Département d'Informatique et de Recherche Opérationnelle
Université de Montréal

Pierre Poulin
Bureau : 2389
poulin@iro.umontreal.ca

IFT3355 : Infographie
Hiver 2011
Travail Pratique #3 : 15%

Disponible :	Jeudi 17 mars	
Remise :	Jeudi 7 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'expliciter, 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 vélocité 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,
- $\vec{v}_i$: vélocité,
- $\vec{a}_i$: 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 projecter la position et la vélocité 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 la paroi, sans considérer le volume de cette particule.

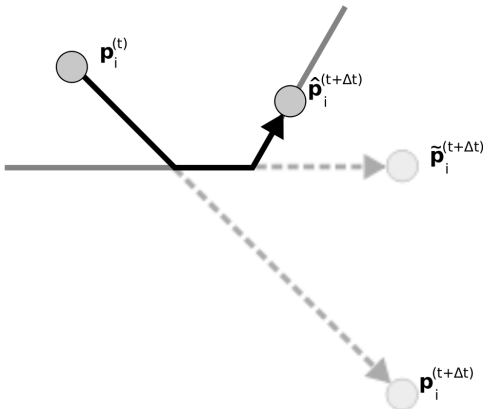


FIGURE 1 – Déviation d'une particule frappant une paroi. Au point de contact, la trajectoire est modifiée afin que la vélocité soit parallèle à la paroi. $\mathbf{p}_i^{(t)}$ est la position initiale, $\mathbf{p}_i^{(t+\Delta t)}$ est la position finale et $\tilde{\mathbf{p}}_i^{(t+\Delta t)}$ et $\hat{\mathbf{p}}_i^{(t+\Delta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

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

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 pouquoi 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 :

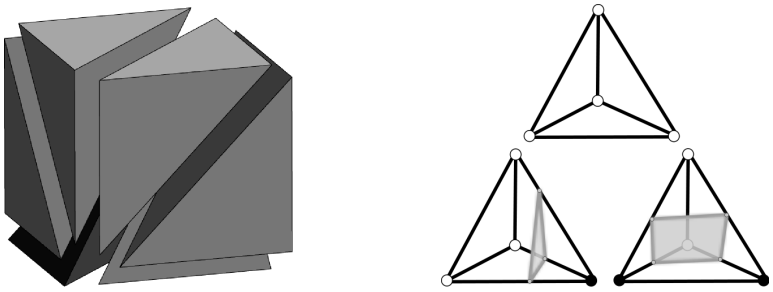
$$f(\mathbf{p}) = \frac{\rho(\mathbf{p})}{\rho_0} - (1 - a). \tag{4}$$

1.2.2 *Marching Tetrahedra*

Cette technique se base sur un échantillonage 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 apparait 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 $f(\mathbf{p}) = 0$.



(a) (b)

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 connaitre 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.

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 comme ceci. Heureusement, la formulation par surface implicite nous permet de définir une normale à chacun des points de l'espace. De la même façon qu'on a calculé les valeurs de f, on calculera une approximation de la normale à chacun des sommets de la grille par la formule suivante :

$$n(\mathbf{p}) = \sum_i (\mathbf{r} - \mathbf{r}_i)\, W\left((\mathbf{r} - \mathbf{r}_i)^2, h\right). \tag{5}$$

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 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 au chemin `~dift3355/pub/tp/tp3/tp3.tar.gz`. Pour extraire l'archive, faites ceci :

```
% tar zxf ~dift3355/pub/tp/tp3/tp3.tar.gz
```

Un exécutable (`prototype{32,64}`) vous permet d'avoir un aperçu du comportement que devrait avoir votre programme.

Pour compiler, utilisez la commande `make debug`. Si vous voulez compiler la version optimisée, exécutez plutôt `make release`. Votre programme fera l'édition des liens avec un librairie statique. Prenez

soin de sélectionner la bonne version (32 ou 64 bits) dans le makefile, à la deuxième ligne, afin d'éviter des incompatibilités. Pour exécuter le programme, la commande est :

```
% ./bin/debug/tp3     # si compilé en mode debug
```

ou

```
% ./bin/release/tp3   # si compilé en mode release
```

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, mais sera caché dans une librairie statique.

3 Fonctionnalité

Usage : `./bin/debug/tp3 [scene]`
ou
`./bin/release/tp3 [scene]`

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 :

`Esc`	:	Quitte
`p`	:	Met l'animation en pause
`0`	:	Remet à zéro la vélocité des particules
`c`	:	Active/désactive l'affichage du contenant
`m`	:	Active/désactive l'affichage de la surface par *Marching Tetrahedra*
`r`	:	Active/désactive l'effet de réfraction approximative du liquide
`w`	:	Active/désactive l'affichage de la surface en fil de fer
`z+souris`	:	Applique une rotation au contenant

Les fichiers de description de scène sont en format texte. Consultez les différents fichiers situés dans le dossier `data/scenes` pour avoir des exemples de création de tels fichiers. Les géométries avec maillage se trouvent dans des fichiers séparés. Le format accepté est *obj*, que vous pouvez éditer avec le logiciel de modélisation gratuit *blender*.

Pour les contenants de type `Mesh`, les particules sont générées aléatoirement dans une sphère de rayon 0.5 centrée à l'origine. Il vous faudra donc tenir cette particularité en compte si vous voulez créer votre propre contenant.

4 Délivrables et rapport du travail

Nous aurons besoin d'une version électronique de votre programme pour l'évaluer. Pour la remise, faites les commandes suivantes :

```
% ssh remise
% cd <repertoire racine du projet>
% make dist                          # Crée l'archive à remettre
% tar zft tp3.tar.gz                 # Vérifie le contenu de l'archive
% remise ift3355 tp2 tp3.tar.gz
% remise -v ift3355 tp2 tp3.tar.gz  # Vérifie la remise
% exit
```

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 reliés aux techniques implémentées, suggestions pour améliorer la solution au problème, extensions possibles.
4. Références.

5 Critères d'évaluation

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

6 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.