera. C'est pourquoi on estimera qu'on se situe à la frontière si $\rho(\mathbf{p})/\rho_0 = 1 - a$, où a est une valeur positive que vous choisirez. Le prototype utilise $a = 0.3$, mais d'autres valeurs peuvent aussi donner de bons résultats. Si on combine le tout, on obtient la fonction définissant notre surface implicite comme ceci :

$$f(\mathbf{p}) = \frac{\rho(\mathbf{p})}{\rho_0} - (1 - a). \quad (4)$$

1.2.2 *Marching Tetrahedra*

Cette technique se base sur un échantillonnage régulier de l'espace afin d'évaluer la fonction f . Pour cette première étape, il s'agit de discrétiser un cube englobant la surface pour en former une grille régulière. Les sommets de cette grille ayant une position dans l'espace, on peut évaluer la fonction f à chacun d'eux.

La deuxième étape consiste à parcourir chacun des cubes de la grille pour les subdiviser en objets plus simples : des tétraèdres. Il faut porter attention à la façon dont elle est effectuée, car chaque cube doit être compatible avec les cubes voisins. C'est-à-dire que si une face du cube est subdivisée dans une direction, sa face voisine doit être subdivisée dans la même direction. Une subdivision qui respecte cette contrainte est illustrée dans la figure 2a.

On doit maintenant déterminer si la surface passe par chacun des tétraèdres de la subdivision. On peut distinguer trois cas illustrés par la figure 2b. Le cas le plus simple est lorsque les quatre valeurs aux sommets du tétraèdre sont de même signe. Ça signifie que le tétraèdre est complètement à l'intérieur ou à l'extérieur du liquide. Il n'y a donc rien à faire. Le deuxième cas apparaît lorsqu'un seul signe est différent. On peut donc déduire que sur chacune des trois arêtes incidentes au sommet, il y aura un endroit tel que $f(\mathbf{p}) = 0$.

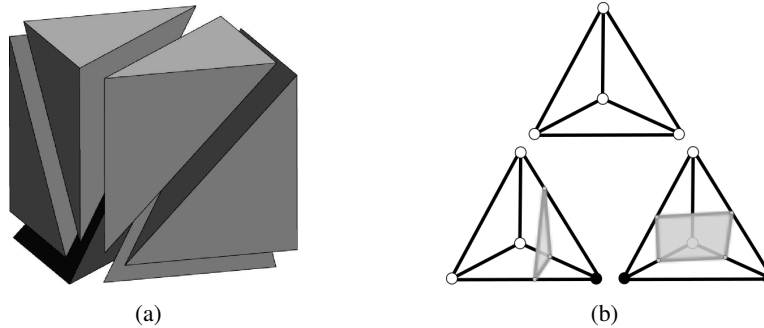


FIGURE 2 – (a) Division d’un cube en six tétraèdres de volume égal. (b) Trois cas distincts : (haut) aucun changement de signe, (bas-gauche) un signe est différent et (bas-droite) deux signes sont différents.

Sans connaître exactement cette position, on peut l’estimer par une interpolation linéaire. Ces trois positions forment un triangle qu’il vous faudra afficher. Le dernier cas est semblable, à l’exception qu’on obtiendra un quadrilatère, pas nécessairement planaire, au lieu d’un triangle. Il suffit de le subdiviser en deux triangles et de les afficher. La direction de subdivision du quadrilatère n’a pas vraiment d’importance ici.

Vous avez maintenant un maillage approximant la surface. Il vous manque cependant les normales à chacun des sommets du maillage. La méthode traditionnelle serait de moyenner les normales des faces incidentes à chaque sommet. Mais aucune information de connexion entre les faces n’est gardée, ce qui vous empêche de procéder ainsi. Heureusement, la formulation par surface implicite nous permet de définir une normale à chacun des points de l’espace. Comme nous l’avons vu dans le cours, la normale d’une surface implicite à un point \mathbf{p} est définie par le gradient de la fonction :

$$n(\mathbf{p}) = -\nabla f(\mathbf{p}). \quad (5)$$

Il suffit maintenant d’interpoler ces valeurs de normales pour chaque sommet de chaque triangle de la même façon qu’on a interpolé leur position.

2 Bonus

Un point supplémentaire peut être obtenu pour l’ajout d’un flotteur sphérique dans le fluide. Ce flotteur possède un volume et une masse, est affecté par la gravité et doit interagir avec le fluide. Au contact du flotteur, les particules du fluide doivent être repoussées et doivent lui appliquer une force de répulsion en retour. Ceci aura pour effet de garder le flotteur à la surface de l’eau.

Vous pouvez le simuler dans un domaine sphérique uniquement.

3 Support au développement

Nous vous fournissons un code de base que vous devez compléter. Si vous modifiez l’interface, vous devez fournir les mêmes fonctionnalités de base afin de faciliter la correction. *A priori*, vous n’avez qu’à modifier les endroits avec des commentaires pour IFT3355 dans certains fichiers. Suivez ces commentaires pour savoir quoi mettre où. Vous pouvez bien entendu ajouter de nouvelles fonctions, méthodes ou classes. L’archive du code est disponible sur le site web du cours.

Un exécutable vous permet d'avoir un aperçu du comportement que devrait avoir votre programme.

Les fichiers qui vous intéressent sont `SPH.cpp` et `MarchingTetrahedra.cpp`. Les intersections se calculent de façon générique sur n'importe quelle géométrie donnée et s'utilisent comme dans le deuxième travail pratique. Le code d'intersection vous est fourni.

4 Fonctionnalité

Lorsque démarré, il ouvre une fenêtre d'affichage où il est possible de visualiser la scène interactivement en mode OpenGL. La caméra se promène autour de son point d'intérêt. Voici la description des touches et contrôles :

<code>p</code>	:	Met l'animation en pause
<code>0</code>	:	Remet à zéro la vitesse des particules
<code>m</code>	:	Active/désactive l'affichage de la surface par <i>Marching Tetrahedra</i>
<code>r</code>	:	Active/désactive l'effet de réfraction approximative du liquide
<code>espace+souris</code>	:	Applique une rotation au contenant

5 Délivrables et rapport du travail

Nous aurons besoin d'une version électronique de votre programme pour l'évaluer. La remise se fera par StudiUM.

Vous devez aussi écrire un rapport d'environ 3-4 pages qui contiendra :

1. Énoncé du problème à résoudre.
2. Description des techniques utilisées. Donnez les formules mathématiques utilisées, et comment vous les avez employées. Nous voulons ici le fondement de vos techniques, pas les détails de la disposition du code.
3. Problèmes liés aux techniques implémentées, suggestions pour améliorer la solution au problème, extensions possibles.
4. Références.

6 Critères d'évaluation

Section	Critères	Pondération
Code	style, clarté, <i>efficacité</i> , résultats	80%
Rapport	esthétique, énoncé, descriptions	10%
Analyse	problèmes, améliorations, extensions	10%

7 Bibliographie

P. Shirley et S. Marschner. *Fundamentals of Computer Graphics*. 3ème édition, AK Peters, 2009.

M. Müller, D. Charypar et M. Gross. *Particle-based fluid simulation for interactive applications*. In Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '03), pp. 154-159, 2003.